04-538-89
05/40 Kčs 16,—

ISBN 80-03-00116-1

PROVOZ A ÚDRŽBA POČÍTAČŮ

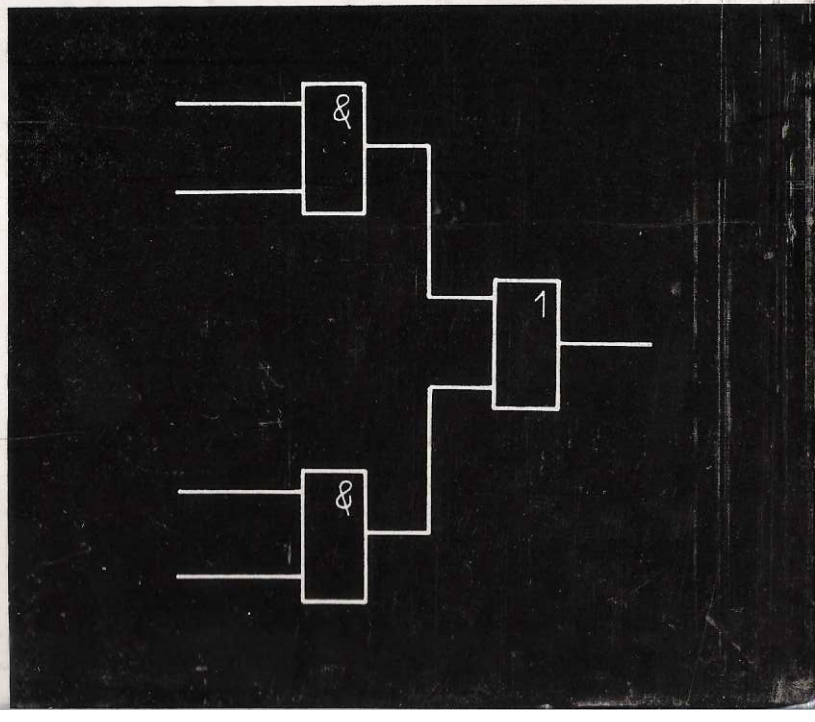
P. PŘÍVĚTIVÝ A KOLEKTIV

SNTL



# Provoz a údržba počítačů

P. PŘÍVĚTIVÝ A KOLEKTIV



ING. PAVEL PŘÍVĚTIVÝ
A KOLEKTIV

# Provoz a údržba počítačů

*Schválilo ministerstvo školství ČSR dne 11. února 1988 č. j.: 9 724/88-211 jako učebnici pro předmět Provoz a údržba počítačů vyučovaný na středních průmyslových školách ve 4. ročníku studia oboru 26-60-6 Elektronická a sdělovací zařízení, AB: Elektronické počítačové systémy.*

PRAHA 1989

SNTL – NAKLADATELSTVÍ
TECHNICKÉ LITERATURY

Učebnice se zabývá provozem a údržbou jednotlivých částí číslicového počítače a měřením, diagnostikou a modelováním činnosti obvodů mikroprocesorové techniky.

Je určena pro předmět Provoz a údržba počítačů vyučovaný na středních průmyslových školách s výukou studijního oboru 26-60-6 Elektronická a sdělovací zařízení, AB: Elektronické počítačové systémy.

Kolektiv autorů: Ing. Svatopluk Hekal (5. kapitola)
Ing. Petr Krejcar (3. a 7. kapitola)
Ing. Pavel Přívětivý (1., 4. a 8. kapitola)
Ing. Karel Šimerda (2. a 6. kapitola)
Ing. Ivan Šup (3. a 7. kapitola)



Lektoři: Ing. Dag Jeger, CSc.
Ing. Petr Kroha, CSc.

Redakce elektrotechnické literatury
Hlavní redaktor Ing. Josef Říha
Odpovědný redaktor Ing. Milan Veselý



ISBN 80-03-00116-1

# Obsah

# Předmluva

Tato učebnice je určena pro žáky čtvrtého ročníku studijního oboru elektronická a sdělovací zařízení, alternativní blok elektronické počítačové systémy. Vzhledem k tomu, že školy, na kterých se počítačové systémy vyučují, jsou vybaveny počítači řady ADT, je část praktických cvičení prováděna na tomto typu počítače. Další cvičení jsou prováděna na mikropočítačích IQ 151 nebo TEMS 8003 s přídavnou deskou TEMS 8006. Těmito mikropočítači jsou příslušné školy také vybaveny. Pokud by některá škola chtěla používat jiný typ počítače, bylo by nutné programy uvedené v této knize upravit. Jinak na náplni cvičení není potřeba nic měnit.

Učebnice je rozdělena na dvě části. První část, tj. kapitoly 1 až 4, je určena pro teoretickou výuku. Zbývající kapitoly popisují vlastní praktická cvičení. Každé cvičení by mělo trvat tři vyučovací hodiny. Praktických cvičení je zde uvedeno více, než kolik jich žáci stihnou provést během školního roku. Z těchto cvičení může učitel vybrat ta, která budou vhodná z hlediska vybavení školy. Další možnost je, že některá cvičení budou sloučena do jednoho cvičení.

Programy uvedené v této učebnici dodá na požádání školní výpočetní středisko při SPŠE v Pardubicích. Program uvedený v druhé části kapitoly 8 (simulace mikroprocesoru 8080) není pro praktické používání vhodný. Je napsán v jazyku BASIC a měl by sloužit žákům pouze pro pochopení principu, na kterém pracuje editor, překladač jazyka symbolických adres a ladicí program. Pro vlastní cvičení na mikropočítači IQ 151 je vhodné použít modul AMOS-Asembler.

# 1. Základní jednotka počítače

## 1.1. POPIS OPERAČNÍHO KÓDU

*Počítače řady JSEP* pracují se standardním souborem instrukcí (operačním kódem), doplněným u vyšších modelů této řady speciálními privilegovanými instrukcemi. Instrukce mají 6 možných formátů, označených RR, RX, RS, SI, SS, S. Jim odpovídá i různá délka instrukce 2, 4 nebo 6 bytů (slabik). Z hlediska počtu adres operandů použitých v instrukci jde o instrukce jednoadresové, dvouadresové nebo tříadresové. Použitý způsob adresování umožňuje efektivně adresovat velký rozsah operační paměti. Typickým operandem instrukce může být;

a) dvojkové číslo o délce 32 bitů;
b) desítkové číslo s maximálně 32 číslicemi;
c) abecedně číslicové pole s délkou u některých instrukcí až 16 Mbytů;
d) číslo v pohyblivé řádové čárce vyjádřené:
– s malou přesností – mantisa 24 bitů,
– s vyšší přesností – mantisa 56 bitů,
– s vysokou přesností – mantisa 112 bitů.

Délka exponentu je ve všech případech 7 bitů, jeden bit je použit pro znaménko.

Soubor instrukcí lze rozdělit na:
– uživatelské instrukce;
– privilegované instrukce.

Privilegované instrukce nelze na rozdíl od uživatelských instrukcí použít v uživatelském programu, používají se v operačním systému. Počet instrukcí např. u počítače EC 1025 je 175 (z toho 26 privilegovaných instrukcí).

U *počítačů řady SMEP* jsou periferní zařízení i operační paměť připojeny na společnou sběrnici. Vstupní/výstupní instrukce, tj. instrukce pro přenos informace mezi registrem základní jednotky a registrem v periferním zařízení (nebo naopak), jsou zde jen zvláštním případem instrukcí pro přenos mezi registrem a operační pamětí. Délka instrukce je 1, 2 nebo 3 slova. Pro adresování operandů lze použít osm různých způsobů adresování.

V této učebnici (kapitoly 2, 5 a 6) se budeme zabývat *počítači řady ADT*. Délka slova těchto počítačů je 16 bitů.
Instrukce se zde dělí na tyto skupiny:

- instrukce s odkazem na paměť;
- registrové instrukce;
- vstupní/výstupní instrukce.

U paměťových instrukcí tvoří adresní část instrukce 10 bitů, které určují adresu operandu v rámci stránky paměti. Operační paměť je zde tedy rozdělena na části po 1 024 slovech, kterým říkáme stránky. Při adresování se může použít běžná stránka (stránka, ve které leží prováděná instrukce), nultá stránka (stránka s adresami 0 až 1 023) nebo lze použít nepřímé adresy.

Operační kód počítače ADT 4000 je uveden v dodatku 3.

## 1.2. ZÁKLADNÍ PROGRAMOVÉ VYBAVENÍ

### Operační systém

Přímé používání počítače bez pomocného programového vybavení k výpočtům je velmi těžkopádné a působilo by řadu potíží. Proto byl mezi počítačem a uživatelem vytvořen mezičlánek nazvaný operační systém.

Operačních systémů existuje více typů. Např. jestliže operační systém umožňuje zpracovávat současně více úloh, mluvíme o *multiprogramovém* operačním systému, v opačném případě nazýváme systém *monoprogramový*.

Základem operačního systému je tzv. *jádro*, které mívá tyto základní funkce:

- přiděluje základní jednotku jednotlivým úlohám;
- přiděluje paměť;
- uskutečňuje základní operace s periferními zařízeními.

Funkce jádra se vyvolávají pomocí signálů přerušení a u minipočítačů a mikropočítačů skoky do podprogramu.

Původně se počítač používal bez operačního systému. Brzy se však ukázalo jako výhodné vybavit počítač jednoduchým monoprogramovým operačním systémem pro usnadnění práce s periferními zařízeními. Jako zařízení pro vstup řídicích příkazů byl pro operační systém obvykle vyhrazen psací stroj. Největší nevýhodou této koncepce bylo velmi malé využití základní jednotky (mnoho času se ztratí manipulací s paměťovými médii



a přemýšlením uživatele). Aby se tato nevýhoda odstranila, byl navržen operační systém pro nepřímý přístup uživatele, pracující v režimu *zpracování po úlohách*. Uživatel již není přítomen u počítače při zpracování, ale pouze předá např. sadu děrných štítků se zadáním úlohy a po zpracování dostane výstup z tiskárny s výsledky a protokolem o zpracování. Zde je využití základní jednotky vyšší, ale vznikají „prostoje" způsobené čekáním na ukončení práce periferních zařízení. Kdybychom pro vstup a výstup použili periferní zařízení s vyšší přenosovou rychlostí, bylo by i využití základní jednotky větší. To je základní myšlenka operačního systému pracujícího v režimu *zpracování v dávkách*. Princip spočívá v tom, že paměťovým médiem pro vstup úloh je magnetická páska, na kterou se nejprve na malém počítači okopíruje několik sad štítků tvořících zadání úloh (vytvoří se dávka úloh). Výsledky se zapisují na jinou magnetickou pásku, jejíž obsah se pak na malém počítači vypíše na tiskárně. Sekvenčností zápisu na magnetické pásce je omezena možnost dát zpracování některých úloh přednost před ostatními. Lze to odstranit tím, že místo magnetických pásek použijeme vnější paměť s adresovým výběrem (např. magnetickou diskovou paměť). V tomto případě je ale výhodnější opustit koncepci s malým počítačem a použít opět jediný počítač, v němž necháme probíhat tři procesy současně. V rámci jednoho procesu bude probíhat přepis zadání úloh z děrných štítků na magnetický disk, v rámci druhého procesu bude probíhat přepis výsledků a protokolů o zpracování úlohy z magnetického disku na tiskárnu. Třetí proces bude vlastní výpočet. Máme-li jednoprocesorový počítač, pak se tyto procesy střídají, a běží tedy zdánlivě současně. U tohoto typu operačního systému jsou snímač děrných štítků a tiskárna vlastně modelovány na vnější paměti s adresovým výběrem (technika nazývaná *spooling*).

Ještě většího využití základní jednotky dosáhneme, upravíme-li předchozí operační systém na multiprogramový. Tak dostaneme operační systém pro přímý přístup (uživatel má k dispozici pro zadávání svých úloh a získávání výsledků přímo periferní zařízení počítače). Požaduje-li se konverzační přístup, nejsou terminály používány technikou spooling. Takový operační systém označujeme jako *multiprogramový operační systém pro konverzační přístup* (používá se i termín *systém sdílení času*).

### Programovací jazyky

Pokud není programovací jazyk „mateřským" jazykem počítače, na němž se má výpočet provádět, je třeba přeložit program z původního,

tzv. *zdrojového* tvaru do tvaru *cílového*. Překlad se provádí programem nazvaným *překladač*. Cílovým programem může být buď přímo jazyk strojových instrukcí počítače, který bude provádět výpočet *(kompilační překlad)*, nebo mezijazyk, jehož příkazy budou při výpočtu interpretovány zvláštním programem *(interpretační překlad)*.

V každém programovacím jazyku jsou vymezeny určité *typy dat*, které charakterizují obory hodnot datových objektů a přípustné operace nad těmito objekty. Mezi základní typy dat patří typ celých čísel, typ reálných čísel atd. Je pro ně charakteristické, že z hlediska operací jsou dále nedělitelné, a proto se jim říká *jednoduché typy*, na rozdíl od typů, u nichž se údaj skládá ze složek (prvků) a které se nazývají *strukturované typy*.

*Datové objekty* v programovacích jazycích jsou konstanty (jejich hodnota se nemění) a proměnné (jejich hodnota se může měnit). Proměnným musí být v cílovém programu vyhrazen paměťový prostor. Podle toho, kdy se toto přidělení paměti provádí a kdo jej provádí, mluvíme o těchto druzích deklarací:

– *statická deklarace* – přidělování provádí překladač při překladu programu;

– *dynamická deklarace* – přidělování je generováno překladačem, ale provádí se při výpočtu;

– *dynamická deklarace ve volné paměti* – přidělování se provádí při výpočtu podle programu uživatele a pod jeho řízením.

Fyzicky se program dělí na části. Účelem členění je rozdělit program na nezávislé části tak, aby buď mohly být samostatně překládány *(programové moduly)*, nebo abychom omezili platnost proměnných na danou část programu – na tzv. *blok*. Proměnné mající platnost pouze uvnitř bloku nebo modulu se nazývají *lokální*, ostatní proměnné jsou *globální*.

Dále se stručně zmíníme o některých programovacích jazycích.

*Jazyk symbolických adres*

Tento jazyk se ze všech jazyků nejvíce podobá jazyku příslušného počítače, kterému říkáme *strojový jazyk*. Od strojového jazyku se jazyk symbolických adres liší především tím, že operační znaky jednotlivých instrukcí jsou označeny mnemotechnickou zkratkou a adresy jsou symbolicky vyjádřeny pomocí identifikátorů. Programování v jazyku symbolických adres je značně pracné, ale počítač je při výpočtu maximálně využit.



*Basic*

Jazyk Basic vznikl v roce 1965 s cílem umožnit programování těm lidem, kteří nejsou profesionálními programátory. S tímto jazykem jste se seznámili ve 2. ročníku v předmětu Výpočetní technika.

*Algol 60*

Jazyk Algol 60 byl navržen jako jazyk pro zápis vědeckotechnických algoritmů, popř. statisticko-ekonomických výpočtů. Máme zde k dispozici dynamická pole, tj. pole s libovolným počtem rozměrů a proměnnými dolními i horními mezemi indexů jednotlivých rozměrů. Prvky pole, stejně jako jednoduché proměnné, mohou být typu celočíselného, reálného nebo logického. Jazyk Algol 60 je z hlediska datových struktur chudý jazyk. Postrádáme zde především možnost pracovat se záznamy, seznamy a množinami. Nespornou výhodou tohoto jazyka je bloková struktura.

*Fortran*

Jazyk Fotran je podobně jako Algol 60 určen především pro vědeckotechnické a statisticko-ekonomické výpočty. Je zde k dispozici také pole, které smí být ale maximálně třírozměrné a statické. Typy dat jsou bohatší než v Algolu. Navíc je zde typ komplexních čísel a typ reálných čísel zobrazených s dvojnásobnou přesností.

*Cobol*

Jazyk Cobol se používá pro zápis programů pro hromadné zpracování dat. Je u něho možné používat heterogenní pole (záznam), tj. pole prvků různých typů. Prvky záznamu tvoří hierarchickou strukturu.

*PL/1*

Jazyk PL/1 vznikl v letech 1963 – 65 jako jazyk vyššího typu postavený na zkušenostech s jazyky Algol, Fortran a Cobol. Jazyk má blokovou strukturu.

*Pascal*

Jazyk Pascal byl definován v roce 1972. Vychází z dobrých vlastností jazyků PL/1, Cobol a Algol 60, ale žádný z nich není jeho podmnožinou. Hlavní výhodou jazyka Pascal je pestrost typů, s nimiž programátor může pracovat. Přípustné typy jsou skalár, interval, množina, pole, záznam, soubor a ukazatel. Deklarace proměnných může být statická nebo dynamická.

*ADA*

Programovací jazyk ADA patří mezi novinky roku 1979. Stavba tohoto jazyka vychází ze zásad strukturovaného a modulárního programování. Stejně jako v jazyku Pascal je zde mnoho typů dat. Dají se dokonce definovat operace pro další datové typy. Deklarace proměnných může být statická nebo dynamická.

*Lisp*

Jazyk Lisp patří mezi speciální programovací jazyky. Je určen především pro úlohy vedoucí ke zpracování seznamů nebo symbolů. Příkladem využití tohoto jazyka je např. zjednodušování algebraických výrazů vyjádřených v symbolické formě. Jazyk se používá i pro řešení úloh umělé inteligence.

*Simula 67*

Jazyk Simula 67 je bohatou nadstavbou jazyka Algol 60. Je určen především pro tyto účely:

- simulace systémů popsaných pomocí diskrétních událostí;
- manipulace s texty;
- zpracování seznamů;
- implementace dalších specializovaných programovacích jazyků.

## 1.3. METODY PRO DIAGNOSTIKU ZÁKLADNÍ JEDNOTKY

Smyslem používání diagnostiky u číslicového počítače je především potřeba získat dostatečné množství informací o skutečném technickém stavu počítače. Prostředkem pro získání těchto informací jsou *diagnostické testy*. Diagnostické testy je možné rozdělit na mikroprogramové a programové, takže při jejich použití mluvíme o mikrodiagnostice a programové diagnostice. Na začátku diagnostického testu je nutné předpokládat, že testujeme počítač, u něhož nemůžeme zaručit správnost funkce žádné části.



Diagnostiku obvykle zahajujeme *mikroprogramovými testy*. Aby bylo možné mikroprogram spustit, je třeba zaručit správnou funkci několika základních funkčních bloků počítače, tzv. tvrdého jádra. Do tvrdého jádra jsou zařazovány ty funkční bloky, jejichž správná činnost je nezbytná pro provádění několika nejjednodušších funkcí počítače. Test tvrdého jádra je nutné provádět ručně, pomocí vnějších diagnostických pomůcek (osciloskopy, sondy, jednoúčelové simulátory apod.).

Po oživení tvrdého jádra je počítač schopen plnit spuštěný mikroprogram, tj. vybírat z pevné paměti mikroinstrukce, přesouvat je do registru mikroinstrukce a volit následující adresu mikroinstrukce na základě provedení minulé mikroinstrukce.

Nejčastěji používaný postup testování se nazývá postupné oživování počítače. Jeho základem je postupné testování jednotlivých funkčních bloků základní jednotky, přičemž každý blok, jehož test byl ukončen, je zahrnut do oživené části a může být použit při testu zbývajících bloků. Výhodou mikroprogramových testů je možnost provést úplný test technického vybavení celé základní jednotky počítače.

K programovému vybavení každého počítače patří *testovací programy*, a to i v případě, že je počítač vybaven mikrodiagnostikou. Pro funkce, které lze ověřit programově, je výhodnější použít programovou diagnostiku (získané informace lze vyhodnotit a vypsat ve srozumitelné formě). Programová diagnostika je starší než mikroprogramová. Během jejího používání bylo vyvinuto mnoho variant její realizace.

Testovací programy představují obvykle dosti rozsáhlý soubor programů, které musí být uloženy v nějaké knihovně. Tato knihovna bývá umístěna na paměťovém médiu, obvykle na magnetickém disku, magnetické pásce nebo na disketě, méně často na děrné pásce nebo děrných štítcích. Tento soubor programů je třeba po částech nahrát do operační paměti počítače a spouštět. Dále je nutné zajistit možnost oboustranné komunikace s technikem, který provádí testování, tedy vyvolávání jednotlivých testů, např. z psacího stroje, a sdělování výsledků testů, např. prostřednictvím tiskárny.

Podle způsobu řízení lze programové testy rozdělit takto:

- autonomní testy;

– testy řízené diagnostickým monitorem;
– testy řízené operačním systémem počítače.

Autonomní programové testy jsou nahrávány do paměti počítače a spouštěny přímo bez obslužného programu. Jejich úkolem je provedení testu základní jednotky počítače. Nacházejí tedy uplatnění u počítačů, které nejsou vybaveny mikroprogramovou diagnostikou.

Diagnostický monitor je řídicí program, který pracuje nezávisle na operačním systému počítače. Zajišťuje všechny funkce, které normálně zastává operační systém, tj. vyhledává testovací programy ve vnější paměti, zavádí je do operační paměti, spouští je a po jejich provedení vypisuje zjištěný technický stav testované jednotky na tiskárně počítače. Z toho, že pracuje nezávisle na operačním systému, plyne také to, že i na počítačích s multiprogramováním (možnost práce více programů současně) blokuje pro sebe celý počítač. Je zde ale možnost kontroly reakce zařízení na nedovolené příkazy nebo na nedovolené sledy signálů.

Programové testy řízené operačním systémem počítače mohou pracovat paralelně s jinými programy řešenými v počítači. Takový test, který lze provést bez přerušení práce počítače, se nazývá *test za provozu.* Nevýhodou tohoto typu řízení testovacích programů je to, že nelze testovat nedovolené stavy zařízení.

Diagnostický monitor a programové testy řízené operačním systémem se používají i pro testování kanálů a přídavných zařízení počítačů.

Závěrem uvedeme metody testování operační paměti. Technologie použitá při výrobě paměti vykazuje určité typické poruchy, pro jejichž odhalení byly odvozeny určité strategie. Vzorky používané při testování pamětí lze rozdělit do tří hlavních skupin, charakterizovaných délkou testu. Je-li počet bitů paměti roven $N$, lze tyto skupiny charakterizovat takto:

– vzorek typu $N$ – počet kroků testu je lineárně závislý na počtu bitů;
– vzorek typu $N^2$ – počet kroků je úměrný druhé mocnině počtu bitů;
– vzorek typu $N^{3/2}$ – počet kroků je úměrný součinu $N \cdot \sqrt{N}$.

Testy založené na použití vzorku typu $N^2$ jsou nejdokonalejší, ale jsou málokdy použitelné pro značné časové nároky. Nejčastěji se používají vzorky typu $N$. Jejich účinnost je však omezena, a proto je v poslední době věnována pozornost vzorkům typu $N^{3/2}$.

Nejčastější vzorky typu $N$ jsou:

– šachovnice – do sousedních bitů jsou zapsány opačné hodnoty;
– řádky – do sousedních řádků jsou zapsány opačné hodnoty;
– diagonála – bity na diagonále mají jinou hodnotu než ostatní bity ap.



Po naplnění paměti příslušným vzorkem následuje čtení celé paměti nebo její části.

Vzorky typu $N^2$ jsou např. tyto:

– putující jednička – po vynulování celé paměti je na každý bit zapsána jednička a opět nula, což se kontroluje čtením celé paměti po každém zápisu;
– putující nula – totéž v inverzi.

Ze vzorků typu $N^{3/2}$ můžeme např. uvést posouvanou diagonálu: pro každou polohu diagnály (jednička v poli nul) se čte celá paměť po sloupcích.

## KONTROLNÍ OTÁZKY A ÚLOHY

1. Popište operační kód počítače ADT.
2. K čemu slouží operační systém?
3. Uveďte některé programovací jazyky.
4. Jaký je rozdíl mezi programovou diagnostikou a mikroprogramovou diagnostikou?

# 2. Periferní zařízení

## 2.1. ZÁKLADNÍ PRINCIPY PERIFERNÍCH ZAŘÍZENÍ

Rozlišujeme tyto základní směry využití výpočetních systémů:

1. hromadné zpracování dat a ASŘ (automatizované systémy řízení);
2. vědeckotechnické výpočty;
3. systémy pro automatizované konstruování (např. návrh plošných spojů);
4. řízení technologických procesů v reálném čase.

Teprve po připojení periferních zařízení k základní jednotce vznikne provozuschopný výpočetní systém. K vybudování některého uvedeného typu výpočetního systému potřebujeme vždy základní jednotku a univerzální soubor periferních zařízení. Periferní zařízení zajišťují styk základní jednotky s okolním prostředím. Dále je nutné podle zvoleného typu výpočetního systému připojit i speciální periferní zařízení. K těmto speciálním periferním zařízením většinou výrobce základní jednotky nedodává rozhraní (interface). Cílem praktické části této učebnice proto je, aby znalosti a dovednosti získané v jednotlivých cvičeních usnadnily technikovi připojit také nové periferní zařízení.

Z hlediska principu činnosti lze univerzální periferní zařízení rozdělit takto:

1. periferní zařízení pracující s papírovým paměťovým médiem (snímače a děrovače děrných štítků a děrných pásek);
2. periferní zařízení pro trvalý výstup abecedně číslicových znaků (tiskárny);
3. periferní zařízení pro ovládání výpočetního systému (abecedně číslicové displeje, psací stroje);
4. vnější velkokapacitní paměti (kazetové diskové a disketové paměti, magnetické páskové paměti).

Pro každý typ periferního zařízení musíme zvolit jinou metodu testování. Metoda závisí především na principu činnosti periferního zařízení.

U první skupiny periferních zařízení musíme nejdříve do papírového média vyděrovat všechny kombinace, které daný druh paměťového média



umožňuje zaznamenat během jedné výstupní operace. Potom lze teprve zkoušet snímače.

U druhé skupiny se musí vyzkoušet celý rozsah znaků, které dokáže periferní zařízení zobrazit. Některé tiskárny nemají malá písmena – u nich je nutné programově zajistit překódování na velká písmena.

Pro třetí skupinu platí plně to, co bylo řečeno o druhé skupině při výstupu dat ze základní jednotky. Vstup u těchto periferních zařízení je nutné testovat ručně, a to stisknutím všech kláves postupně po sobě. Stisknutá klávesa se pro kontrolu zobrazí na výstupu testovaného periferního zařízení.

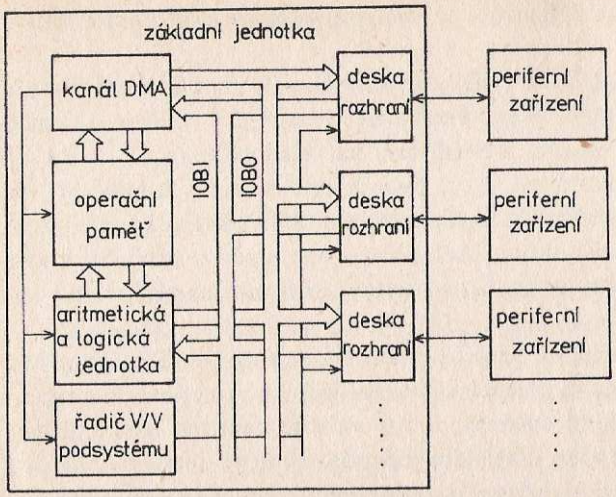
U čtvrté skupiny neplatí bezprostředně, že periferní zařízení zajišťuje styk s okolním prostředím. Paměťová média však lze vyměňovat například mezi dvěma výpočetními systémy, a tím vlastně nepřímo uskutečňovat styk s okolním prostředím. Základní jednotka využívá vnější paměti pro dočasnou nebo trvalou úschovu obsahu operační paměti, která je pro obsáhlé programy nedostatečná. Při testování se postupuje tak, že po zápisu na magnetická média se provede kontrolní čtení.

## 2.2. PŘIPOJENÍ PERIFERNÍHO ZAŘÍZENÍ K POČÍTAČI

Spojení periferního zařízení se základní jednotkou zajišťuje přenos dat prostřednictvím elektrických signálů na střední vzdálenost, tj. od několika metrů až do několika desítek metrů.

Jak při zpracování v základní jednotce, tak při přenosu dat se k zobrazení dat používá dvojková číslice – *bit* (vzniklo zkrácením anglických slov *bi*nary digi*t*). Z bitů se skládají další jednotky informace, jako jsou *slabika* a *slovo*. Osm bitů tvoří slabiku (slabika = byte). Do slabiky lze zakódovat až 256 znaků abecedy, které umožňují bohatou komunikaci mezi člověkem a počítačem. Proto se u periferních zařízení, která pracují se znaky v kódu ISO 7, používá spojení paralelní osmibitovou sběrnicí doplněnou o řídicí signály. Základní jednotka současně zpracovává slovo, které je většinou násobkem slabiky. U minipočítačů je typická délka slova 16 bitů.

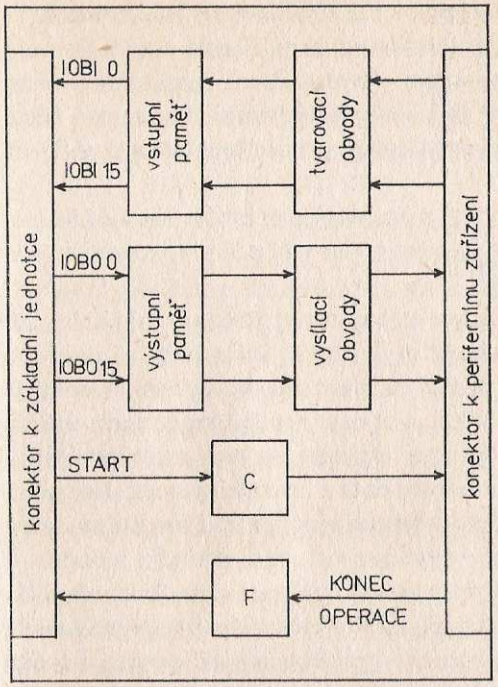
*Podsystém vstupů a výstupů* (zkráceně V/V) základní jednotky zabezpečuje přenos dat mezi deskou rozhraní a pracovními registry A a B aritmetické a logické jednotky po sběrnicích označených IOBI a IOBO (obr. 1). Data mezi deskou rozhraní a operační pamětí lze předávat také přímo pomocí kanálu DMA. Během jedné vstupní/výstupní operace se může přenést pouze jedno slovo, tj. u minipočítače typu ADT 16 bitů.




*Obr. 1.* Podsystém vstupů a výstupů základní jednotky

Připojení periferního zařízení k základní jednotce se realizuje zasunutím desky rozhraní do vany v základní jednotce; kabel pak spojuje desku rozhraní s periferním zařízením. Každý typ periferního zařízení vyžaduje jiný typ desky rozhraní a kabelu. Deska rozhraní obsahuje vstupní a výstupní vyrovnávací paměti, výstupní vysílací obvody a vstupní tvarovací obvody (obr. 2). Také obsahuje synchronizační klopné obvody F a C, které jsou popsány v následujícím článku 2.3. Vyrovnávací paměti jsou cílem nebo zdrojem přenášené informace. Obvykle jsou šestnáctibitové. Délka závisí na délce přenášených dat u daného periferního zařízení. Protože už integrované obvody pro sériový přenos jsou v současnosti dostupné, nahrazují se u abecedně číslicových displejů a tiskáren paralelní přenosy přenosy sériovými. Výhoda sériového přenosu dat, zvláště je-li spojení realizováno proudovou smyčkou, je především v tom, že kabel je pouze čtyřvodičový nebo dvojvodičový a může být dlouhý až 3 km. Výstupní vysílací obvody výkonově zesilují signály z výstupních vyrovnávacích pamětí nebo je převádějí na požadované úrovně napětí a proudu. Nejčastějšími vysílači na deskách rozhraní jsou logické členy integrovaného obvodu MH 7440. Vstupní tvarovací obvody převádějí signály vysílané z periferního zařízení na napěťové úrovně TTL nebo zakončují kabel charakteristickým odporem, a tím zamezují vzniku odrazů signálů zpět do kabelu. Délku kabelu je omezena kvalitou vysílacích a tvarovacích obvodů a také kvalitou samotného






*Obr. 2.* Připojení periferního zařízení k základní jednotce

kabelu. Elektrické signály v kabelu se mohou ovlivňovat vzájemně vlivem kapacity mezi jednotlivými vodiči kabelu (vznikají tzv. přeslechy). Do kabelu může také pronikat rušení z vnějších zdrojů (např. silná elektromagnetická pole od silových kabelů, motorů a spínačů). Proto je nutné u delších kabelů signálové vodiče stínit proti rušení. I tak nelze při zachování úrovní TTL překonat vzdálenost větší než 15 až 20 m, protože logická hodnota 0 je málo odolná proti rušení. Při logické hodnotě 0 totiž napětí nesmí překročit hranici 0,8 V. Velkým zdrojem rušení je síťové napájení jednotlivých periferních zařízení a nedokonalé společné zemnění se základní jednotkou. Vodiči kabelu, které spojují nulovou úroveň základní jednotky a periferního zařízení, mohou procházet takové vyrovnávací proudy, že vzniklé úbytky napětí jsou srovnatelné s úrovní logické hodnoty 0. Závady tohoto typu se musí odstranit dokonalým společným zemněním základní jednotky s periferním zařízením. Všechny zemnicí vodiče se musí hvězdicovitě sbíhat k základní jednotce.

Deska rozhraní se zasouvá do vany, která tvoří konstrukční celek se základní jednotkou. Deska rozhraní má z jedné strany konektory, kterými

se připojuje k obvodům v základní jednotce. Z druhé strany jsou konektory, na které se připojuje kabel od periferního zařízení. Každá pozice ve vaně má pevně přidělenou vstupní/výstupní adresu. První použitelná adresa je $10_8$. Vstupní/výstupní adresy 0 až 7 využívá základní jednotka pro účely řízení kanálu DMA, přerušovacího systému a při ošetření technologických chyb v činnosti základní jednotky.

Přenos dat mezi deskou rozhraní a operační pamětí lze uskutečnit pomocí aritmetické a logické jednotky přes registry A a B nebo kanálem přímého přístupu do paměti (DMA).

V prvním případě jde o programově řízený přenos dat, který lze použít jen při vypnutém přerušovacím systému. Informace zapsané ve vyrovnávací paměti na desce rozhraní se přenesou do registru A nebo B instrukcí LIA nebo LIB. Další instrukcí se data zapíší z registru do určené buňky v operační paměti. U výstupní operace se postupuje obráceně. Nejdříve se načtou do zvoleného registru data z operační paměti. Instrukcí OTA nebo OTB se potom zapíší do výstupní vyrovnávací paměti na desce rozhraní. Výhodou programového přenosu dat mezi základní jednotkou a periferním zařízením je snadná programová realizace a možnost okamžitého zásahu do přenosu (například při zjištění chybné parity přenášeného znaku). Nevýhodou je menší přenosová rychlost než při použití kanálu DMA. Hodí se pro řízení pomalých periferních zařízení, jako jsou abecedně číslicové displeje, tiskárny, děrovače a snímače děrné pásky.

Přímý přístup k operační paměti kanálem DMA (Direct Memory Access) použijeme tehdy, jestliže musíme zajistit velkou rychlost přenosu a značný objem přenášených informací. Přenos dat mezi operační pamětí a periferním zařízením se provádí po celých blocích a většinou při zapnutém přerušovacím systému. Pro dílčí přenos jednoho slova se využívá princip kradení cyklů operační paměti. Znamená to, že kanál DMA provádí svou činnost paralelně s činností základní jednotky. Základní jednotka je pouze v okamžiku přístupu kanálu DMA do operační paměti pozdržena ve vykonávání instrukcí. Tento přenos se používá především u připojení vnějších velkokapacitních pamětí.

## 2.3. KOMUNIKACE POČÍTAČE S PERIFERNÍM ZAŘÍZENÍM

Komunikaci počítače s periferním zařízením je třeba ovládat *řídicími postupy*, kterými se synchronizuje činnost základní jednotky a periferního zařízení.



Doby operací vykonávaných v periferních zařízeních jsou většinou mnohonásobně delší, než je doba vykonání vstupní/výstupní instrukce v základní jednotce. Proto je nutné vzájemně synchronizovat činnost v základní jednotce a v periferním zařízení při přenosu dat mezi nimi. K tomu slouží klopné obvody C a F na desce rozhraní a instrukce k jejich ovládání (STC, CLC, STF, CLF, SFS, SFC).

Klopný obvod C (Control) spouští svým nastavením periferní operaci a zároveň uvolňuje veškeré komunikační cesty mezi periferním zařízením a základní jednotkou. Po přechodu do stavu 1 se generuje příkaz START. Příkaz START spouští jeden operační cyklus v periferním zařízení (čtení nebo zápis jednoho slova nebo znaku). Zároveň slouží klopný obvod C jako maska přerušení daného zařízení. Pokud není obvod C nastaven, přerušovací signál je od tohoto zařízení blokován. Klopný obvod C se nastavuje instrukcí STC a nuluje se instrukcí CLC. Periferní zařízení nemůže klopný obvod C nijak ovlivnit.

Klopný obvod F (Flag) používá periferní zařízení k indikaci vstupní/výstupní operace. Pokud je obvod F nastaven, je přenos mezi periferním zařízením a deskou rozhraní ukončen. Instrukcemi SFS/SFC lze testovat stav klopného obvodu F. Z programu v základní jednotce lze instrukcí simulovat nastavení klopného obvodu F. Periferní zařízení může klopný obvod pouze nastavit. Pokud je při nastavení F současně i odmaskováno přerušení od této jednotky (C = 1), pak vznikají základní podmínky pro generování přerušovacího signálu. S nastavením klopného obvodu F se současně při vstupní operaci zapíší data do vstupní vyrovnávací paměti na desce rozhraní.

Při vypnutém přerušovacím systému můžeme plně řídit vstupní/výstupní operace pouze *programovými prostředky*. Při vstupu dat z periferního zařízení do základní jednotky postupujeme takto:

| | |
|---|---|
| STC 10B, C | start periferního zařízení na vstupní/výstupní adrese $10_8$ a nulování F. (Poznámka: Písmeno B za číslem znamená, že číslo je v osmičkové soustavě) |
| SFC 10B | dotaz na F = 1 |
| JMP *−1 | pokud F = 0, vrať se o instrukci zpět |
| LIA 10B | čtení dat ze vstupní vyrovnávací paměti na desce rozhraní do registru A |

Při výstupu dat na periferní zařízení postupujeme takto:

| | |
|---|---|
| OTA 11B | naplnění vyrovnávací paměti na desce rozhraní z registru A |

```
STC  11B, C   start periferního zařízení na vstupní/výstupní adrese 11₈
              a nulování F
SFS  11B      dotaz na F=1
JMP  *−1      pokud F=0, vrať se o instrukci zpět
```

K úplnému vysvětlení komunikace mezi periferním zařízením a základní jednotkou je třeba mít alespoň základní znalosti o *přerušovacím systému.* Přerušovací systém slouží k efektivnějšímu využití základní jednotky. Dobu, po kterou vykonává periferní zařízení operaci, může základní jednotka věnovat řešení jiného programu. Ukončení činnosti periferního zařízení způsobí přerušení vykonávaného programu v základní jednotce. Po ukončení obsluhy přerušení bude základní jednotka pokračovat v původním (přerušeném) programu.

Přerušovací systém minipočítačů typu ADT umožňuje rozeznat 60 úrovní přerušení. Každé *úrovni přerušení* odpovídá jedna buňka v operační paměti základní jednotky, kde je umístěna první instrukce *obslužného programu přerušení.* Adresa této buňky je shodná se vstupní/výstupní adresou desek rozhraní. Každou úroveň přerušení lze samostatně maskovat. *Maskováním* se rozumí potlačení přerušení základní jednotky. Programově lze také maskovat celý přerušovací systém, kromě úrovní přidělených k ošetření technologických chyb a kromě ošetření výpadku napájení.

Přidělení funkcí k přerušovacím úrovním:

04 přerušení výpadkem napájení
05 paritní chyba/ochrana paměti
06 přerušení od konce přenosu DMA 1
07 přerušení od konce přenosu DMA 2
10 první deska rozhraní s nejvyšší prioritou
11 druhá deska rozhraní
. .
. .

Jestliže byl nastaven klopný obvod C instrukcí STC a periferní zařízení hlásí konec své operace signálem FLAG, dojde k přerušení činnosti základní jednotky, pokud nenastane některá z následujících událostí:

1. Je zamaskován přerušovací systém.
2. Není dokončena instrukce JMP nebo JSB při nepřímém adresování (a navíc ještě jedna fáze následující instrukce).



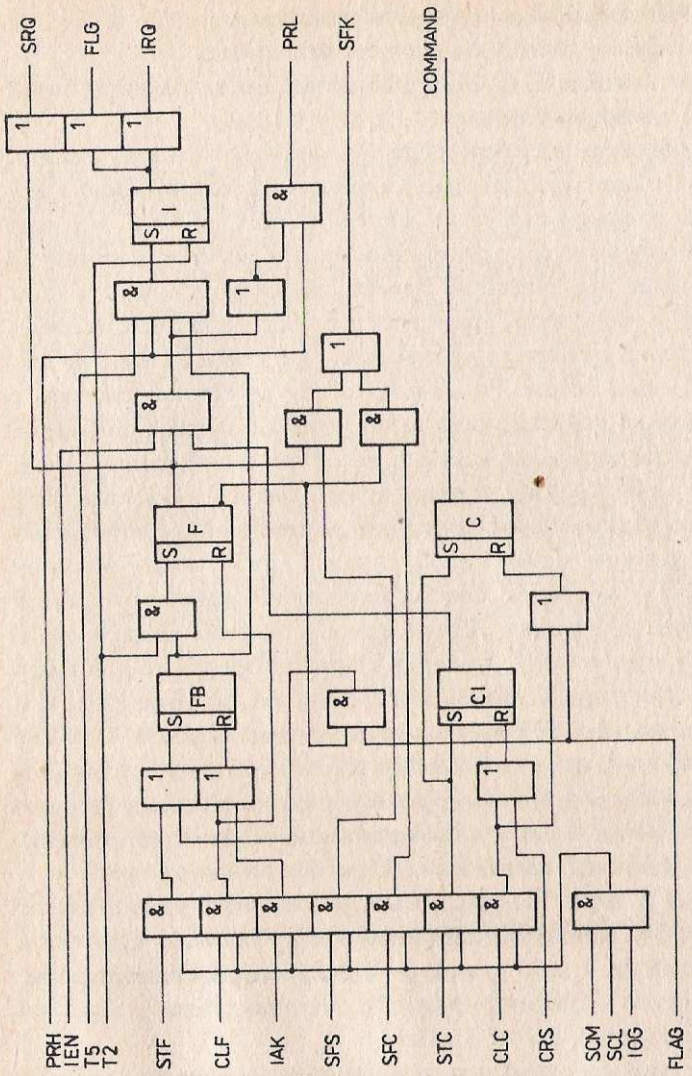
3. Není dokončena instrukce v přerušovací buňce.
4. Kanál DMA právě přenáší do operační paměti data.
5. Provedla se instrukce STC, CLC, STF nebo CLF a neskončila dosud další jedna fáze následující instrukce.

Priorita vyhodnocení přerušení klesá se vzrůstající vstupní/výstupní adresou. Všemi deskami rozhraní prochází maskovací prioritní vodič, který se při dokončení operace v periferním zařízení, tj. při C = 1 a F = 1, rozpojí. Přítomnost maskovacího prioritního signálu na desce rozhraní je nutnou podmínkou pro přerušení činnosti základní jednotky. Jestliže dojde současně k dokončení operací u několika periferních zařízení, vyhodnotí se pouze u periferního zařízení, které má desku rozhraní na nejnižší vstupní/výstupní adrese. Po obsluze tohoto periferního zařízení se opět spojí maskovací prioritní vodič, a tím pronikne signál k další desce rozhraní, která čeká na obsluhu základní jednotkou nebo kanálem DMA.

Na obr. 3 je úplné logické zapojení komunikačních a přerušovacích obvodů na desce rozhraní. Jakmile periferní zařízení dokončí dříve zahájenou činnost, generuje signál FLAG. Signál FLAG nastaví pomocný klopný obvod FB (Flag Buffer), jenž je programově nepřístupný a slouží jako paměť žádosti o přerušení. Klopný obvod F se může nastavit pouze v taktu T2, je-li nastaven FB. Výstup z klopného obvodu je hradlován signálem IEN (Interrupt ENable), který centrálně povoluje přerušení a je nastavován instrukcí STF 00 a nulován instrukcí CLF 00. Tyto dvě instrukce zapínají a vypínají přerušovací systém. Dále je výstup z F hradlován výstupem z klopného obvodu CI, který maskuje přerušení na desce rozhraní. Klopný obvod I slouží k indikaci přerušení v základní jednotce. Klopný obvod I je v každé fázi nulován taktem T2 a nastavován při F = 1, C = 1, PRH = 1 a taktu T5. Signál PRH je maskovací prioritní vodič od předchozí desky rozhraní s vyšší přerušovací prioritou a signál PRL vede k desce rozhraní s nižší prioritou. Klopný obvod C slouží pouze ke generování signálu COMMAND (start periferního zařízení). Signálem FLAG se obvykle klopný obvod C nuluje.

Na příkladu ukážeme strukturu typického programového způsobu ošetření přerušení na vstupních/výstupních adresách $10_8$ a $11_8$.

```
ORG  10B
JSB  P10      přerušovací buňka vstupní/výstupní adresy 10₈
JSB  P11      přerušovací buňka vstupní/výstupní adresy 11₈
HLT  12B
```




*Obr. 3.* Zapojení komunikačních a přerušovacích obvodů na desce rozhraní



```
P10 NOP         sem se uloží adresa přerušeného programu
     .
     .          tělo podprogramu P10
    CLC 10B     prioritní odmaskování nižších úrovní
    JMP P10, I  návrat do přerušeného programu
P11 NOP         podprogram obsluhy přerušení od vstupní/výstupní
                adresy 11₈
     .
     .
    CLC 11B
    JMP P11, I
```

Jestliže dojde k přerušení na vstupní/výstupní adrese $10_8$, vykoná se instrukce JSB P10 z adresy $10_8$ v operační paměti místo instrukce zrovna vykonávaného programu. Do vstupního bodu podprogramu se uloží adresa vykonávaného programu. Po ukončení obsluhy přerušení se odmaskují vyšší vstupní/výstupní adresy instrukcí CLC 10B. Instrukcí JMP P10,I se řízení základní jednotky vráti do přerušeného programu. Pro přerušení na vstupní/výstupní adrese $11_8$ bude obsluha probíhat stejně jako pro vstupní/výstupní adresu $10_8$. Dojde-li k přerušení na vstupní/výstupní adrese $12_8$, základní jednotka se zastaví na instrukci HLT 12B. Po zastavení bude v čítači instrukcí adresa přerušeného programu.

Všechny testovací programy uvedené ve cvičeních s periferními zařízeními využívají pouze programového ovládání bez využití přerušovacího systému. Testovací programy lze přepsat tak, aby využívaly přerušovacího systému podle následujícího vzoru:

```
        ORG  10B
        JSB  VYZN      přerušovací buňka vstupní/výstupní adresy
                       10₈
        ORG  100B
START   STF  00        zapnutí přerušovacího systému
        CLA
        STA  CT        vynulování CT
        JSB  INIC      zahájení přenosu
OP1     LDA  CT        simulovaný hlavní program
        INA            s čítáním obsahu CT
        STA  CT
```

```
        OTA  1          a zobrazením v registru S
        JMP  OP1        opakuj

INIC    NOP             zahájení přenosu dat na periferní zařízení
        CLA
        STA  ZN         vynulování ZN
        OTA  10B        zápis do desky rozhraní
        STC  10B, C     start periferního zařízení a nulování F
        JMP  INIC, I    návrat z procedury

VYZN    NOP             obsluha přerušení
        LDA  ZN         převezmi vysílaná data
        INA             inkrementuj
        STA  ZN         a ulož zpět do ZN
        CPA  B400       je konec přenosu?
        JMP  KONEC      ano, ukonči program
        OTA  10B        ne, zapiš další znak do desky rozhraní
        STC  10B, C     start periferního zařízení a nulování F
        JMP  VYZN, I    návrat do přerušeného programu
KONEC   CLF  00         vypnutí přerušovacího systému
        HLT  01B        konec programu

CT      BSS  1          čítač
ZN      BSS  1          znak pro výstup
B400    OCT  400        konečná hodnota znaku
```

K úpravě byl zvolen testovací program pro děrovač z čl. 6.1. Upravený testovací program má tyto tři části:

1. hlavní program;
2. zahájení přenosu dat na periferní zařízení;
3. obsluha přerušení.

Hlavní program vyvolá zahájení výstupní operace na periferním zařízení podprogramem INIC. Aby bylo dále zřejmé, že se vykonává hlavní program, neustále se vysílá do registru S obsah čítače CT.

Podprogram zahájení přenosu INIC připraví počáteční hodnotu v buňce ZN a odstartuje výstupní operaci prvního znaku. Místo čekání na dokončení výstupní operace se vrátí do hlavního programu.

Podprogram obsluhy přerušení VYZN se aktivuje při každém přerušení od periferního zařízení. Přitom se vykonávání hlavního programu přeruší. Podprogram připraví k výstupu další znak a odstartuje výstupní operaci



v periferním zařízení. Po ukončení obsluhy přerušení předá podprogram řízení do přerušeného místa v hlavním programu. Jestliže je vyslán poslední povolený znak, podprogram se nevrátí do hlavního programu, ale dojde k zastavení základní jednotky.

## KONTROLNÍ OTÁZKY A ÚLOHY

1. Jakými způsoby se testují periferní zařízení?
2. Vysvětlete rozdíl mezi programovým ovládáním periferních zařízení a přenosem pomocí kanálu DMA?
3. K čemu slouží klopné obvody C a F na desce rozhraní?
4. Vyjmenujte výhody při použití přerušovacího systému v počítači.

# 3. Mikropočítač

## 3.1. TECHNICKÝ POPIS MIKROPOČÍTAČE

### 3.1.1. Úvod

V poslední době se mikroelektronika rychle rozvíjí, což se promítá i do dalších oborů, jako je strojírenství, řídicí a regulační technika i samotný elektrotechnický průmysl. Kvalitativně nová úroveň výrobků je často podmiňována právě použitím obvodů velké a velmi velké integrace, především mikroprocesorů. Pro další léta je reálný předpoklad, že udané tempo rozvoje mikroprocesorů bude ještě zvyšováno, proto je třeba vychovat mnoho specialistů, jak pro oblast technického vybavení (hardware) těchto systémů, tak i programového vybavení (software). Ve výkladu věnovaném mikropočítačům bychom se měli zaměřit na řešení základních úloh pomocí mikropočítače na bázi mikroprocesoru MHB 8080A, ke kterým budeme přistupovat z obou zmíněných hledisek, technického i programového.

Jako vzor mikropočítače pro ukázky v kapitolách 3 a 7 byl vybrán jednodeskový mikropočítač typu TEMS 80-03. Patří k nejjednodušším mikropočítačům s relativně chudým programovým vybavením. Z hlediska technického vybavení má ale největší výhodu ve snadné přístupnosti všech základních i periferních obvodů, je stavebnicově řešený a lze jej rozšířit o další obvody. Další možné aplikace přináší deska přídavných periferních zařízení TEMS 80-04 a především deska TEMS 80-06. Pomocí těchto desek je možné řešit téměř všechny úlohy z kapitoly 7.

Obdobou mikropočítače TEMS jsou systémy PMI 80 a JPR-1, které je možné použít po doplnění několika drobnými přípravky pro všechny úlohy. Obecně lze použít (s malými úpravami) většinu systémů vybavených klávesnicí, zobrazovačem a systémovou sběrnicí přístupnou z vnějšku.

V této kapitole uvedeme některé základní teoretické poznatky o konstrukci i programování mikropočítačových systémů, a to jak obecnějšího charakteru, tak přímo pro vybraný typ mikropočítače TEMS 80-03.

V kapitole 7 uvedeme dvanáct tematických bloků různých úloh a měření



pro konkrétní aplikace, které se v praxi často používají. Některé úlohy jsou voleny tak, aby ukazovaly bohaté možnosti periferních obvodů i za cenu, že řešení nebude praktické. Každá úloha obsahuje teoretickou část, slovní popis zapojení, výpisy potřebných programů i doplňující otázky pro samostatnou práci studentů. Zůstává na volbě vyučujících, budou-li se programy obsažené v úlohách (určené pro zavedení do paměti RAM) zadávat do mikropočítačů přímo z klávesnice, nahrávat z kazetového magnetofonu nebo kopírovat z přídavných pamětí EPROM.

### 3.1.2. Blok obvodů mikroprocesoru

Základem mikropočítače je mikroprocesor 8080 se svými podpůrnými obvody 8224 a 8228. Hlavní částí generátoru hodinových impulsů 8224 je oscilátor řízený vnějším krystalem (nejčastěji se užívá max. frekvence 18,432 MHz), který generuje hodinové impulsy Φ1, Φ2, dále Φ2 (TTL) a signál OSC, potřebné pro obvody sériového kanálu i případné časování vnějších systémů. Generátor dále synchronizuje signály RESET (od tlačítka RESET) a READY pro mikroprocesor, generuje vzorkovací signál STSTB pro obvod 8228. Obvod 8228 je budič obousměrné datové sběrnice a systémový řadič, který vydává řídicí signály pro jednotlivé bloky (/MEMR,/MEMW,/IOR,/IOW,/INTA).

Do obvodů bloku mikroprocesoru můžeme u TEMS 80-03 zahrnout i obvody pro provoz mikroprocesoru v krokovém režimu a obvody generování vnějšího přerušení stisknutím tlačítka INT. (Zde se využívá generování instrukce RST 7 – 0FFH – pomocí připojení výstupu /INTA obvodu 8228 přes rezistor s odporem 1 kΩ na +12 V.)

### 3.1.3. Blok obvodů paměti

U mikropočítačových systémů se vyskytují paměti různého typu, kapacity a konfigurace. Obecně lze říci, že paměť má tyto tři části:

1. *Pevnou paměť* (pouze pro čtení). Je sestavena převážně z obvodů 2708 (popř. 8708, 2758) o kapacitě 1 KB, případně 2716 (popř. 8716) o kapacitě 2 KB atd. Používají se paměti PROM (bez možnosti vymazání) nebo EPROM (s možností smazat a znovu naprogramovat obsah). V těchto pamětech bývá uložen základní systémový program (monitor), u rozsáhlej-

ších systémů ještě zaváděcí program, který se však hned po iniciaci systému překrývá pamětí RAM. U systému TEMS 80-03 je v základní konfiguraci osazena 1 paměť EPROM 2708 (1 KB), popř. 2716 (2 KB). Je však možnost rozšířit kapacitu paměti až na 4 KB, popř. 8 KB.

2. *Operační paměť* (RAM – pro čtení i zápis). Její kapacita se může u jednotlivých systémů značně lišit (bývá od velikosti 0,5 KB až do 64 KB). Rozlišují se dva typy, a to

a) Statická paměť – informace je po zápisu trvale uložena až do výpadku napájení, jsou ale paměti, které uchovávají informaci i při výpadku (feritová jadérka), nebo stačí napájení z baterie (paměť CMOS aj.). Nejčastěji se používají obvody 2102 (1 K × 1 bit), 2114 (1 K × 4 bity) aj.

b) Dynamická paměť – informaci zapsanou v paměti je nutné periodicky obnovovat, neboť má pouze krátkodobý charakter. Z tohoto důvodu přibývají k systému regenerační (refresh) obvody (někdy bývají součástí samotného mikroprocesoru), nepatrně se zpomaluje výpočet (při regeneraci je třeba do instrukčního cyklu vkládat další doby $T_w$), ale paměti se vyrábějí s mnohem většími kapacitami než paměti statické. U nás jsou nejpoužívanější paměti MHB 4116 (16 K × 1 bit) nebo sovětské paměti K565RU5 (65K × 1 bit).

U TEMS 80-03 má paměť RAM kapacitu 1 KB a je složena z osmi obvodů MHB 2102.

3. *Vnější paměť.* Bývá realizována mnoha způsoby: jako magnetická pásková paměť, kazetová pásková paměť, magnetická disková paměť, paměť s pružnými disky (floppy disc) atd. Vyznačuje se větší kapacitou, ale delší vybavovací dobou než operační paměť. U TEMS 80-03 tuto funkci může zastávat běžný kazetový magnetofon, který se připojuje přes dvě linky obvodu 8255.

### 3.1.4. Stykové obvody pro připojení periferních zařízení

I v této oblasti poskytují výrobci integrovaných obvodů velké možnosti výběru. Podrobněji budou tyto obvody, včetně přerušovacího systému, popsány v čl. 3.3.

V této části se zmíníme pouze o způsobu připojení displeje a klávesnice u systému TEMS 80-03. Tyto obvody zajišťují samostatnou činnost displeje, dekódování stisknuté klávesy a předání jejího kódu na sběrnici.

Klávesnice mikropočítače TEMS je tvořena 16 hexadecimálními a 6



funkčními tlačítky, dále jsou zde dvě systémová tlačítka (RESET – pro iniciaci systému a INT pro vyvolání vnějšího přerušení). Je organizována jako vstupní zařízení s paralelním přenosem dat.

Mikropočítač TEMS obsahuje dále šestimístný sedmisegmentový displej rozdělený do dvou částí – čtyři levé segmentové zobrazovací prvky tvoří adresové pole, dva zbývající tvoří datové pole. Displej je připojen jako paměťově mapované periferní zařízení (je adresován paměťovými instrukcemi s adresou 3C00 až 3C04H směrem zprava) a jeho pomocné obvody zajišťují obnovování zobrazované informace. Podrobněji bude připojení klávesnice a displeje ukázáno v článcích 7.3. a 7.4.

### 3.1.5. Deska přídavných zařízení

Deska přídavných zařízení obsahuje tři funkční bloky, které slouží pro demonstraci řídicích a regulačních možností, jsou to:

a) blok ovládání elektromotorku – obsahuje převodník čísla na šířkově modulovaný impuls a umožňuje i zpětný chod motorku, dále obsahuje snímač otáček motorku;

b) blok snímání teploty – obsahuje termistor (nebo jiný snímač teploty), který převádí teplotu na šířku impulsu; mikropočítač spustí monostabilní klopný obvod a testuje dobu jeho návratu do klidového stavu;

c) blok akustického výstupu – obsahuje jednotranzistorový zesilovač, který připojuje reproduktor k výstupu CØ obvodu 8225; změnou periody se ovlivňuje i výška generovaného tónu.

Kromě těchto hlavních bloků obsahuje deska i rozsáhlé univerzální pole pro zapojení dalších obvodů; v tomto poli jsou již rozvedeny systémové signály včetně všech napájecích napětí.

### 3.1.6. Obvody desky TEMS 80-06

K systému TEMS může být dodána ještě další deska s přídavnými obvody, která je nejčastěji využívána při řešení úloh z kapitoly 7. Skládá se z několika souvislých bloků, které je možné propojovat pomocí propojovacích polí, konektorů a připojovat k nim potřebné jednoduché logické obvody, které jsou rovněž na desce. Hlavní bloky desky TEMS 80-06 jsou tyto:

1. matice diod LED 5 × 7 bodů s obvody pro její řízení – znakový generátor v paměti ROM (2501) adresovaný bránou 3212, z níž je matice buzena z dekódovaného čítače;

2. dva přídavné sedmisegmentové zobrazovací prvky, které mají vstupy na jednotlivé segmenty vyvedeny na propojovací pole;

3. dva vstupní, popř. výstupní registry 3212, připojené na datovou sběrnici a adresované instrukcemi IN, popř. OUT 0DEH;

4. obvody 3214 a 3212 pro generování signálů přerušení RST 6 a RST 7;

5. šestnáctibitový dělič frekvence Φ2 s nastavitelnou počáteční hodnotou;

6. převodník A/D složený z obvodů MDAC 08C (převodník D/A) a MHC 1502 (8bitový aproximační registr);

7. další univerzální paralelní rozhraní 8255 s adresou 7E, 7FH;

8. převodník D/A z obvodu MDAC 08C, včetně operačního zesilovače a komparátoru z obvodů MAA 741.

Na desce je však mnoho dalších malých obvodů, spínačů DIL, diod LED pro mnoho různých praktických aplikací.

## 3.2. PROGRAMOVÉ VYBAVENÍ MIKROPOČÍTAČOVÝCH SYSTÉMŮ

### 3.2.1. Obecné rozdělení programového vybavení

Stejně jako v technickém vybavení různých mikropočítačových systémů jsou i ve vybavení programovém velké rozdíly. Obecně lze programové vybavení mikropočítačů (software) rozdělit do těchto skupin:

1. *Základní programové vybavení – monitor;* je obsažen ve většině obecných mikropočítačových systémů. Je to soubor programů, které obsluhují základní technické jádro mikropočítače a periferní zařízení (klávesnici, displej, vnější paměťová zařízení – snímač/děrovač děrné pásky, kazetovou páskovou paměť nebo kazetový magnetofon, popř. i paměť s pružnými disky a další periferní zařízení – programátor pamětí, tiskárnu, dálnopis atd.).Kromě toho monitory obsahují i programy, které jsou u systémů nejčastěji využívány (vstupní/výstupní podprogramy celých bloků, znaků a některé další systémové podprogramy) a konečně soubor příkazů pro ladění uživatelských programů na elementární úrovni (o prostředcích pro ladění programů se podrobněji zmíníme v čl. 3.4.). Rozsah monitoru bývá 1 až 4 KB a bývá uložen v pamětech typu (E)PROM.



2. *Operační systém* – u většiny vývojových systémů bývá nadstavbou monitoru. Zahrnuje rozšířené obslužné programy pro periferní zařízení, především pro jednotky s vnějšími paměťovými médii (pružné disky); programy jsou zpracovány ze systémového hlediska tak, aby programátorovi poskytovaly vyšší úroveň styku s těmito periferními zařízeními (práci se soubory apod.). Operační systém často obsahuje i základní operace se soubory na magnetickém disku (výpis obsahu disku, mazání souborů atd.). Operační systém se zavádí do operační paměti většinou z paměťových médií po iniciaci systému. V operačních systémech bývá tzv. *jádro*, jehož účelem je připojit operační systém na konkrétní mikropočítač. Ostatní části operačního systému, včetně uživatelských programů, se pak stávají nezávislými na použitém druhu vývojového systému, a tím i kompatibilními (slučitelnými) z hlediska programátora. U mikropočítačů patří mezi nejrozšířenější systémy ISIS-II (slučitelná československá verze se označuje DOS), CP/M (československý ekvivalent MIKROS), z novějších systémů lze např. uvést MS-DOS a UNIX. Kromě toho existují i operační systémy určené pro výpočetní systémy pracující v reálném čase, z nichž nejznámější je RMX (náš ekvivalent ERČ).

3. *Rozšiřující programové vybavení systému* – vyvolává se z vnějšího paměťového média. Do této skupiny můžeme zařadit další programy pro přídavná zařízení, pro práci se soubory, redakční programy (editory), překladače programovacích jazyků, sestavovací (LINK) a zaváděcí (LOCATE) programy, různé knihovní systémy atd. Překladače můžeme rozdělit do dvou hlavních skupin:

a) Kompilátory, které nejprve vygenerují přeložený strojový kód, jenž se po případném sestavení a umístění v paměti spouští. Nejpoužívanější z nich jsou tyto: jednodušší – ASEMBLER, PL/M (pouze pro systém ISIS-II), popř. PL/I (pro systém CP/M), složitější – nový jazyk C, FORTRAN, COBOL, PASCAL a nový jazyk ADA. Největší výhodou kompilátorů je rychlost vykonávání přeloženého programu (přibližně 100krát až 1 000krát vyšší než u interpretačních překladačů).

b) Interpretační překladače, které každý příkaz nejprve překládají do strojového kódu a pak hned vykonávají. Jejich největší výhodou je jednoduchost ladění programů, velká časová úspora vzniká tím, že zcela odpadá fáze vlastního překladu. Proto se překladač jazyka BASIC rozšířil téměř do všech druhů osobních mikropočítačů a stal se, ne zrovna nejšťastněji, nejznámějším programovacím jazykem.

4. *Uživatelské programy* – do této skupiny lze zařadit jednak programy,

kterými uživatel doplňuje programové vybavení systému, a přizpůsobuje tak jeho vlastnosti ke konkrétnímu využití, a jednak zcela speciální programové vybavení pro konkrétní aplikace.

### 3.2.2. Programové vybavení mikropočítače TEMS 80-03

Vzhledem ke své jednoduchosti, technickému vybavení a předpokládanému způsobu využití obsahuje základní programové vybavení mikropočítače TEMS 80-03 pouze *monitor*, který má velikost 1 KB a je uložen na adresách 0 až 3FFH. Monitor zajišťuje tyto funkce:

– *Iniciaci systému.* Po stisknutí tlačítka RESET monitor uvede všechny obvody do počátečního stavu, nastaví čítač instrukcí (PC) na 0, ukazatel zásobníku na adresu 23C9H, na displeji se objeví zpráva 80-03, přerušení jsou zakázána.

– *Zpracování vnějšího přerušení.* Po stisknutí tlačítka INT se vyvolá požadavek na přerušení RST 7. Není-li procesor ve stavu zákaz přerušení (DI), procesor předá řízení na adresu 38H (s uložením návratové adresy), kde je instrukce skoku do paměti RAM na adresu 23CCH, kam si uživatel může zapsat instrukci skoku do vlastního podprogramu.

– *Zobrazení a změnu obsahu registrů.* Při stisknutí klávesy REG, klávesy se jménem registru a klávesy NEXT se zobrazují postupně obsahy registrů v pořadí A, B, C, D, E, F, H, L, SP (horní), SP (dolní), PC (horní) a PC (dolní byte), počínajíce zadaným registrem (jeho jméno se zobrazí v adresovém poli, obsah v datovém). Uživatel má možnost obsah registru změnit zadáním hodnoty z klávesnice, stiskem klávesy NEXT se přejde na další registr, klávesa EXEC příkaz ukončuje.

– *Prohlížení a změnu obsahu paměti.* Po stisknutí klávesy MEM, zadání adresy a stisknutí klávesy NEXT se zobrazí obsah daného paměťového místa, které je možné měnit obdobně jako při příkazu REG. Snaha o změnu obsahu paměti EPROM vyvolá chybové hlášení (Err) a návrat do monitoru.

– *Spuštění programu.* Stiskem klávesy GO (popř. se zadáním adresy) a klávesy EXEC odstartujeme program od okamžitého stavu čítače instrukcí PC, popř. od zadané adresy – zobrazuje se v adresovém poli.

– *Krokování programu.* Stiskem klávesy STEP (popř. se zadáním adresy) a klávesy NEXT odstartujeme krokování programu od stavu čítače instrukcí PC (od zadané adresy). Každým stiskem klávesy NEXT se provede jedna instrukce programu, v adresovém poli se zobrazuje její



*Tabulka 1.* Vnitřní kódy kláves

| Klávesa | Kód | Klávesa | Kód | Klávesa | Kód | Klávesa | Kód |
|---|---|---|---|---|---|---|---|
| 0 | 00H | 6 | 06H | C | 0CH | GO | 12H |
| 1 | 01H | 7 | 07H | D | 0DH | MEM | 13H |
| 2 | 02H | 8 | 08H | E | 0EH | REG | 14H |
| 3 | 03H | 9 | 09H | F | 0FH | STEP | 15H |
| 4 | 04H | A | 0AH | EXEC | 10H | | |
| 5 | 05H | B | 0BH | NEXT | 11H | | |

*Tabulka 2.* Vnitřní tvary znaků

```
DSPTB:  DB 0F3H  ; znak 0 ,      index 0
        DB 60H   ;      1              1
        DB 0B5H  ;      2              2
        DB 0F4H  ;      3              3
        DB 66H   ;      4              4
        DB 0D6H  ;      5 nebo S       5
        DB 0D7H  ;      6              6
        DB 70H   ;      7              7
        DB 0F7H  ;      8              8
        DB 76H   ;      9              9
        DB 77H   ;      A              A
        DB 0C7H  ;      b              B
        DB 93H   ;      C              C
        DB 0E5H  ;      d              D
        DB 97H   ;      E              E
        DB 17H   ;      F              F
        DB 67H   ;      H              10
        DB 83H   ;      L              11
        DB 37H   ;      P              12
        DB 05H   ;      r              13
        DB 00H   ;    mezera           14
```

adresa, v datovém poli kód následující instrukce. Pro krokovací režim se využívá přerušení RST 7, proto není možné krokovat programy, které toto přerušení využívají. Klávesou EXEC se způsobí návrat do monitoru.

– *Hexadecimální sčítání a odčítání.* Stiskem klávesy HEX a zadáním dvou maximálně čtyřciferných hexadecimálních hodnot ukončených klávesou NEXT se na displeji zobrazí nejprve jejich součet, po dalším stisku klávesy NEXT jejich rozdíl. Čísla jsou brána modulo $2^{16}$, záporná jsou zobrazována v dvojkovém doplňku. Klávesa EXEC program ukončuje.

*Tabulka 3.* Systémové programy dostupné uživateli

| Název<br>Adresa (volání) | Vstupy/výstupy | Činnost | Mění obsahy |
|---|---|---|---|
| AKTDS<br><br><br>213H | registr C – volba pole[1]) – 02 – datové pole<br>04 – adresové pole<br>registr D, popř. dvojice DE – 2 nebo 4 číslice k zobrazení | aktualizace adresového nebo datového pole displeje z DE (D) | |
| CLDIS<br>20AH | | vymaže displej, skočí do monitoru, zobrazí příznak „–“, očekává monitorovský příkaz | všech registrů |
| CLDAT<br>019H | | vymaže datové pole displeje | všech registrů |
| CLEAR<br>028H<br>(RST 5) | C = 84H – zobrazí nápovědnou tečku na začátku adresového pole<br>C = 04H – nezobrazí nápovědnou tečku | vymaže adresové i datové pole displeje | všech registrů |
| DELAY<br>218H | dvojice DE – číslo $N$ = délka čekací smyčky | volitelný čekací interval $T = (27 + 34N) \cdot 0{,}48$ μs | všech registrů |
| ERR<br>232H | | ohlášení chyby, předání řízení do monitoru, zobrazí se „–“ | všech registrů |



| | | |
|---|---|---|
| EXCHG<br>2E3H | HL, DE – adresy začátku bloků paměti<br>C – jejich délka<br>Výstup: HL, DE – adresy konce bloků paměti + 1 | výměna obsahu dvou bloků paměti |
| GTHEX<br>248H | B=82H – 2 byty z datového pole<br>B=84H – 4 byty z adresového pole<br>Výstup: DE – zadané číslo nebo 0 – pro nesprávné ukončení<br>A – ukončující znak<br>CY=1 po zadání alespoň 1 číslice + správné ukončení | očekává zadání řetězce hexadecimálních číslic, vždy se zobrazí nápovědná tečka v daném poli, ukončení nehexadecimálním znakem (správné EXEC, NEXT, ostatní ruší funkci!) |
| GTNUM<br>2EFH | obdoba GTHEX | vstup hexadecimálního čísla z adresového pole |
| OUTPT<br>020H<br>(RST 4) | HL – adresa znaků<br>registr C – volba pole[1]) – 02 – datové<br>– 04 – adresové | zobrazí znaky na displeji. Všechny znaky jsou zadávány ve formě indexu (viz tab. 2) |
| RDKBD<br>010H<br>(RST 2) | | čtení znaků z klávesnice, čeká zadání znaku, odevzdá je v kódu (viz tab. 1); podprogram obsahuje programové zabezpečení proti ztrátě nebo vícenásobnému čtení téhož znaku |

*Poznámka:* [1]) U volby pole je možné přičtením hodnoty 80H k parametru (82H, nebo 84H) způsobit zobrazení nápovědné tečky na začátku určeného pole.

– *Příkazy pro vstup/výstup dat na kazetový magnetofon* (příkaz SAVE případně s adresou). Po stisku klávesy EXEC se zapíše blok dat počínaje stavem čítače instrukcí PC (popř. počínaje zadanou adresou) až po konec stránky (adresa OR 0FFH), tedy max. 256 bytů na kazetový magnetofon (spuštěný!). Po skončení zápisu se řízení předá do monitoru.

– *Příkaz GET* (popř. se zadanou adresou). Po stisku klávesy EXEC se ze spuštěného (!) kazetového magnetofonu nahraje blok dat (zapsaných příkazem SAVE) do paměti počínaje stavem čítače instrukcí PC (popř. počínaje zadanou adresou). Po nahrání se řízení vrací do monitoru.

Využití přerušovacích vektorů

RST 0 – RESET – studený start – spuštění systému
RST 1 – EXIT – předá řízení do monitoru bez změny obsahu registrů
RST 2 – RDKBD – čtení z klávesnice
RST 3 – nedefinováno
RST 4 – OUTPT – zápis na segmentový displej
RST 5 – CLEAR – vymazání displeje
RST 6 – uživatelský – JMP 23C9H sem se umístí odskoky
RST 7 – uživatelský – JMP 23CCH do uživatelských podprogramů

## 3.3. PŘIPOJENÍ PERIFERNÍCH ZAŘÍZENÍ, PŘERUŠOVACÍ SYSTÉM

### 3.3.1. Úvod

Samotný mikropočítač nemá pro praktické využití žádný význam, je třeba k němu připojit *periferní zařízení*, která umožní jeho styk s okolím. Do pojmu *okolí* můžeme zahrnout člověka (k periferním zařízením mikropočítače pro styk s člověkem řadíme např. různé klávesnice, zobrazovače, tiskárny, přepínače, tlačítka aj.) nebo vlastní fyzikální okolí, tj. řízený proces (k periferním zařízením patří převodníky A/D, D/A, snímače, zesilovače atd.). Mezi periferní zařízení řadíme i jednotky vnějších pamětí, které slouží k uchování informace. Všechna tato zařízení jsou k mikropočítači připojena přes *stykové obvody (obvody rozhraní* neboli *interface)* ke sběrnici mikropočítače. Mikropočítač a periferní zařízení tvoří tedy dva různé systémy, jejichž činnost je třeba vzájemně *synchronizovat*. Řídicí úlohu zde



má mikropočítač, který podle svého programu přikazuje periferním zařízením, aby zahájily činnost. Pokud určitou činnost vyvolá periferní zařízení, mohou nastat tyto případy:

1. Technické vybavení daného periferního zařízení je na takové úrovni, že určenou operaci uskutečňuje bez vědomí mikropočítače. Do této skupiny lze zařadit velmi rychlá periferní zařízení (s odezvou 1 μs a kratší), ale i vyšší periferní zařízení s *vlastním procesorem*, kde se komunikace mikropočítače a periferního zařízení mění na součinnost dvou procesorů.

2. Periferní zařízení používá pro styk s mikropočítačem *stavový bit* (popř. *byte*), který mikropočítači sdělí okamžitý stav periferní operace (průběh, ukončení, poruchu atd.). Mikropočítač pak musí testovat stav těchto příznaků a podle nich řídit další činnost (připraví/vybere data, vyšle řídicí slovo atd.). Toto řešení se využívá pro běžná (pomalá) periferní zařízení.

3. Součinnost periferních zařízení a mikropočítače je synchronizována pomocí *přerušovacího systému*. Periferní zařízení, které např. žádá obsluhu nebo hlásí ukončení činnosti, předá mikropočítači signál, který vyvolá přerušení běhu programu a způsobí (podle svého původu) odskok na příslušný obslužný podprogram; po jeho skončení se mikropočítač vrátí ke své původní činnosti. Tento postup se používá pro rychlejší periferní zařízení nebo v případě, kdy mikropočítač obsluhuje více těchto periferních zařízení současně.

4. Některá periferní zařízení, která přenášejí větší souvislé bloky dat z/do paměti mikropočítače, mohou tato data vybírat/ukládat bez spolupráce mikropočítače *(přímý přístup do paměti – DMA)*. Je to běžné především u periferních zařízení přenášejících bloky dat, u nichž je žádána vyšší rychlost.

### 3.3.2. Obvody pro přerušovací systém

Důležitým prvkem pro synchronizaci činnosti mikropočítače a periferních zařízení jsou přerušovací signály. Pro zpracování těchto signálů jsou u mikropočítače určeny specializované obvody, nazývané *řadiče přerušení*. U systémů s mikroprocesorem 8080 se nejčastěji používají tyto dva obvody:

– Obvod 8214 (MH 3214) je řadič přerušení, který zpracovává maximálně 8 žádostí o přerušení s pevně určenou prioritou, programově může

být zadávána prioritní úroveň povolených přerušení. Obvod vybere požadavek s nejvyšší prioritou a generuje přerušovací signál doprovázený přerušovacím vektorem (kódem instrukce RST *n*). Pro větší počet přerušení je obvody možné spojovat kaskádně. Blíže bude obvod popsán v čl. 7.11.

– Obvod 8259 (sovětský ekvivalent je KR580VN59) je programovatelný řadič přerušení. Může zpracovávat až 8 žádostí o přerušení (kasádním spojením lze tento počet zvýšit až na 64). Obvod může dynamicky (programově nebo automaticky) měnit prioritu jednotlivých požadavků, jednotlivé požadavky mohou být libovolně maskovány. Obvod generuje kód instrukce CALL (0CDH) následovaný adresou obslužného podprogramu, vybranou v přerušovací tabulce. Obvod lze používat ve čtyřech základních režimech s různými variantami.

### 3.3.3. Stykové obvody

Komunikace mezi mikropočítačem a periferním zařízením může být vedena na různé úrovni. U některých periferních zařízení pro přenos jediného bytu informace je třeba několik řídicích povelů, jiné na jeden povel přenesou až bloky o velikosti několik kilobytů. V mnoha složitějších zařízení je obsluha periferních zařízení svěřena dalšímu procesoru, ať již univerzálnímu nebo specializovanému. Např. firma Intel dodává pro řadu 8086 vstupní/výstupní procesor (IOP) I8089, který podle svého vnitřního programu zpracovává data z různých periferních zařízení a teprve takto předzpracovaná je předává např. do paměti mikropočítače. Snahou všech výrobců je odlehčit mikropočítač o tuto časově náročnou činnost a předat většinu těchto úkolů samotným periferním zařízením. Proto se ve světě objevuje množství pomocných stykových obvodů – univerzálních nebo specializovaných, které pomáhají při komunikaci mikropočítače s periferními zařízeními.

Mezi *univerzální obvody* vhodné pro mikroprocesor 8080 můžeme zařadit tyto obvody:

– Obvod 8212 (MH 3212) – 8bitový vstupní/výstupní obvod s třístavovým hradlovaným výstupem, vnitřní osmibitovou vyrovnávací pamětí a řídicími obvody. Obvod lze použít jako paměť, hradlovaný výstupní obvod nebo jako multiplexor.

– Obvod 8255 (MHB 8255) – programovatelný paralelní stykový obvod (UART) – slouží k připojení periferních zařízení mikropočítače k datové sběrnici mikroprocesoru. Všechny vlastnosti obvodu jsou dány programem, takže obvykle nejsou třeba další obvody. Obvod obsahuje 3 × 8 linek, které se programují ve dvou skupinách do třech různých druhů provozu (modů). Obvod bude blíže popsán v čl. 7.2.

– Obvod 8251 (MHB 8251) – programovatelný komunikační stykový obvod (USART) – je univerzální obvod pro sériový synchronní a asynchronní přenos dat z/do mikropočítače. Funkce obvodu může být naprogramována, je použitelný téměř pro všechny druhy přenosu dat. USART převádí paralelní informaci na straně mikropočítače na sériový tok dat (včetně synchronizace a zabezpečení) na straně periferních zařízení, a to i oběma směry současně. Blíže bude popsán v čl. 7.10.

Obdobou tohoto obvodu, ale s ještě bohatšími programovacími vlastnostmi je obvod Z80-SIO (verze NDR U856D), který lze rovněž připojit k mikroprocesoru 8080.

– Obvod 8253 (sovětský ekvivalent je KR580VI53) – programovatelný časovač – skládá se ze tří nezávislých 16bitových čítačů, které mohou být programovány v jednom ze šesti modů. Čítače čítají dolů, počáteční hodnota se zadává buď binárně, nebo dekadicky. Obvod je vhodné používat v systémech (periferních zařízeních) se silnými časovými vazbami (bude podrobněji popsán v čl. 7.12.).

– Obvod 8257 (sovětský ekvivalent je KR580IK57) – programovatelný řadič DMA – jde o čtyřkanálový řadič přímého přístupu do paměti sloužící rychlému přenosu bloků dat z/do paměti mikroprocesoru. Obvod má řadič priority periferních zařízení, na základě jejich žádosti generuje signály pro DMA.

Mezi částečně *specializované obvody* patří např.

– Obvod 8275 – programovatelný řadič obrazovkového displeje s řádkovým rozkladem. Obsahuje vyrovnávací paměť na dva řádky dat, obvody řízení kurzoru, světelného pera apod. Spojení s pamětí mikropočítače je realizováno kanálem DMA.

– Obvod 8271 – programovatelný řadič jednotky pružných disků – umožňuje řídit dvě jednotky pružných disků a provádět základní operace s bloky dat. Používá formát IBM.

– Obvod 8273 – programovatelný řadič protokolu HDLC/SDLC – umožňuje sériový přenos dat v daných formátech včetně synchronizace a zabezpečení.

Ve světě se objevují i další jednoduché univerzální obvody, které jsou vybaveny kombinacemi vstupních/výstupních linek, pamětmi (E)PROM a RAM (případně i s časovačem) – jsou to např. obvody 8155, 8355 aj.

Na vývojově vyšší úrovni než tyto obvody jsou *univerzální stykové procesory*. Jedním z prvních byl obvod 8741, který kromě procesoru kompatibilního s jednočipovým mikropočítačem 8048 obsahuje paměť EPROM s kapacitou 1 Kbyte a paměť RAM s kapacitou 64 byte, 18 vstupních/výstupních linek, 8bitový časovač a generátor hodinových impulsů. S tímto obvodem (vzhledem k tomu, že obsahuje kompletní mikropočítač) lze snadno realizovat i složité periferní operace.

Mezi nejpoužívanější stykové obvody v praxi patří i *převodníky A/D a D/A*. Převodníky byly dříve konstruovány většinou bez ohledu na spolupráci s mikropočítači. V poslední době se však ve světě objevily obvody připojitelné přímo na sběrnici mikropočítačů. Charakteristika obvodového řešení převodníků bude uvedena v čl. 7.6. a 7.7. Zde se však pouze zmíníme o integrovaném 8bitovém převodníku A/D firmy Intersil, sestaveném ze dvou obvodů – 8052 (analogová část) a 7103 (číslicová část). Tento převodník pracuje metodou postupné aproximace. K mikropočítači se připojuje přes paralelní nebo sériový stykový obvod.

Novějším členem rodiny 8bitových mikropočítačů je Intel 8022, s pamětí PROM s kapacitou 2 Kbyte a pamětí RAM s kapacitou 64 byte, třemi skupinami vstupních/výstupních obvodů s 28 linkami, 8bitovým časovačem a především převodníkem A/D se dvěma vstupními kanály.

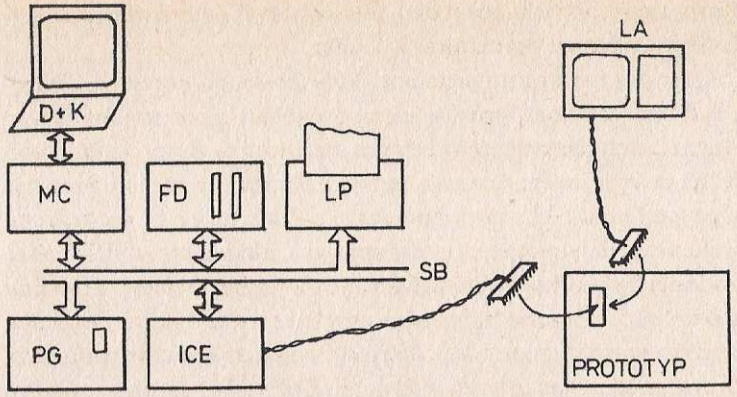
Dalším přínosem v oblasti číslicového zpracování analogových signálů jsou tzv. *signální procesory*. Příkladem těchto procesorů je obvod Intel 2920, který obsahuje v jednom čipu převodník A/D, mikroprocesor, operační paměť a převodník D/A na výstupu, takže může kompletně pracovat v analogovém prostředí. Nahradí tak celou řadu analogových obvodů, jako jsou filtry, oscilátory, modulátory apod., které mají většinou nižší přesnost (přesnost uvedeného procesoru je 9 bitů).

Na závěr našeho výkladu je třeba zdůraznit, že stykové obvody budou v blízké budoucnosti hrát rozhodující roli v nasazování mikropočítačů do různých aplikací ve výrobě a řízení technologických procesů, a proto je třeba jim věnovat zvýšenou pozornost.

## 3.4. PROSTŘEDKY PRO LADĚNÍ PROGRAMŮ, EMULACE

Se vzrůstající univerzálností mikropočítačových systémů hraje stále více rozhodující roli programové vybavení přístrojů a celků s mikropočítači. Nejnovější údaje světových výrobců uvádějí, že podíl nákladů






*Obr. 4.* Prostředky pro vývoj a ladění programů
D + K – displej + klávesnice (operátorský terminál), MC – mikropočítač vývojového systému, FD – jednotka pružných disků, LP – tiskárna, PG – programátor paměti EPROM, ICE – obvodový emulátor, LA – logický analyzátor, SB – systémová sběrnice

na programové vybavení se blíží 90 % celkových nákladů na vývoj mikropočítačových systémů.

Proto se věnuje mnoho prostředků na zefektivnění této činnosti. Vytváření programového vybavení můžeme rozdělit do těchto tří fází:

1. algoritmizace (určení způsobu řešení problému) a sestavení vývojových diagramů nebo systémové řešení ve formálních jazycích,
2. sestavení programů v určeném jazyce – základní úlohu zde hrají mikropočítačové *vývojové systémy* (popř. hostitelský počítač s křížovým programovým vybavením) vybavené redakčními programy (editory), příslušnými překladači, sestavovacím a zaváděcím programem,
3. ověření funkční správnosti vzniklých programů – odladění programů.

Je to nejdůležitější a také časově nejnáročnější fáze tvorby programového vybavení systémů. Pro usnadnění této činnosti je určeno mnoho pomůcek, jak programového, tak technického charakteru, např.:

a) Využívají se *podprogramy obsažené v monitoru vývojového systému* – pro spouštění programových sekvencí, jejich zastavování v určeném místě programu, výpis a změnu obsahu paměti i registrů, podprogramy pro zavádění programů a další pomocné podprogramy. Uživatelský program je uložen v paměti RAM vývojového systému a nemusí mít žádné spojení s mikropočítačovým systémem, pro nějž se vyvíjí programové vybavení. Všechny příkazy jsou zadávány z klávesnice, programátor často upravuje

uživatelské programy, vkládá kontrolní výpisy atd. Touto formou se odstraní především hrubé programátorské chyby.

b) Další fází může být kontrola pomocí *ladicího programu* (angl. *debugger*). Je to další programový prostředek pro ověření algoritmické správnosti a částečně i funkčnosti uživatelských programů. Bývá nadstavbou monitoru (ať už ve vývojovém systému, nebo (stále častěji) přímo v prototypu vyvíjeného systému). Má podobné funkce jako monitor (zobrazení, změny registrů a paměti, spouštění programu aj.), ale některé další funkce jsou zde přidány (zastavení běhu programu na mnoha různě určených bodech, krokování, trasování programu, simulace vstupních/výstupních bran pomocí klávesnice/displeje atd.). Největší výhodou ladicího programu je, že může pracovat i s programy v paměti (E)PROM. Je opět ovládán uživatelem z klávesnice.

c) Pomocným prostředkem při ladění na řádově vyšší úrovni u vývojových systémů bývá *obvodový emulátor* (In Circuit Emulator – ICE). Je to soubor programových a technických prostředků vývojového systému, který je schopen vykonávat programy vytvořené pro vyvíjený mikropočítač. Pro ovládání emulátoru jsou v programových prostředcích obsaženy programy pro řízení emulátoru. Emulátor umožňuje vykonávat programy vyvíjeného mikropočítače na technickém vybavení vývojového systému (nazýváme to *emulace*) nebo vykonávat tyto programy na vyvíjeném mikropočítači, ale pod řízením a kontrolou vývojového systému *(obvodová emulace)*. Emulátor se připojí do vyvíjeného mikropočítače pomocí sondy, která se zasune do konektoru místo mikroprocesoru, a tak umožní, aby vývojový systém sdílel s prototypem svůj procesor, část operační paměti i vstupní/výstupní obvody. Emulátor pak při zkoušení vyvíjeného mikropočítače umožňuje využívat vlastnosti operačního systému vývojového systému (výpisy paměti a registrů, trasování programu). Další výhodou emulace je, že technické vybavení vyvíjeného mikropočítače ještě nemusí být dokončeno, neboť části paměti a vstupních/výstupních obvodů mohou být nahrazeny příslušnými částmi zařízení vývojového systému. Emulátor umožňuje napodobovat práci vyvíjeného mikropočítače v reálném čase, popř. téměř v reálném čase (např. při vkládání malého počtu čekacích taktů do strojového cyklu instrukcí). Umožňuje však v požadovaných sekcích programu i trasovací režimy, např. s výpisem registrů po každé instrukci, které jsou ale několikanásobně pomalejší než v reálném provozu.

Emulátor umožňuje, obdobně jako monitor nebo ladicí program, nastavit body zastavení programu při dosažení určitých adres programu, ale kromě



toho i při zápisu/čtení na/z určené adresy paměti, vstupní/výstupní obvody. Při zastavení může emulátor vypsat i informace o strojových cyklech naposledy vykonávaných instrukcí. Emulátor ale umožňuje se na všechny adresy z testovaného programu odvolávat i symbolickými adresami (tj. tak, jak jsou nazvány ve zdrojovém programu). Dále umožňuje měřit časové závislosti a počet průchodů ve vyvíjeném programu. Je vybaven celou řadou zobrazovacích a testovacích funkcí. Stává se tak neodmyslitelnou součástí vývoje všech nových vzorků a prototypů systémů s mikropočítači.

d) Vhodným doplněním emulátoru pro kontrolu stavu technického vybavení vyvíjeného mikropočítače se stávají *mnohakanálové oscilografy* a *programové analyzátory*. Umožňují sledovat průběhy potřebných vnějších i vnitřních signálů vyvíjeného miropočítače. Některé programové analyzátory ale, kromě své obvyklé funkce zobrazení průběhu graficky nebo i číselně, mají v sobě vestavěny moduly disasembleru (zpětného překladu do jazyka symbolických adres), a stávají se tak neocenitelnou pomůckou pro programátory při ladění programů. Tyto přístroje nejsou tak rozsáhlé jako kompletní vývojové systémy, a proto jsou vhodné především pro užití přímo v existujících zařízeních, pro určování jak programátorských chyb, tak i většiny technických chyb. Obdobně jako emulátor bývají tyto přístroje připojeny k testovanému zařízení sondou, která je buď nasunuta na mikroprocesor, nebo ho přímo nahrazuje. Programové analyzátory umožňují většinou obdobné funkce jako emulátor – nastavení kontrolních bodů na libovolná místa programu, výpis několika desítek naposledy vykonávaných instrukcí v okolí těchto bodů, výpisy stavu obsahu vybraných paměťových míst nebo vstupů/výstupů nejen při zastavení programu, ale i dynamicky při jeho běhu, použití vnějších signálů k synchronizaci a sledování programu, zobrazení vybraných signálů spolu s probíhajícím programem, měření doby nebo počtu průchodů sekvencemi programu a mnoho dalších funkcí. Pokud je sonda pouze napojena na mikroprocesor, analyzátor nemá možnost do vykonávání programu zasahovat ani jej zpomalovat. Pokud sonda mikroprocesor nahrazuje, má uživatel možnost běh programu zastavovat, program krokovat apod. Jedinou nevýhodou je, že analyzátor nemůže měnit obsahy paměťových míst v zařízení, ani obsahy vnitřních registrů procesoru. Přesto se však stává neocenitelným pomocníkem při ladění programů i zjišťování technických chyb.

e) V současné době se již vytvářejí *mikropočítačové vývojové systémy*, které soustřeďují prostředky pro vývoj programového vybavení, emulátory i programové analyzátory do jediného kompaktního celku.

Všechny uvedené ladicí prostředky zvyšují značně efektivnost programátorské práce a lze doufat, že i posledně jmenované prostředky se u nás rozšíří v potřebném měřítku.

KONTROLNÍ OTÁZKY A ÚLOHY

1. Vyjmenujte základní části a obvody mikropočítače TEMS 80-03, nakreslete bloková schémata těchto obvodů.
2. Seznamte se podrobně se schématy mikropočítače, objasněte funkci jednotlivých částí.
3. Vyjmenujte typy pamětí mikropočítačů, zjistěte jejich kapacity u jiných typů mikropočítačů (IQ 151, PMD-85 apod.).
4. Obdobně roztřiďte i programové vybavení těchto mikropočítačů do skupin; kategorizujte i dostupné překladače.
5. Seznamte se podrobněji s podprogramy monitoru TEMS 80-03.
6. Vyjmenujte stykové obvody používané pro mikroprocesor 8080 a uveďte jejich základní určení.
7. Určete dostupné způsoby ladění uživatelských programů na jednotlivých mikropočítačích, vyjmenujte a charakterizujte ostatní způsoby.



# 4. Diagnostika a modelování

## 4.1. ZÁKLADNÍ POJMY A ÚLOHY DIAGNOSTIKY

Základní úlohou technické diagnostiky je zjistit technický stav daného objektu. Takovým objektem může být libovolný celek v rámci konstrukce systému – v číslicovém systému např. integrovaný obvod, zásuvná deska nebo základní jednotka počítače. Podle toho, zda výrobek je nebo není schopen plnit předepsanou funkci, rozlišujeme dva základní stavy, a to *bezporuchový* a *poruchový*.

Rozlišení poruchového a bezporuchového stavu je jednou ze základních úloh diagnostiky, které říkáme *detekce poruchy*. Nejde ale o žádné bližší určení poruchového stavu. Tato informace je dostatečná jen pro vyřazení příslušného výrobku. Pro potřeby údržby je třeba poruchu najít, a to s dostatečnou přesností tak, aby ji bylo možné odstranit. Této druhé základní úloze diagnostiky říkáme *lokalizace poruchy*. V údržbářské praxi se oprava většinou provádí formou výměny nějakého montážního celku (modulu). Úlohou lokalizace poruchy by tedy mělo být určení vadného modulu. Lokalizace poruchy s přesností na jeden výměnný modul je ale většinou velmi obtížná úloha. Detekce a lokalizace poruchy se řeší pomocí *diagnostických testů*.

(V této kapitole a v kapitole 8 se z důvodu zkrácení textu používá obrat „přivést na vstup hodnotu" místo správného „přivést signál s logickou hodnotou ... na vstup".)

Diagnostický test číslicového systému tvoří množina dvojic vzájemně přiřazených kombinací hodnot vstupních a výstupních proměnných. Jedna kombinace hodnot vstupních proměnných a jí odpovídající kombinace hodnot výstupních proměnných (odezva) se nazývá *krok testu* a počet kroků udává *délku testu*. Diagnostický test se provádí postupně, tj. po přivedení jedné kombinace hodnot vstupních proměnných a zjištění odezvy se přivádí na vstup testované jednotky kombinace hodnot vstupních proměnných odpovídající následujícímu kroku a zjišťuje se její odezva. Podle toho, jakým způsobem se vybírá následující krok, dělíme testy na nezávislé a závislé. V *nezávislém testu* je vstupní posloupnost testu pevně předem stanovena

a provádí se v předepsaném pořadí vždy celá až do konce. Pořadí kroků je tedy nezávislé na výsledcích předchozích kroků. Naproti tomu *závislý test* je charakterizován výběrem následujícího kroku testu v závislosti na výsledku předchozích kroků. Použití závislých testů je účelné pouze pro lokalizaci poruch.

V číslicových systémech hrají vedle hodnot logických proměnných důležitou roli i časové závislosti přenášených signálů, tj. délka impulsů, strmost hran impulsů apod. Je proto třeba rozlišovat *statické testy*, kdy měříme jen hodnoty logických proměnných v ustáleném stavu, a *dynamické testy*, při nichž zjišťujeme i zmíněné časové závislosti. Dynamický test je velmi obtížný, a proto se snažíme provádět dynamické testy součástek pokud možno před montáží a u hotového obvodu se omezit na statické testy.

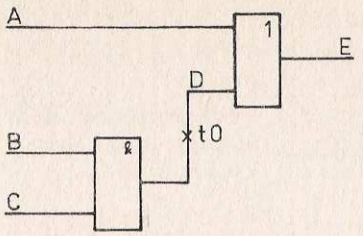
## 4.2. VOLBA MODELU PORUCHY

Pro potřebu diagnostiky rozlišujeme fyzikální a logickou poruchu. *Fyzikální poruchou* rozumíme určení skutečné změny parametrů, k níž v logickém obvodu došlo, např. přerušení napájecího vodiče, zkrat mezi vodiči apod. *Logická porucha* bude popisem fyzikální poruchy pomocí hodnot logických proměnných. Je potřeba si ale uvědomit, že pro všechny fyzikální poruchy nelze najít dostatečně přesné logické vyjádření. Cílem volby *modelu poruchy* je přiblížit se skutečnému chování co nejvíce.

Nejstarší model poruchy je reprezentace poruch pomocí trvalé nuly a trvalé jedničky. *Trvalá nula* (t0) spočívá v přerušení vodiče přivádějícího signál do místa poruchy a v jeho náhradě zdrojem napětí odpovídajícím logické úrovni 0. Podobně *trvalá jednička* (t1) vznikne přerušením vodiče a jeho náhradou zdrojem napětí odpovídajícím logické úrovni 1. Poruchy typu t0 a t1 se zásadně přiřazují vodičům zkoumaného obvodu. Náhradní logické schéma poruchového obvodu pak získáme tak, že o všech použitých logických členech předpokládáme, že pracují správně, a na vodič postižený poruchou dosadíme hodnotu, která se tam v důsledku poruchy trvale objeví. Příklad takové situace vidíme na obr. 5 a v tab. 4, kde je poruchou t0 postižen vstup $D$ výstupního logického součtu. I když vstupní součin pracuje správně, je na vodiči $D$ trvale nula, takže výstup $E$ se trvale rovná hodnotě $A$.

Velmi důležitým problémem je i počet poruch, které se v testovaném obvodu mohou vyskytnout současně. Značné množství diagnostických me-






*Obr. 5.* Logický obvod s poruchou t0

*Tabulka 4.* Pravdivostní tabulka logického obvodu s poruchou t0 ($D = 0$)

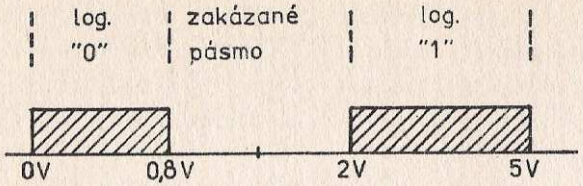
| $A$ | $B$ | $C$ | $E$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

tod je založeno na předpokladu výskytu *jednotlivých poruch*. Výskyt většího počtu poruch může detekci nepříznivě ovlivnit jen výjimečně, ale lokalizace může být vážně zkomplikována.

Volbu vhodného modelu poruchy komplikuje také časová nestálost některých fyzikálních poruch. Tyto poruchy nazýváme *nestálé poruchy*. Příkladem může být studený spoj, kolísání napájecího napětí apod. Porucha, která ovlivňuje funkci obvodu stále stejným způsobem, bude označována jako *stálá*. Je důležité připomenout, že trvalost a stálost poruchy jsou dvě nezávislé vlastnosti, tedy např. trvalá jednička může být stálá, je-li přívod zcela přerušen, nebo nestálá, jestliže se ulomený vodič dotýká kontaktu a mění svůj přechodný odpor.

Jako poruchy t0 a t1 lze popsat značnou část fyzikálních poruch. Patří sem především všechny zkraty mezi signálním a napájecím vodičem (t1) a mezi signálním a zemnicím vodičem (t0). Poruchy uvnitř základních logických členů TTL (integrovaných obvodů) vedou buď ke změně logické funkce, kterou lze ve většině případů modelovat poruchami t0 a t1, nebo k degeneraci parametrů, která se jako změna logické funkce neprojeví. Poruchy mimo základní logické členy, tj. na jejich propojovacích vedeních, lze poruchami typu t modelovat ve všech případech kromě zkratu mezi signálními vodiči. Přerušení vstupního vodiče je ekvivalentní poruše t1 na vstupu, zatímco přerušení napájecího nebo zemnicího vodiče je ekvivalentní poruše t1 na výstupu (při připojení na vstup správně pracujícího členu).

Zkrat mezi signálními vodiči se projeví jako chyba samozřejmě jen tehdy, je-li na každý ze zkratovaných vodičů přivedena jiná logická hodnota. Vzhledem k tomu, že každý vodič je napájen výstupem nějakého logického




*Obr. 6.* Rozlišovací úroveň napětí pro logické hodnoty v obvodech TTL

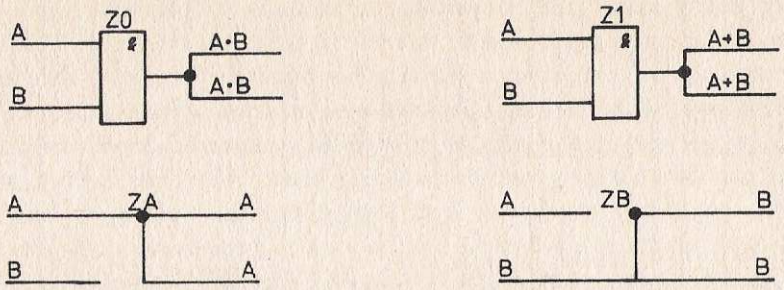
*Tabulka 5.* Pravdivostní tabulka poruch typu z

| Původní hodnoty | | Hodnota v místě zkratu A – B | | | |
|---|---|---|---|---|---|
| A | B | Z0 | ZA | ZB | Z1 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

členu, je třeba každý zkrat vidět především jako propojení dvou výstupních obvodů logických členů. Podle toho, jakého typu jsou tyto výstupní obvody, bude hodnota zkratovaného signálu rovna hodnotě na jednom z nich nebo se ustálí v zakázaném pásmu (obr. 6). Napájí-li zkratovaný signál kaskádu logických členů, dojde s největší pravděpodobností k regeneraci signálu, protože vstupní obvody každého stupně rozlišují pouze jednu prahovou úroveň. Zkraty tedy vykazují jednu ze čtyř variant chování, popsaných v tab. 5. Jde o zkrat vodičů A a B. Chování popsané v prvním sloupci odpovídá logickému součinu a můžeme je označit jako převahu nuly (označujeme Z0). Chování uvedené v posledním sloupci (součtové), označujeme jako převahu jedničky (Z1). Zbylé dva sloupce popisují převahu vodiče A (ZA), popř. B (ZB). To znamená, že vodič A, popř. B vnutil druhému vodiči svoji hodnotu logické proměnné bez ohledu na to, jaká hodnota to byla. Náhradní logická schémata pro uvedené čtyři typy zkratového chování jsou na obr. 7.

Z popsaných vztahů je zřemé, že zkrat nelze popsat jako poruchu typu t. Z toho vyplývá nová otázka, a to do jaké míry je třeba respektovat možnost výskytu zkratů při generování testů pro logické obvody, tj. do jaké míry vystačíme s modelem t0 a t1. Podrobný rozbor ukazuje, že zkraty mezi vstupními vodiči různých logických členů a zkraty na vodičích které vstu-






*Obr. 7.* Náhradní logická schémata poruch typu z

pují do jednoho logického členu, ale přitom se větví, nelze spolehlivě detektovat modelem t0 a t1. Ostatní zkraty, tj. zkraty na nevětvících se vstupech jednoho logického členu a zkraty mezi vstupem a výstupem logického členu nebo složitějšího obvodu lze detektovat modelem typu t. V některých případech zde ale záleží na pořadí kroků testu (zkratem vstupu a výstupu může vzniknout paměťový člen) a vhodným uspořádáním těchto kroků můžeme tyto zkraty detektovat modelem typu t. V dalším výkladu budeme pro zjednodušení používat model poruchy t0 a t1.

## 4.3. TVORBA DIAGNOSTICKÝCH TESTŮ

Podstatou všech metod tvorby diagnostických testů je vytváření *citlivé cesty* z místa předpokládané poruchy na výstup. Rozdíl mezi těmito metodami spočívá zejména ve způsobu hledání citlivé cesty – v tom, zda se hledá jedna nebo více citlivých cest –, a v systematičnosti postupu. V této učebnici se budeme zabývat pouze metodou, která se nazývá *intuitivní zcitlivění cesty*. Jak naznačuje sám název, není tato metoda systematická, takže nezaručuje nalezení výsledku, i když je známo, že řešení existuje. Vychází především ze zkušenosti pracovníka, který test sestavuje a přitom klade značné nároky na jeho pozornost. Náročnost přirozeně roste se složitostí obvodu, především s počtem citlivých cest vedených paralelně.

Jako *cestu* ve schématu budeme označovat posloupnost vodičů a logických členů, která začíná a končí vždy vodičem a v níž jsou každé dva po sobě následující členy propojeny vodičem. *Citlivá cesta* je cesta, která je schopna přenášet změny logické proměnné ze svého začátku na svůj konec.

Smyslem použití citlivé cesty je přenášení změn hodnot logických proměnných na výstup přístupný našemu měření, tj. např. na konektor. Tento *výstup* budeme nazývat *primární*. Analogicky budeme termínem *primární vstup* označovat vstup, na němž jsme schopni zadat hodnotu logické proměnné. Pokud cesta není citlivá, je *blokována*. V bezporuchovém obvodu může citlivou cestu zablokovat pouze logický člen, nikoliv vodič. Proto si odvodíme podmínky průchodu citlivé cesty základními logickými členy. Citlivá cesta prochází ze zvoleného vstupu $n$-vstupového členu součtového typu (OR, NOR) na výstup, přivedeme-li na $n$-1 zbývajících vstupů hodnotu 0. V tomto případě změna logické hodnoty na zvoleném vstupu způsobí změnu logické hodnoty na výstupu. Pokud by ale na některém ze zbývajících $n$-1 vstupů byla logická hodnota 1, nebude se logická hodnota na výstupu měnit, cesta je blokována. Podobně ze vstupu $n$-vstupového členu součinového typu (AND, NAND) prochází citlivá cesta na výstup, jsou-li na všech $n$-1 zbývajících vstupech přivedeny hodnoty 1. Invertorem (negátorem) prochází citlivá cesta vždy.

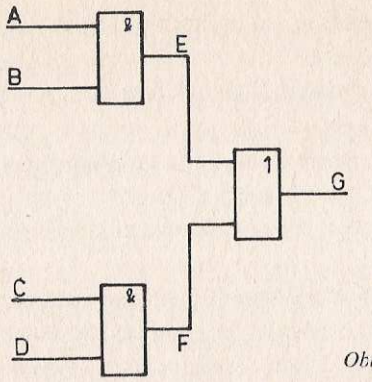
Jestliže se hodnota logické proměnné na konci citlivé cesty mění při změně na začátku cesty, musí se změnit i při změně v libovolném místě této cesty. Proto je změnou na výstupu detektována také porucha v libovolném místě této cesty. Postup jednoho *kroku testu* pro určité poruchy lze popsat takto:

1. Volba poruchy, která má být detektována.
2. Přivedení hodnoty 0 do místa výskytu poruchy t0.
3. Zcitlivění cesty z místa výskytu poruchy na primární výstup obvodu.
4. Odvození hodnot proměnných na primárních vstupech obvodu.
5. Nalezení všech poruch, které odhaluje (pokrývá) sestavený krok testu.

I když popsaný postup působí na první pohled přímočaře, jde o metodu zkoušek a omylů, zejména při provádění bodů 3 a 4. Zde lze totiž najít většinou celou řadu alternativních řešení. Požadavek přivést do místa výskytu poruchy hodnotu opačnou k předpokládané hodnotě je důsledek práce s dvouhodnotovými proměnnými. Pokud se tedy chceme přesvědčit, že nevznikla porucha t1 ani t0, musíme do testovaného místa přivést hodnotu 0 i 1.

Tímto způsobem najdeme vždy jeden krok testu pro určitou zvolenou poruchu. *Úplný test*, jehož sestavení se v praxi nejčastěji požaduje, vznikne spojením kroků detektujících všechny poruchy zvoleného typu. Přitom krok testu odvozený pro jednu poruchu detektuje současně celou řadu dalších poruch.






*Obr. 8.* Příklad obvodu bez rozvětvení

Chceme-li se vyhnout zbytečnému vytváření nadbytečných kroků testu, je třeba pro každý vytvořený krok vyhodnotit, které poruchy *pokrývá* (detektuje). Seznam pokrytých poruch je třeba průběžně doplňovat po každém nově vytvořeném kroku a poruchu, která má být detektována v dalším kroku, vybírat vždy pouze z dosud nepokrytých poruch. Tak se postupně snažíme pokrýt všechny poruchy. Po každém vyhodnocení následuje rozhodnutí, zda se má pokračovat ve vytváření dalšího kroku nebo ne.

Výsledkem tohoto postupu je test, jehož délka nemusí být minimální a navíc v něm mohou být i nadbytečné kroky. Z hlediska praktických požadavků je však jeho délka většinou dostatečně blízká minimu, takže nebývá nutné jej zkracovat.

Při vytváření úplných testů pro určité typy kombinačních logických obvodů lze tento postup zjednodušit na základě *vět o úplnosti testu*, které nám umožňují předběžně redukovat množinu sledovaných poruch.

*První věta* říká: Test pro kombinační logický obvod bez rozvětvení, sestavený ze základních logických členů, je úplný vzhledem k poruchám t0 a t1, jestliže detektuje všechny poruchy t0 a t1 na primárních vstupech.

Důkaz této věty je velmi jednoduchý. Od každého vstupního vodiče vede na primární výstup pouze jedna cesta. Ta je během testu nejméně dvakrát zcitlivěna, přitom přenáší na výstup dvě různé hodnoty, takže na každém vnitřním vodiči jsou detektovány poruchy t0 a t1.

*Příklad.* Sestavte úplný diagnostický test pro obvod zadaný na obr. 8.

Podle uvedené věty v tomto případě pro úplnost testu stačí, aby test detektoval poruchy t0 a t1 na všech primárních vstupech. První krok testu si sestavíme takto:

1. na vodiči $A$ volíme poruchu t0;

2. na vodič $A$ přivedeme hodnotu 1;

3. citlivou cestu z vodiče $A$ na vodič $E$ zajistí hodnota $B=1$, citlivou cestu z $E$ na $G$ zajistí hodnota $F=0$;

4. hodnotu $F=0$ lze realizovat třemi způsoby, a to následujícími dvojicemi hodnot $C$ a $D$: $CD=10$, $CD=01$ nebo $CD=00$; zvolme $CD=10$;

5. kromě poruchy t0 na vodiči $A$ je tímto krokem testu detektována také porucha t0 na vodiči $B$.

Při detekci poruchy t1 na vodiči $A$ můžeme k realizaci citlivé cesty z vodiče $A$ na vodič $G$ opět použít kombinaci tří hodnot $BCD=110$. Mění se pouze hodnota na vodiči $A$ na 0. Tímto krokem testu se dále detektuje porucha t1 na vodiči $D$, protože cesta $DFG$ byla zcitlivěna také. Pro detekci poruchy t1 na vodiči $B$ musíme použít hodnotu $B=0$, dále zcitlivit cestu $BE$ vstupem $A=1$ a cestu $EG$ hodnotou $F=0$. Pro její realizaci bychom opět mohli použít kombinaci $CD=10$ jako v předchozích krocích, ale v zájmu zkrácení délky testu můžeme současně testovat i poruchu t1 na vodiči $C$ vstupem $CD=01$. Poslední dvě nedetektované poruchy jsou poruchy t0 na vodičích $C$ a $D$. Cestu $CF$ (a zároveň $DF$) zcitliví kombinace $CD=11$. Průchod citlivé cesty výstupním součtovým členem zajistí hodnota $E=0$, získaná např. z kombinace $AB=01$.

*Tabulka 6.* Test zadaného obvodu

| $A$ | $B$ | $C$ | $D$ | $G$ | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | t0 | t0 | | |
| 0 | 1 | 1 | 0 | 0 | t1 | | | t1 |
| 1 | 0 | 0 | 1 | 0 | | t1 | t1 | |
| 0 | 1 | 1 | 1 | 1 | | | t0 | t0 |

Úplný test má čtyři kroky zapsané v tab. 6. Každý krok testu je zde zapsán na jednom řádku. Je zde uvedena hodnota vstupních proměnných, hodnota výstupní proměnné za předpokladu bezporuchového stavu obvodu a seznam poruch detektovaných příslušným krokem testu. Můžeme se snadno přesvědčit o tom, že tento test detektuje i poruchy t0 a t1 na vodičích $E$, $F$ a $G$.

Ve všech krocích testu jsme zcitlivěli dvě cesty, k čemuž došlo v některých případech neúmyslně (např. v prvním kroku), jindy záměrně (např. ve třetím kroku). To je velmi častý úkaz a je zřejmé, že čím více cest je zcitlivěno v jednom kroku, tím je výsledný test kratší, ale také tím obtížnější je lokalizace poruchy.



Jestliže se v obvodu vodiče rozvětvují, existují body, z nichž lze vést citlivou cestu na primární výstup několika způsoby. Proto je zde i podmínka úplnosti testu složitější.

Pro tento případ platí *druhá věta o úplnosti testu:* Test pro kombinační logický obvod sestavený ze základních logických členů je úplný vzhledem k poruchám t0 a t1, jestliže detektuje poruchy t0 a t1 na všech primárních vstupech a na všech větvích vodičů, které se rozvětvují.

Důkaz je analogický důkazu první věty. Test můžeme opět považovat za úplný, jestliže dokážeme, že každým bodem, který nebyl samostatně testován, musí procházet citlivá cesta od některého testovaného bodu na primární výstup. Od každého primárního vstupu je až k nejbližšímu bodu rozvětvení citlivá cesta určena jednoznačně. Testem primárního vstupu jsou tedy pokryty také všechny poruchy vodičů v tomto úseku cesty. V místě rozvětvení máme možnost volit směr citlivé cesty, a proto samotný test bodů před rozvětvením není zárukou, že budou testovány obě větve. Proto je ve druhé větě o úplnosti testu uveden požadavek, že je třeba provést test všech větví. Testem začátku větve jsou pokryty poruchy všech vodičů až k nejbližšímu dalšímu bodu rozvětvení.

Vytváření testu pro kombinační logický obvod s rozvětvenými vodiči si ukážeme při praktickém cvičení (čl. 8.1). Testování sekvenčních logických obvodů představuje podstatně složitější problém než testování kombinačních logických obvodů. Existence vnitřního stavu komplikuje detekci i lokalizaci poruchy, protože její projev na výstupu se může zpozdit o několik taktů. Jako nový problém vzniká také otázka výchozího stavu obvodu, protože určitý, předem připravený test dá ve správně fungujícím obvodu různé odezvy pro různé výchozí stavy. Vzhledem k obtížnosti tohoto problému se zde jeho řešením nebudeme zabývat.

Po sestavení testu se může daný test minimalizovat, a to pouze přibližně, což je mnohem snazší. Pro *minimalizaci testů* potřebujeme tabulky poruch.

*Tabulka poruch* (někdy nazývaná také *matice poruch*) je dvourozměrné pole hodnot, v němž je každému kroku testu vyhrazen jeden řádek a každému poruchovému stavu nebo kombinaci poruchových stavů je vyhrazen jeden sloupec.

Pro hodnotu $f_{ij}$, zapsanou do průsečíku $i$-tého řádku a $j$-tého sloupce, platí, že $f_{ij}=1$, jestliže $i$-tý krok testu detektuje $j$-tou poruchu. V opačném případě platí $f_{ij}=0$. Tuto tabulku je možné teoreticky rozšířit i na kombi-

nace poruch, tj. přiřadit další sloupce poruchovým stavům vyvolaným dvojicemi, trojicemi až $k$-ticemi poruch. Tabulka poruch nám tedy udává, které poruchy jsou detektovány jednotlivými kroky testu. Vytváření tabulky poruch si ukážeme při praktickém cvičení (čl. 8.2).

Tabulkou poruch jsme si úlohu minimalizace testu převedli na řešení problému, kolik poruch test pokrývá. Omezující podmínkou minimalizace testu je to, aby pokrýval všechny poruchy. To znamená, že v tabulce poruch hledáme nejmenší podmnožinu řádků, která má v každém sloupci alespoň jednu jedničku.

Při praktickém cvičení (čl. 8.3) se budeme zabývat minimalizací nezávislých testů. Minimalizace závislých testů je složitější problém, zabývat se jím nebudeme.

V předchozích odstavcích jsme se seznámili s vytvářením a minimalizací testů. Ověření správnosti a úplnosti testu provádíme *simulací testu*. Teprve test, jehož vlastnosti jsme zkontrolovali simulací, můžeme považovat za *věrohodný*. Za nejdůležitější kontrolu při simulaci lze označit *kontrolu odezev testu*. Z hlediska detekce poruch je velmi důležitá *kontrola úplnosti testu* vzhledem k typům poruch, které v testované jednotce za daných podmínek považujeme za možné.

Obecně můžeme rozlišovat tři formy simulace, a to ruční, fyzikální a číslicovou. *Ruční simulace* spočívá v odvození potřebných hodnot na výstupech všech logických členů pomocí papíru a tužky. Je to postup téměř shodný s ručním vytvářením testu, a je tedy vhodné, aby kontrolu prováděl jiný pracovník než autor testu. I v tomto případě jde o postup značně nespolehlivý, náročný na čas, pozornost a soustředění.

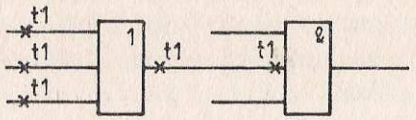
*Fyzikální simulace* se provádí přímo pomocí testované jednotky. Zde vlastně nejde o simulaci v pravém slova smyslu, protože pro ověření chování testované jednotky nepoužíváme model, ale jednotku samu. Provádíme-li pouze kontrolu nebo odvozování odezev, není třeba v testované jednotce nic měnit, ale používáme-li fyzikální simulaci při kontrole úplnosti testu, musíme postupně vyvolat v testované jednotce všechny poruchové stavy, které má test detektovat nebo lokalizovat. Vytváření poruch může mít škodlivé důsledky pro použité součástky. Jinak je to ale metoda velmi spolehlivá.

*Číslicovou simulací* nazýváme simulaci prováděnou pomocí počítače. Její použití při kontrole testu dává nejrychlejší a nejspolehlivější výsledky. Mezi její četné výhody patří i to, že umožňuje jednoduše vytvářet podklady pro údržbu, zejména *slovníky poruch*. Podstatou simulace je zde sestavení



*modelu*, který má stejné chování jako modelovaný objekt. Číslicová simulace se často používá také při návrhu číslicových systémů (čl. 4.4). Kontrola testů číslicovou simulací navazuje na funkční simulaci na úrovni logických členů a liší se od ní tím, že kromě zjišťování odezev zahrnuje navíc především kontrolu úplnosti testu.

Ověřený test se používá pro *lokalizaci poruchy*. Technik pověřený opravou číslicového systému potřebuje pro svoji práci dostatečně přesné informace o poruše, která narušila správnou funkci systému. Přesnost těchto informací závisí na úrovni, na které se oprava provádí, tj. na velikosti výměnné jednotky. V současné technologii je při opravě výměnnou jednotkou obvykle zásuvná deska. Tím je dána požadovaná přesnost lokalizace na této úrovni, protože označení poruchové desky je veškerá informace, kterou technik ke své práci potřebuje.




*Obr. 9.* Příklad obvodu s nerozlišitelnými poruchami

Jestliže jsme poruchovou desku vyjmuli ze systému, přichází na řadu její oprava. Při testování zásuvné desky obvykle požadujeme lokalizaci poruch s přesností na jednu součástku. Tento požadavek na lokalizaci poruch většinou nelze splnit. Údaje získané vyhodnocením testu většinou specifikují poruchu podle jiných hledisek, než vyžaduje údržba. Model poruchy t0 a t1 přiřazuje poruchy jednotlivým vodičům, nerozlišuje tedy mezi „začátkem" a „koncem" vodiče. Porucha vodiče, který spojuje výstup logického členu $A$ se vstupem členu $B$, se může ve skutečnosti vyskytovat přímo na výstupu členu $A$ (uvnitř jeho pouzdra), na vývodu pouzdra (nedokonale zapájený spoj), na plošném spoji, který vývody obou členů propojuje, na vývodu členu $B$ nebo uvnitř pouzdra členu $B$.

Dále nejsme schopni blíže lokalizovat poruchu v rámci nevětvících se vodičů propojujících jednotlivé logické členy. Jestliže např. vznikne porucha t1 na výstupu členu logického součtu v obvodu z obr. 9, nejsme schopni ji odlišit od ostatních poruch t1 označených na tomto obrázku. Existence nerozlišitelných poruch je velmi nepříjemná, neboť znemožňuje použít výsledky automatické lokalizace přímo jako podklady pro opravu. Mezi nerozlišitelnými poruchami je třeba poruchu najít na základě poloautomatických nebo ručních měření prováděných pomocí logické sondy nebo složitějších měřicích přístrojů.

Pro potřebu dalšího zpracování údajů při vyhodnocování výsledků testu, tj. pro lokalizaci poruchy, se používají *slovníky poruch*. Slovníky poruch představují v současné době nejrozšířenější nástroj automatické lokalizace poruch. Každý slovník poruch je seznam poruch, které jsou utříděny podle určitého kritéria. Objem slovníku poruch roste velmi rychle s požadovanou přesností lokalizace. Se slovníkem poruch se seznámíme v praktickém cvičení (čl. 8.4).

## 4.4. MODELOVÁNÍ (SIMULACE) ČÍSLICOVÝCH ZAŘÍZENÍ

Pojmem modelování rozumíme cílevědomou činnost, která slouží k získání informací o jednom systému (*originálu*) pomocí jiného systému (*modelu*). Modely lze v zásadě rozdělit na *fyzikální* (nějaký reálný systém, např. malé letadlo v aerodynamickém tunelu) a *matematické* (např. soustava diferenciálních rovnic). Matematické modely bývají ještě rozčleňovány na *analytické* nebo *numerické*. Numerické matematické modely bývají také nazývány *simulační modely*.

Pojem modelování (simulace) číslicových zařízení je velmi obecný. Je možné provádět simulaci na různých úrovních. Nejčastěji rozlišujeme tyto úrovně:

1. fyzikální simulace;
2. funkční simulace;
3. systémová simulace.

Zmíněné úrovně můžeme charakterizovat takto:

### Fyzikální simulace

Jde o nejpodrobnější úroveň vyšetřování reálného zařízení – úroveň jednotlivých elektronických součástek. Lze říci, že tato úroveň je opodstatněná u „menších celků", např. při návrhu integrovaných obvodů. Cílem simulace na této úrovni je získat příslušné průběhy proudů a napětí elektronických obvodů.

### Funkční simulace

Tato úroveň znamená jistou abstrakci. Není nutné popsat zkoumané zařízení z hlediska fyzikálních zákonů, ale z hlediska chování. Funkční simulace může být prováděna na několika úrovních. Např. na úrovni



logických členů, registrů apod. Při simulaci na úrovni logických členů se používají logické proměnné (vstupní, výstupní, vnitřní). Funkční simulace na vyšších úrovních představuje další zjednodušení. Není třeba brát v úvahu kompletní logický návrh zařízení, ale stačí uvážit popis operací (funkcí), které simulované zařízení provádí.

### Systémová simulace

Tímto názvem rozumíme simulaci, která představuje obecný pohled na uvažované zařízení. Zde již neuvažujeme skutečné informace zpracovávané v daném zařízení, ale bereme v úvahu pouze četnost různých nároků na určitou část zařízení, zkoumáme otázky využití, zdržení požadavků ve frontách apod.

Systémovou simulaci můžeme použít i v případě, kdy na jednom počítači simulujeme jiný počítač, např. při tvorbě programového vybavení pro nově navrhovaný, zatím nedokončený počítač. Dalším příkladem použití je ladění programů (zapsaných v jazyku symbolických adres) v diagnostickém režimu. Zde je možné získat podrobné informace o průběhu laděného programu, což usnadní hledání chyb v tomto programu.

Při praktickém cvičení (čl. 8.5 až 8.8) se budeme zabývat simulací mikroprocesoru 8080. Při těchto praktických cvičeních se kromě vlastní simulace seznámíte i s editorem, pomocí něhož budete zapisovat a opravovat programy, a také s principem překladače jazyka symbolických adres. Ukážeme si, jak je možné vytvářet programové vybavení v jazyku symbolických adres pro mikroprocesorové systémy s mikroprocesorem 8080.

### KONTROLNÍ OTÁZKY A ÚLOHY

1. Jaký je rozdíl mezi lokalizací a detekcí poruchy?
2. Jaké známe modely poruch?
3. Je možné popsat zkrat mezi signálními vodiči jako poruchu typu t?
4. Jaký je princip intuitivního zcitlivění cesty při vytváření diagnostických testů?
5. Jak zjistíme úplnost testu pro obvod bez větvení?
6. Jak zjistíme úplnost testu pro obvod, ve kterém se vodiče větví?
7. Jakým způsobem minimalizujeme test?
8. Co to jsou nerozlišitelné poruchy?
9. K čemu se používá slovník poruch?
10. Co to je modelování (simulace)?
11. Na jakých úrovních můžeme simulovat činnost číslicových zařízení?

# 5. Základní jednotka počítače ADT 4000 — cvičení

Při práci počítače vytváří časovací generátor cyklus paměti délky 3,2 μs, obsahující povely „čtení" a „zápis" do operační paměti. Řadič počítače ADT 4000 rozlišuje čtyři základní fáze vykonávání instrukce. Jsou to *přípravná fáze* (Fetch), *fáze nepřímého adresování* (Indirect), *prováděcí fáze* (Execute) a *fáze přerušení* (Interrupt), z nichž první tři spolupracují s operační pamětí. Pořadí fází není předem dáno, ale určuje se během výpočtu podle nastávajících podmínek. Přechod řadiče počítače z jedné fáze do druhé je znázorněn na obr. 10. Přitom vnější zařízení vyvolá přechod do fáze přerušení po skončení kterékoliv fáze.

| Fáze počítače | Cyklus paměti čtení zápis | Rozhodování o následující fázi |
|---|---|---|
| fetch | probíhá | fetch<br>indirect<br>execute |
| indirect | probíhá | execute<br>indirect<br>fetch |
| interrupt | nemá cyklus paměti | fetch (na adrese přerušení) |

*Obr. 10.* Přehled fází řadiče počítače ADT

Základní fází řadiče počítače je přípravná fáze. Touto fází začíná kterákoliv instrukce a u větší části instrukcí se i plně během ní vykoná.

a) *Přípravná fáze*

Během této fáze se ve čtecí části cyklu paměti přečte obsah právě adresované buňky paměti do registru T a v zápisové části cyklu paměti se zapíše zpět do paměti. Takto získaná informace v registru T se interpretuje jako instrukce. V případě, že přečtená instrukce patří do skupiny instrukcí s adresováním paměti MR (Memory Reference) a její 15. bit je jedničkový



(bit přímého/nepřímého adresování), nastaví se v počítači podmínky pro přechod do fáze nepřímého adresování. Není-li v případě instrukcí skupiny MR 15. bit jedničkový, následuje prováděcí fáze. V ostatních případech se instrukce nastavená v registru T plně provede během přípravné fáze a řadič počítače zůstává v této fázi i pro další cyklus paměti. Výjimku tvoří pouze instrukce JMP (nepodmíněný skok), která je zahrnuta ve skupině instrukcí s adresováním paměti, ale nepotřebuje prováděcí fázi. Instrukce JMP se vykoná během přípravné fáze nebo fáze přerušení a potom se nastaví podmínky opět pro přípravnou fázi, ve které následuje další cyklus paměti.

b) *Fáze nepřímého adresování*

Obsah paměťové buňky, jejíž adresa je určena během přípravné fáze adresní části instrukce, se přečte do registru T. Potom celé šestnáctibitové slovo v registru T (bity 0 až 14 adresy a bit 15 — bit přímého/nepřímého adresování) znamená novou adresní část původní instrukce. Použití patnácti bitů pro adresování dovoluje adresovat maximální možnou kapacitu paměti (32 K slov). Jestliže 15. bit opět určí nepřímé adresování, zůstane řadič počítače nastaven ve fázi nepřímého adresování pro další cyklus paměti a přečtené šestnáctibitové číslo je opět adresou paměti. Tento proces (multiadresování) skončí vynulováním bitu přímého/nepřímého adresování. Tím vzniknou podmínky pro přechod řadiče počítače do prováděcí fáze, pouze v případě instrukce JMP *X*, I (nepodmíněný skok s nepřímou adresou) do fáze nepřímého adresování.

c) *Prováděcí fáze*

Během této fáze se do registru T přečte operand, jehož adresa se určila během předchozí fáze (buď přípravné fáze, nebo fáze nepřímého adresování) a příslušná instrukce se vykoná. Po skončení této fáze přechází řadič počítače opět do přípravné fáze, aby přečetl z operační paměti další instrukci.

d) *Fáze přerušení*

Požádá-li o obsluhu některé periferní zařízení a není-li přerušení blokováno z nějakého důvodu právě probíhajícím programem, jsou splněny podmínky pro přechod počítače do fáze přerušení. Přitom do této fáze přejde řadič počítače po skončení kterékoliv právě probíhající fáze. Fáze přerušení neobsahuje žádné cykly paměti. Aby se nepřeskočila žádná instrukce v hlavním programu ani nevykonala dvakrát, sníží se obsah čítače instrukcí během fáze přerušení o jedničku. Na konci této fáze se adresa přerušujícího zařízení přenese do registru M. Zároveň se nastaví

přípravná fáze, aby se při následujícím cyklu paměti přečetla instrukce, kterou přerušující zařízení vysílá. Dokud se tato instrukce nevykoná, nemůže dojít k dalšímu přerušení.

## 5.1. REALIZACE VYBRANÝCH INSTRUKCÍ

### Zadání

Podle následujícího výkladu zaveďte do počítače posloupnost instrukcí a tisknutím tlačítka SINGLE CYCLE kontrolujte vykonávání jednotlivých instrukcí, stav registru T a registrů M a P, u některých instrukcí také stav registrů A a B. Po bezchybném vykonání instrukcí byste měli pokračovat od začátku, nebo pokud je to uvedeno, měl by se nastavit kód instrukce HLT 77B v registru T, tj. hodnota $1Ø2Ø77_8$. Pokud se nastaví instrukce HLT *ØX*, byl program proveden nesprávně, a je nutné hledat chybu.

Prováděním jednotlivých instrukcí krok po kroku pomocí tlačítka SINGLE CYCLE se provede *statická zkouška*. Po úspěšném ověření se provede *dynamická zkouška* tím, že se stiskne tlačítko RUN. Při vadně provedené instrukci se počítač zastaví s indikací HLT *ØX*. Pokud se počítač zastaví s indikací HLT 77B, program proběhl v pořádku; v tomto místě se má provést změna vstupních parametrů nebo provést kontrola.

### Popis cvičení

V následujících úlohách budeme do počítače zadávat řadu instrukcí od jednoduchých až po složitější s tím, že dále budeme používat instrukce již ověřené. Tímto způsobem postupně ověříme celou řadu instrukcí.

Vkládání instrukcí do počítače bude probíhat takto:

1. Na registru SW (registru stavového slova) nastavíme adresu 2ØØB, stiskneme tlačítko LOAD ADDRESS.

2. Na registru SW nastavíme žádanou instrukci a stiskneme tlačítko LOAD MEMORY. Tento krok opakujeme tak dlouho, dokud nenahrajeme celý sled instrukcí.

Po nahrání instrukcí budeme kontrolovat jejich provádění. Nejprve nastavíme adresu 2ØØB. Potom budeme tisknout tlačítko SINGLE CYCLE a kontrolovat obsah registrů (statická zkouška). Při dynamické zkoušce



opět nastavíme adresu 2ØØB a stiskneme tlačítko RUN. U každého příkladu je napsáno, co se v tomto případě děje – zda počítač neustále opakuje sled instrukcí – cykluje – nebo zda se zastaví.

*Poznámka:* Hvězdička v symbolickém zápisu instrukcí znamená autorelativní adresování (vztažené k okamžité hodnotě čítače instrukcí).

a) *Instrukce* NOP (No Operation) – žádná operace. Při této instrukci se neprovede žádná operace, pouze se zvětší obsah čítače instrukcí.

```
ØØ2ØØ    ØØØØØØ    NOP
ØØ2Ø1    ØØØØØØ    NOP
ØØ2Ø2    ØØØØØØ    NOP
```

Od adresy ØØ2ØØB nahrajeme několikrát instrukci ØØØØØØ – NOP. Při provádění instrukcí tlačítkem SINGLE CYCLE se pouze zvětšuje obsah registrů P a M.

POZOR! Dynamická zkouška se v tomto případě neprovádí.

b) *Instrukce* HLT (HaLT) – zastavení vykonávání programu. Při dynamickém provádění se počítač zastaví s indikací HLT *nn* – 1Ø2Ø*nn* v registru T. Registry P a M zobrazují adresu následující instrukce.

```
ØØ2ØØ    ØØØØØØ    NOP
ØØ2Ø1    ØØØØØØ    NOP
ØØ2Ø2    ØØØØØØ    NOP
ØØ2Ø3    1Ø2Ø77    HLT 77B
```

Po provedení instrukce na adrese ØØ2Ø3 se v registru T zobrazí hodnota 1Ø2Ø77B – což znamená bezchybné vykonání.

Dynamickou zkoušku provedeme tak, že na registru SW nastavíme adresu ØØØ2ØØ, stiskneme tlačítko LOAD ADDRESS a stiskneme tlačítko RUN. Je-li vše v pořádku, počítač se zastaví, v registru T je hodnota 1Ø2Ø77B a v registrech P a M hodnota ØØØ2Ø4.

c) *Instrukce* JMP *X* (JuMP) – skok na adresu. Po vykonání této instrukce pokračuje provádění dalších instrukcí z adresy, která je uvedena v poli operandů této instrukce. Tuto adresu sledujeme na registrech P a M.

```
ØØ2ØØ    Ø242Ø1    JMP 2Ø1B
ØØ2Ø1    ØØØØØØ    NOP
ØØ2Ø2    Ø242ØØ    JMP *-2
ØØ2Ø3    1Ø2ØØ1    HLT Ø1
```

Pokud se po provedení instrukce na adrese ØØ2Ø2 nenastaví registry P a M na adresu ØØ2ØØ, program pracuje nesprávně. Buď se neprovádí

správně instrukce JMP, nebo je na adrese ØØ2Ø2 nastavena jiná hodnota než Ø242ØØ.

Při dynamické zkoušce počítač cykluje. Pokud dojde k zastavení počítače s indikací HLT Ø1 – 1Ø2ØØ1 v registru T, program pracuje nesprávně – viz předchozí odstavec (b).

d) *Instrukce* JMP *X*, I (JuMP Indirect) – skok na adresu nepřímo. Při vykonávání této instrukce se neskočí na adresu uvedenou v instrukci, ale podle příznaku nepřímého adresování se z buněk s touto adresou vybere cílová adresa. Pokud adresa nemá příznak nepřímého adresování, je to již konečná adresa, u které se pokračuje v provádění dalších instrukcí. Pokud je to nepřímá adresa, pokračuje se stejným způsobem ve vyhledávání cílové adresy. Nepřímé adresování může pokračovat libovolně hluboko.

```
ØØ2ØØ    1242Ø1    JMP*+1, I
ØØ2Ø1    ØØØ2Ø2    OCT 2Ø2
ØØ2Ø2    1Ø2ØØ1    HLT Ø1
```

Pro vykonání instrukce z adresy ØØ2ØØ se musí tlačítko SINGLE CYCLE stisknout dvakrát. Dávejte pozor, jak se mění obsah registrů T a M a kdy se rozsvítí žárovka s nápisem INDIRECT.

e) *Instrukce* JSB *X* (Jump to SuBroutine) – skok do podprogramu. Instrukce JSB *X*, která je na adrese *Y*, způsobí, že se na adresu *X* uloží hodnota *Y* + 1, a další instrukce se provádí z adresy *X* + 1.

```
ØØ2ØØ    Ø142Ø2    JSB*+2
ØØ2Ø1    Ø242ØØ    JMP*-1
ØØ2Ø2    ØØØØØØ    NOP
ØØ2Ø3    1242Ø2    JMP*-1, I
ØØ2Ø4    1Ø2ØØ1    HLT Ø1
```

Při vykonávání instrukce JSB musíme dvakrát stisknout tlačítko SINGLE CYCLE. Všimněte si, že po vykonání této instrukce pokračuje program instrukcí z adresy ØØ2Ø3. Jestliže si zobrazíme obsah buňky ØØ2Ø2, zjistíme, že je na ní uložena hodnota ØØ2Ø1. Vlastní podprogram začíná na adrese ØØ2Ø2 instrukcí NOP a končí na adrese ØØ2Ø3 instrukcí JMP *X*, I, kde *X* je první adresa podprogramu, na níž je uložena adresa návratu z podprogramu (tj. ØØØ2Ø1). V tomto případě tedy vlastní podprogram neobsahuje žádnou pracovní instrukci (obsahuje jedinou instrukci – JMP*–1, I). Při dynamické zkoušce počítač cykluje.



f) *Instrukce* LIA *SC* (Load Input into A) – vložení osahu vyrovnávacího registru vstupního zařízení do registru A. V instrukci LIA udává *SC* adresu zařízení (SourCe = zdroj). Je-li tato hodnota rovna jedné, ukládá se do registru A obsah registru SW.

```
ØØ2ØØ    1Ø25Ø1    LIA Ø1
ØØ2Ø1    Ø242ØØ    JMP*-1
ØØ2Ø2    1Ø2ØØ1    HLT Ø1
```

Do registru SW nastavíme libovolnou kombinaci nul a jedniček. Po provedení instrukce LIA Ø1 z adresy ØØ2ØØ musí obsah registru A souhlasit s obsahem registru SW. Při dynamické zkoušce počítač cykluje. Měníme-li za chodu obsah registru SW, mění se ihned i obsah registru A.

g) *Instrukce* MIA *SC* (Merge Input into A) – logický součet obsahu vyrovnávacího registru vstupního zařízení a obsahu registru A.

```
ØØ2ØØ    1Ø25Ø1    LIA Ø1
ØØ2Ø1    1Ø24Ø1    MIA Ø1
ØØ2Ø2    Ø242ØØ    JMP*-2
ØØ2Ø3    1Ø2ØØ1    HLT Ø1
```

Postup je stejný jako u instrukce LIA. Registr A ale obsahuje logický součet původního obsahu registru A a obsahu registru SW. Pokud obsah registru SW neměníme, je obsah registru A po instrukci MIA shodný s obsahem registru SW.

h) *Instrukce* LIB *SC* (Load Input into B) – vložení obsahu vyrovnávacího registru vstupního zařízení do registru B.

```
ØØ2ØØ    1Ø65Ø1    LIB Ø1
ØØ2Ø1    Ø242ØØ    JMP*-1
ØØ2Ø2    1Ø2ØØ1    HLT Ø1
```

Platí totéž jako při instrukci LIA, operace se však provádí s registrem.

i) *Instrukce* STA *X* (STore A) – uložení obsahu registru A do paměti. Při provádění této instrukce se obsah registru A uloží na tu adresu paměti, která je v instrukci STA *X* (tj. *X*). Pokud jde o přímé uložení, je na vykonání této instrukce třeba dvakrát stisknout tlačítko SINGLE CYCLE, při nepřímém adresování se musí toto tlačítko stisknout vícekrát.

```
ØØ2ØØ    1Ø25Ø1    LIA Ø1
ØØ2Ø1    Ø7Ø2Ø4    STA *+3
ØØ2Ø2    Ø242ØØ    JMP *-2
```

```
ØØ2Ø3   1Ø2ØØ1   HLT Ø1
ØØ2Ø4   ØØØØØØ   NOP
```

Do registru SW vložíme kontrolní vzorek. Po vykonání instrukce LIA Ø1 se tato hodnota přenese do registru A. Při vykonávání instrukce STA *+3 se tato hodnota přenese do buňky ØØØ2Ø4. Po provedení instrukce JMP 2ØØB se přesvědčíme, zda se na adrese ØØØ2Ø4 skutečně nachází kontrolní vzorek.

Při dynamické zkoušce počítač cykluje. Stiskneme tlačítko HALT – počítač se zastaví. Tiskneme tlačítko DISPLAY MEMORY tak dlouho, až se zobrazí obsah buňky ØØØ2Ø4 – tj. v registrech P a M se indikuje adresa ØØØ2Ø5. Obsah registru T musí souhlasit s obsahem registru SW. Další spuštění programu provedeme od adresy ØØØ2ØØB.

j) *Instrukce* LDB *X* (Load into B) – vložení do registru B. Po vykonání této instrukce se do registru B uloží obsah buňky, jejíž adresa je v instrukci LDB *X*.

```
ØØ2ØØ   1Ø25Ø1   LIA Ø1
ØØ2Ø1   Ø7Ø2Ø5   STA *+4
ØØ2Ø2   Ø642Ø5   LDB *+3
ØØ2Ø3   Ø242ØØ   JMP *-3
ØØ2Ø4   1Ø2ØØ1   HLT Ø1
ØØ2Ø5   ØØØØØØ   NOP
```

Do registru SW vložíme kontrolní vzorek. Ten se instrukcí LIA Ø1 přenese do registru A, instrukcí STA *+4 do buňky ØØØ2Ø5 a instrukcí LDB *+3 do registru B. Pro proběhnutí celého cyklu můžeme zkontrolovat obsah buňky ØØ2Ø5, zda souhlasí s obsahem registru SW.

Při dynamické zkoušce počítač cykluje. Obsah registru A a registru B musí souhlasit s obsahem registru SW. Měníme-li za chodu obsah registru SW, mění se ihned i obsah registrů A a B.

k) *Instrukce* STA *X*, I (STore A, Indirect) – uložení obsahu registru A na nepřímou adresu. Při provádění této instrukce se obsah registru A neukládá přímo na adresu *X*, ale za cílovou adresu se bere obsah buňky s adresou *X*. Pokud tento obsah představuje přímou adresu, ukládá se obsah registru na tuto adresu, pokud obsah buňky *X* představuje nepřímou adresu, pokračuje výběr adresy dále. Pokud je adresa *X* rovna hodnotě ØØØØØ1, jde o registr B.

```
ØØ2ØØ   1Ø25Ø1   LIA Ø1
```



```
ØØ2Ø1   Ø642Ø6   LDB *+5
ØØ2Ø2   17ØØØ1   STA Ø1, I
ØØ2Ø3   16ØØØ1   LDA Ø1, I
ØØ2Ø4   Ø242ØØ   JMP *-4
ØØ2Ø5   1Ø2ØØ1   HLT Ø1
ØØ2Ø6   ØØØ2Ø7   OCT 2Ø7
ØØ2Ø7   ØØØØØØ   NOP
```

Do registru SW nastavíme kontrolní vzorek, který se přenese do registru A. Do registru B se přenese hodnota ØØØ2Ø7. Instrukcí STA Ø1, I se obsah registru A přenese do buňky ØØØ2Ø7. Instrukcí LDA Ø1, I se obsah z buňky ØØØ2Ø7 přenese opět do registru A. Jestliže po vykonání instrukce STA Ø1, I vynulujeme registr A, obnoví se po vykonání instrukce LDA Ø1, I opět obsah registru A. Na adrese ØØØ2Ø7 by měl být kontrolní vzorek.

Při dynamické zkoušce počítač cykluje.

l) *Instrukce* CPB *X* (ComPare to B) – porovnání obsahu registru B, přeskok při nerovnosti. Při provádění této instrukce se porovnává obsah registru B s obsahem buňky, jejíž adresa je uvedena v instrukci CPB *X*. Je-li obsah shodný, provádí se následující instrukce, je-li rozdílný, následující instrukce se přeskočí a provádí se až další instrukce.

```
ØØ2ØØ   Ø642Ø6   LDB *+6
ØØ2Ø1   1Ø25Ø1   LIA Ø1
ØØ2Ø2   Ø54ØØØ   CPB *+4
ØØ2Ø3   1Ø2Ø77   HLT 77B
ØØ2Ø4   Ø242ØØ   JMP *-4
ØØ2Ø5   1Ø2ØØ1   HLT Ø1
ØØ2Ø6   Ø52525   OCT 52525
```

Na registru SW nastavíme kontrolní vzorek a uložíme ho na adresu ØØ2Ø6. Po provedení instrukce CPB buňky ØØØ2Ø2 se bude dále provádět instrukce JMP 2ØØB z buňky ØØØ2Ø4, pokud kontrolní vzorek bude jiný než Ø52525. Pokud bude kontrolní vzorek Ø52525, provede se po instrukci CPB instrukce HLT 77B, tj. počítač se zastaví. Tato činnost bude lépe vidět při dynamické zkoušce.

m) *Instrukce* CPA *X*, I (ComPare to A, Indirect) – porovnání s obsahem registru A nepřímo, skok při nerovnosti. Tato instrukce se provádí podobně jako instrukce CPB, bere se však obsah registru A, tj. CPA. Obsah adresy *X* se neporovnává, ale bere se jako nová adresa – opět buď jako přímá, nebo nepřímá podle příznaku nepřímého adresování.

```
ØØ2ØØ   Ø642Ø6   LDB *+6
ØØ2Ø1   Ø6Ø2Ø4   LDA *+4
ØØ2Ø2   15ØØØ1   CPA 1,I
ØØ2Ø3   1Ø2Ø77   HLT 77B
ØØ2Ø4   Ø242ØØ   JMP *-4
ØØ2Ø5   Ø52525   OCT 52525
ØØ2Ø6   ØØØ2Ø5   OCT 2Ø5
```

Do registru B se přesune obsah buňky ØØØ2Ø6, tj. hodnota ØØØ2Ø5. Je to vlastně adresa, kde se nachází kontrolní vzorek. Dále se do registru A nahraje kontrolní vzorek. Ten se porovná s obsahem buňky, jejíž adresa je v registru B. V našem případě jsou obě adresy stejné – ØØ2Ø5. Proto se počítač zastaví s indikací HLT 77B.

Při stisknutí tlačítka RUN se počítač zastaví s indikací HLT 77B – 1Ø2Ø77 – v registru T. V registrech P a M je adresa ØØØ2Ø4. Po stisknutí tlačítka RUN se tato činnost opět opakuje.

n) *Instrukce* INB (INcrement B) – zvětšení obsahu registru B o jednu. Při vykonání této instrukce se obsah registru B zvětší o jednu.

```
ØØ2ØØ   1Ø65Ø1   LIB Ø1
ØØ2Ø1   ØØ6ØØ4   INB
ØØ2Ø2   Ø242ØØ   JMP *-2
ØØ2Ø3   1Ø2ØØ1   HLT Ø1
```

Na registru SW nastavíme kontrolní vzorek. Po provedení instrukce INB z buňky ØØØ2Ø1 se obsah registru B zvětší o jednu, a je tudíž o jednu větší než kontrolní vzorek.

Při dynamické zkoušce se kontrolní vzorek v registru B průběžně zvětšuje o hodnotu 1.

o) *Instrukce* ALF (rotate A Left Four) – rotace registru A vlevo o 4 bity. Při vykonání této instrukce se obsah registru A posune vlevo o čtyři místa. Použijí-li se dvě instrukce ALF za sebou, dojde k posunu vlevo o 8 míst. V obou případech obsah nejvyššího (tj. 15.) bitu přechází na pozici nultého bitu.

```
ØØ2ØØ   1Ø25Ø1   LIA Ø1
ØØ2Ø1   ØØ17ØØ   ALF
ØØ2Ø2   ØØ1727   ALF, ALF
ØØ2Ø3   Ø242Ø1   JMP *-2
ØØ2Ø4   1Ø2ØØ1   HLT Ø1
```



Na registru SW nastavíme kontrolní vzorek, který se přenese do registru A. Instrukcí ALF se tato hodnota posune o 4 místa vlevo, dvěma instrukcemi ALF se tato hodnota posune o 8 míst vlevo. Při dynamické zkoušce se obsah registru A trvale posunuje.

p) *Instrukce* CLE (Clear E) – nulování registru E. Při vykonání této instrukce se vynuluje registr E.

```
ØØ2ØØ   Ø642Ø5   LDB *+5
ØØ2Ø1   ØØ6ØØ4   INB
ØØ2Ø2   1Ø2Ø77   HLT 77B
ØØ2Ø3   ØØØØ4Ø   CLE
ØØ2Ø4   Ø242ØØ   JMP *-4
ØØ2Ø5   177777   OCT 177777
```

Instrukcí LDB *+5 nahrajeme do registru B hodnotu 177777 (oktalově) – tj. samé jedničky. Další instrukcí (INB) zvětšíme obsah registru B o jednu, v registru B bude tedy hodnota ØØØØØØ a zároveň se registr E nastaví na hodnotu 1 – na ovládacím panelu počítače se rozsvítí signalizace indikující obsah registru E. Nyní stiskneme tlačítko SINGLE CYCLE a registr E se musí vynulovat, tj. signalizace zhasne. Stiskneme-li opět tlačítko RUN, celý děj se opakuje.

r) *Instrukce* CLO (Clear O) – nulování registru O. Při vykonání této instrukce se vynuluje registr O. Dynamická zkouška je obdobná.

Úkoly:

1. Z ovládacího panelu nahrajte do počítače ADT příslušný program.
2. Pomocí tlačítka SINGLE CYCLE prověřte správnou funkci programu.
3. Kontrolujte obsah registrů T, P a M, popř. registrů A nebo B.
4. Zjistíte-li chybnou funkci programu, najděte chybu.
5. Po bezchybné zkoušce podle bodu 2 proveďte dynamickou zkoušku.

## 5.2. PROGRAMOVÁ OBSLUHA VÝSTUPU DAT NA PERIFERNÍM ZAŘÍZENÍ

### Úvod

V tomto cvičení se budeme zabývat ovládáním vstupních a výstupních periferních zařízení na počítači ADT 4000. Půjde o přenos z počítače do periferních zařízení, z periferních zařízení do počítače, a to jednak bez použití přerušovacího systému v normálním nebo zrychleném režimu, jednak s přerušovacím systémem.

### Příprava cvičení

Rozhraní počítače s periferním zařízením má za úkol přenášet data z periferních zařízení do počítače, kde se zpracovávají, a po zpracování naopak přenášet data z počítače na periferní zařízení tak, aby byly výsledky zobrazeny vhodným způsobem. Základní periferní zařízení počítače ADT 4000 jsou snímač děrné pásky, děrovač děrné pásky, psací stroj, dále rychlotiskárna, displej, magnetická disková paměť. Další periferní zařízení jsou časový generátor, číslicový voltmetr, převodník číslo–napětí (D/A), přepínač měřicích míst aj. Připojení všech těchto periferních zařízení je založeno na jednotném principu. Musí být přitom splněny určité technické předpoklady, podle toho je připojení realizováno vhodným programem. Pro základní periferní zařízení jsou programy pro připojení periferních zařízení dodávány výrobcem počítače, a to i pro některá nestandardní periferní zařízení, pro ostatní periferní zařízení si musí uživatel vytvořit ovládací programy sám, a to konkrétně podle vlastních požadavků. Protože všechna periferní zařízení se připojují standardním způsobem, může uživatel připojit – za dodržení určitých podmínek – k počítači libovolné zařízení, a to ať jde o zařízení dodávající data do počítače nebo o zařízení přejímající z počítače určité informace. Periferní zařízení musí být schopno předat (případně převzít) počítači vícebitové slovo ve formě elektrických signálů. Dále je nutné, aby toto periferní zařízení během své činnosti informovalo počítač o stavu, ve kterém se nachází, signálem F (Flag). Naopak počítač aktivuje periferní zařízení signálem C (Control). Pokud je periferní zařízení schopno přenášet data, vysílá signál F s úrovní logické hodnoty 1. Při aktivaci periferního zařízení se signál F změní na úroveň logické hodnoty 0. Teprve když toto zařízení ukončí svoji činnost, tj. vstupní zařízení



přeneslo data do počítače a výstupní zařízení převzalo data z počítače, změní opět signál F na úroveň logické hodnoty 1, a tím oznámí počítači, že je schopno přenášet další data. Typickým příkladem vstupního periferního zařízení je děrovač děrné pásky. Psací stroj je vstupní i výstupní zařízení. Musí se přitom dodatečným signálem určit, zda jde o přenos do počítače, tj. zda počítač čeká na to, co člověk napíše na psacím stroji, nebo zda počítač přímo ovládá výpis na psacím stroji.

Spolupráce snímače děrné pásky s počítačem je následující: V určitém místě programu je nutné sejmout z děrné pásky vyděrovaný znak. Počítač nastaví signál C = 1. Tímto signálem se spustí mechanika snímače pro posun pásky. Zároveň s přijetím signálu C = 1 změní snímač signál F z logické hodnoty 1 na logickou hodnotu 0, a tím oznámí počítači, že je snímač v činnosti. Než se děrná páska posune o jeden znak, uplyne u fotoelektrického snímače doba asi 1 ms. Po tuto dobu počítač nemá k dispozici data a musí čekat na ukončení práce snímače. Po posunu děrné pásky pod fotodiody a po ustálení signálů na propojovací vedení do počítače vydá snímač signál F = 1, a tím oznámí počítači, že ukončil sejmutí jednoho znaku z děrné pásky a je připraven k další činnosti. Tímto signálem se zároveň překlopí klopné obvody na rozhraní, a tak se přenesou hodnoty přečtené snímačem děrné pásky do počítače. Počítač zjistí, že signál F = 1, převezme hodnoty z desky rozhraní a pokračuje v plnění dalšího programu.

Připojení děrovače děrné pásky k počítači je obdobné. V určitém místě programu má počítač předat děrovači znak k vyděrování do děrné pásky. Počítač přenese tuto hodnotu ve formě osmibitového slova na klopné obvody na desce rozhraní. Tím se tato informace po spojovacím vedení dostane až na vstupní obvody děrovače. Potom počítač změní signál C z hodnoty 0 na hodnotu 1, tak vlastně spustí činnost děrovače. Děrovač okamžitě změní signál F z hodnoty 1 na 0, a tím oznámí, že děrovač je v činnosti. Zároveň se spustí mechanika a vyděruje se jeden znak do děrné pásky. Po celou dobu děrování znaku je signál F = 0. Na rozdíl od vstupu dat do počítače, kdy počítač musí čekat na ukončení činnosti snímače děrné pásky, při výstupu dat z počítače může počítač buď čekat na ukončení činnosti děrovače a potom pokračovat v další činnosti, např. předání dalších dat do děrovače a opětovné spuštění děrovače, nebo nemusí čekat na dokončení činnosti děrovače a pokračovat v dalším vykonávání programu. V tomto případě bude ale spolupráce počítače s děrovačem poněkud pozměněna. Přitom se využívá přerušovací systém.

V tomto případě počítač spustí periferní zařízení a pokračuje v další

činnosti. Jakmile zařízení dokončí započatou činnost, přihlásí se signálem F = 1. Pokud jsou splněny i další nutné podmínky, dojde k přerušení právě prováděného programu a provede se instrukce, která je na adrese určené hodnotou výběrového kódu desky rozhraní aktivovaného periferního zařízení. Tato instrukce může být jednoduchá, např. NOP, HLT, CLC, nebo jde o instrukci skoku do určité části programu (JMP), ale většinou se používá instrukce skoku do podprogramu (JSB). Tímto způsobem máme zajištěno, že po skončení podprogramu, který obsluhuje přerušení, se může původní program správně dokončit. Je zde ještě nutné zajistit, aby po přerušení byl obsah registrů A, B, E, O stejný jako před skokem do podprogramu. Podprogram musí nejdříve zajistit úklid obsahů registrů do paměti a před ukončením podprogramu opět obnovit původní obsah registrů. Při návratu z podprogramu se obnoví adresa, na níž došlo k přerušení hlavního programu, a protože i obsah registrů je stejný jako v okamžiku přerušení, vykonávání programu může pokračovat.

Při vstupu dat do počítače se výhody přerušovacího systému tolik neprojeví, protože počítač musí čekat na vstup dat a nemůže většinou pokračovat v dalším výpočtu. Při výstupu dat z počítače je situace jiná, počítač může výstupní hodnoty uchovat v paměti, do výstupního zařízení předávat výsledky po jednotlivých znacích, a než zařízení provede výstupní operaci, může počítač provádět další část programu. V tomto případě se ušetří čas, protože počítač může obsluhovat i několik zařízení „najednou", např. děrovač i tiskárnu. Počítač předá data na desku rozhraní a spustí děrovač. Potom provede totéž pro tiskárnu. A do té doby, než se některá zařízení přihlásí signálem F = 1, se počítač věnuje provádění další části programu. V procesoru se v každém časovém okamžiku zpracovává pouze jeden program, ale přitom jsou rozpracovány tři různé programy – jeden pro děrování na děrovači, druhý pro výstup na tiskárnu a třetí je vlastní program. Tento způsob práce – *sdílení času procesoru* – je možný pouze s přerušovacím systémem.

Velice důležitý je přerušovací systém při řízení výroby, kdy je v počítači rozpracováno několik programů a podle požadavků z výroby nebo časového generátoru se spouští určitý program. Programy, stejně jako signály přerušení, mají svou prioritu. Žádají-li např. dvě zařízení o přerušení, zpracuje se ten požadavek, který má vyšší prioritu, a tedy větší důležitost. Teprve když se vykoná program pro přerušení s vyšší prioritou, může se provést program pro přerušení s nižší prioritou. A naopak, provádíme-li program pro přerušení s nižší prioritou a přijde přerušení s vyšší prioritou,



přestane se provádět předcházející program a začne se provádět program pro obsluhu přerušení s vyšší prioritou. Priorita je dána pořadovým číslem (výběrovým kódem) rozhraní desky. Platí – čím menší je číslo desky, tím větší je priorita.

## Zadání

Podle dalšího výkladu zaveďte do počítače program pro spojení počítače s příslušným periferním zařízením. Program upravte podle konkrétního připojení periferního zařízení k počítači. U programů, které nevyužívají přerušovací systém, odzkoušejte pomocí tlačítka SINGLE CYCLE správnou funkci programu. Pokud bude program správně pracovat, proveďte zkoušku v režimu RUN, tj. spuštění programu od startovací adresy. Pokud při provádění programu zjistíte chybu, zkuste ji samostatně opravit.

## Popis cvičení

Zadávejte do počítače posloupnost instrukcí pro ovládání jednotlivých periferních zařízení, postupně od jednodušších případů ke složitějším. Pokud některému programu plně neporozumíte, zkuste se nad ním trochu zamyslet a nepřecházet k dalšímu programu, dokud neporozumíte programu předcházejícímu. Postup vložení programu do počítače byl vysvětlen v předcházejícím cvičení.

a) *Přenos dat ze snímače děrné pásky do počítače bez využití přerušovacího systému*

Jde o nejjednodušší způsob přenosu dat z periferního zařízení do počítače. Pokud vstupní zařízení nemá výběrový kód 10B, je nutné změnit tuto hodnotu u instrukcí STC, SFS a LIA.

```
00100    103710    STC 10B, C
00101    102310    SFS 10B
00102    024101    JMP *-1
00103    102510    LIA 10B
00104    024100    JMP 100B
```

Od adresy 000100 nahrajeme program do počítače. Do snímače děrné pásky dáme větší cívku děrné pásky s libovolným obsahem, např. starými

výsledky, nebo jinou nepotřebnou pásku. Teprve potom začneme se zkoušením programu pomocí tlačítka SINGLE CYCLE.

Po provedení instrukce STC 1ØB,C se spustí snímač děrné pásy, tj. děrná páska ve snímači se přesune o jeden znak. Touto instrukcí se zároveň vynuluje signál F jdoucí do počítače. Po ukončení čtení znaku snímač nastaví signál F = 1.

Při ručním prověřování instrukce SFS 1ØB již bude signál F roven jedné, proto dojde k přeskočení následující instrukce. Pokud budeme v další části cvičení provádět zkoušku v režimu RUN, bude nejprve F=Ø a počítač přejde na následující instrukci. Tou je instrukce JMP * − 1, tj. skok na instrukci SFS 1ØB. Tyto dvě instrukce bude počítač provádět tak dlouho, dokud nedojde ke změně signálu F na jedničku, tj. dokud snímač neukončí přenos jednoho znaku.

Při ručním prověřování se instrukce JMP * − 1 přeskočí a přejde se k provádění instrukce LIA 1ØB. Tou se přenese obsah vyrovnávací paměti na rozhraní do registru A. Porovnáním této hodnoty s hodnotou znaku, který byl právě přečten ve snímači, se přesvědčíte o správném přenosu znaku do počítače.

Další instrukce je JMP 1ØØB, tj. skok na začátek programu. Zkuste stlačit tlačítko SINGLE CYCLE vícekrát a pak je tiskněte tak rychle, jak jen budete moci. Děrná páska ve snímači se bude postupně posunovat vždy o jeden znak.

Nyní provedeme dynamickou zkoušku. Zkontrolujte, zda pod snímačem děrné pásky je dostatečné množství děrné pásky a zda se páska volně odvíjí. Potom nastavte adresu ØØØ1ØØ, stiskněte tlačítko RESET a tlačítko RUN. Snímač začne snímat znaky z děrné pásky, a to poměrně velkou rychlostí. Program není ukončen, proto činnost počítače zastavíme stisknutím tlačítka HALT. Snímání ukončete včas, nenechte vyběhnout pásku ze snímače. Čtení je možné zpomalit tím, že pásku přidržíte v ruce, nikdy ale pásku nezastavujte úplně, mohlo by dojít k poškození snímače.

*Upozornění:* Pokud vyběhla páska ze snímače, ihned nadzvedněte snímací hlavu a pak zastavte počítač tlačítkem HALT. Snímač je stále v činnosti. Vložte do snímače kousek pásky s vodicími otvory a uzavřete raménko. Páska se přesune o jeden znak, snímač ukončí snímání.



b) *Přenos dat z počítače do děrovače děrné pásky bez využití přerušovacího systému*

Opět jde o nejjednodušší přenos dat z počítače do výstupního zařízení, kdy počítač čeká na dokončení výstupní operace. Pokud výstupní zařízení nemá výběrový kód 11B, je nutné opravit tuto hodnotu u instrukcí OTA, STC, SFS.

```
ØØ1ØØ   1Ø2611   OTA 11B
ØØ1Ø1   1Ø3711   STC 11B,C
ØØ1Ø2   1Ø2311   SFS 11B
ØØ1Ø3   Ø241Ø2   JMP *−1
ØØ1Ø4   Ø241ØØ   JMP 1ØØB
```

Program vložíme do počítače od adresy ØØØ1ØØ. Zkontrolujeme, zda v děrovači děrné pásky je dostatečné množství děrné pásky (stačí i menší zbytek děrné pásky).

Nejprve do registru A vložíme libovolný vzorek, který chceme vyděrovat. Instrukcí OTA 11B se tato hodnota přenese do klopných obvodů na desce rozhraní a odtud na vstup děrovače.

Po provedení instrukce STC 11B,C se spustí děrování a zároveň se vynuluje signál F v počítači. Jestliže děrovač byl předtím v klidu, trvá několik sekund, než se rozeběhne mechanika děrovače a než děrovač vyděruje daný znak. O tom se přesvědčíme tím, že posuneme pásku děrovače a zkontrolujeme vyděrované místo.

Další instrukce je SFS 11B. Pokud stiskneme tlačítko SINGLE CYCLE dostatečně rychle, dokud ještě děrovač neskončil svoji činnost, bude signál F=Ø a bude se provádět následující instrukce. Tou je JMP*−1, tj. skok na adresu ØØØ1Ø2. Tam je instrukce SFS 11B a tyto dvě instrukce se budou opakovat tak dlouho, dokud nedojde k vyděrování znaku na děrovači a ke změně signálu na F=1. Pak dojde k přeskočení následující instrukce a bude se provádět instrukce JMP 1ØØB. Tím se opět dostaneme na začátek programu.

Pokud budeme tisknout tlačítko SINGLE CYCLE dostatečně rychle, budeme muset při prvním spuštění děrovače při instrukci SFS 11B tisknout tlačítko několikrát, než se znak vyděruje.

Po každém vyděrování můžeme změnit obsah registru A, a tím můžeme vyděrovat libovolnou kombinaci znaků.

Pokud program pracuje bez závad, provedeme dynamickou zkoušku.

Zkontrolujeme, zda v děrovači je dostatečné množství pásky, nastavíme adresu ØØØ1ØØ, stiskneme tlačítko PRESET a RUN. Děrovač se rozběhne a bude děrovat znak podle posledních osmi bitů obsažených v registru A. Děrování zastavíme stisknutím tlačítka HALT. Změníme obsah registru A a opět stiskneme tlačítko RUN. Děrovač bude děrovat jiný znak. Porovnáním s registrem A potvrdíme správnou funkci.

c) *Přenos dat ze snímače děrné pásky do počítače bez využití přerušovacího systému zrychleným způsobem*

```
ØØ1ØØ    1Ø371Ø    STC 1ØB,C
ØØ1Ø1    ØØØØØØ    NOP
ØØ1Ø2    ØØØØØØ    NOP
ØØ1Ø3    1Ø231Ø    SFS 1ØB
ØØ1Ø4    Ø241Ø3    JMP *-1
ØØ1Ø5    1Ø251Ø    LIA 1ØB
ØØ1Ø6    1Ø371Ø    STC 1ØB,C
ØØ1Ø7    ØØØØØØ    NOP
ØØ11Ø    ØØØØØØ    NOP
ØØ111    Ø241Ø3    JMP 1Ø3B
```

Proti prvnímu příkladu je zde jistě změna. Program začíná instrukcí STC 1ØB,C, tj. nastartováním snímače. Protože snímač sejme hodnotu až po určité chvíli, může nyní počítač uskutečňovat některé přípravné operace nutné pro přenos dat. Je to naznačeno tím, že na buňkách ØØØ1Ø1 a ØØØ1Ø2 jsou instrukce NOP, ale místo nich můžeme dát libovolný počet instrukcí, které je nutné provést. Teprve potom se vykonává instrukce SFS 1ØB, tj. zjištění, zda snímač dokončil sejmutí jednoho znaku. Pokud snímač tuto činnost nedokončil, provede se následující instrukce JMP *-1 a počítač čeká na dokončení práce snímače. Zatímco snímač pracoval, počítač využil část této doby ke zpracování části programu.

Po nastavení signálu F=1 snímačem provede počítač instrukci LIA 1ØB, tj. přenese hodnotu sejmutého znaku do registru A a provede instrukci STC 1ØB,C, tj. opět okamžitě nastartuje snímač. V původním řešení by nyní počítač opět čekal na dokončení práce snímače, v tomto řešení může zpracovávat načtené hodnoty, např. ukládat je do určitého místa paměti apod. Je to naznačeno tím, že na adresách ØØØ1Ø7 až ØØØ11Ø jsou instrukce NOP, ale může tam být opět libovolný počet instrukcí.

Po dokončení těchto operací se může testovat, je-li již dokončen vstup



dat ze snímače do počítače. Pokud chceme ještě něco načíst, provedeme instrukci JMP 1Ø3, popř. skok na instrukci SFS 1ØB a čekáme na dokončení práce snímače. Pokud zjistíme, že přenos dat ze snímače je již dokončen, pokračujeme v dalším provádění programu.

Jestliže se někde dále v programu opět vyskytne potřeba snímat hodnoty ze snímače, můžeme vyjít z toho, že první hodnotu z tohoto nového souboru snímač už vlastně přečetl. Provedeme tedy pouze instrukci SFS 1ØB, tj. přesvědčíme se, zda snímač skutečně dokončil předcházející činnost. Pokud ano, provedeme instrukci LIA 1ØB a STC 1ØB,C, tj. přeneseme přečtenou hodnotu do registru A a opět nastartujeme snímač. Pokud by se stalo, že další požadavek na snímač přišel v okamžiku, kdy snímač ještě nedokončil předcházející čtení, tj. signál F=Ø, je nutné čekat na dokončení práce snímače. V každém případě je alespoň část doby, po kterou se dříve čekalo na snímač, využita k jiným účelům, a tedy celková doba činnosti počítače se zkrátí. Zároveň je také vidět, že snímač je při snímání téměř stoprocentně využit, tj. okamžitě po skončení sejmutí jednoho znaku je znovu nastartován.

Rozdíl mezi snímáním dat ze snímače podle předcházejícího programu a tohoto programu je velmi malý, protože se neprovádějí žádné mezioperace s načtenými hodnotami. Pokud ale v počítači používáme základní zaváděcí program (BBL) buď s normálním čtením, nebo s rychlým čtením, je vidět zřetelný rozdíl mezi prací snímače v normálním nebo v rychlém režimu. V normálním režimu páska probíhá snímačem sice velmi rychle, ale trochu trhaně, kdežto v rychlém režimu probíhá páska snímačem naprosto hladce.

Při rychlém čtení si musíme uvědomit, že při zastavení snímače máme vlastně již jeden znak přečtený dopředu, aniž si počítač tuto hodnotu převzal k dalšímu zpracování. Chceme-li program spustit znovu od začátku, ale data chceme zpracovávat v tom místě, kde skončilo předcházející zpracování, musíme pásku ve snímači posunout o jeden znak zpět. Jiný případ je ten, kdy se vyskytne chyba v datech. Potom je chyba nikoli v tom znaku, který je pod snímačem, ale ve znaku předcházejícím.

d) *Přenos dat ze snímače děrné pásky do počítače bez využití přerušovacího systému zrychleným způsobem (s využitím skoku do podprogramu)*

```
ØØ1ØØ    1Ø371Ø    STC 1ØB,C
ØØ1Ø1    ØØØØØØ    NOP
```

```
ØØ1Ø2    ØØØØØØ    NOP
ØØ1Ø3    Ø1611Ø    JSB 11ØB
ØØ1Ø4    ØØØØØØ    NOP
ØØ1Ø5    ØØØØØØ    NOP
ØØ1Ø6    ØØØØØØ    NOP
ØØ1Ø7    Ø241Ø3    JMP 1Ø3B
ØØ11Ø    ØØØØØØ    NOP
ØØ111    1Ø231Ø    SFS 1ØB
ØØ112    Ø24111    JMP *-1
ØØ113    1Ø251Ø    LIA 1ØB
ØØ114    1Ø371Ø    STC 1ØB,C
ØØ115    12411Ø    JMP 11ØB,I
```

Princip činnosti je stejný jako u případu c). Podprogram pro přenos dat ze snímače je uložen na adresách ØØØ11Ø až ØØØ115. Na adrese ØØØ11Ø je instrukce NOP, protože sem se při instrukci JSB nahraje adresa návratu. Program je zakončen instrukcí JMP 11ØB,I, tj. návrat do hlavního programu nepřímo.

e) *Přenos dat z počítače do děrovače děrné pásky bez využití přerušovacího systému zrychleným způsobem*

Způsob práce je podobný jako při rychlém čtení. Vlastní provedení je ale jednodušší z toho důvodu, že zatímco při vstupu musíme čekat na dokončení vstupní operace a pak teprve můžeme pokračovat v dalším provádění programu, při výstupu dat z počítače nemusí počítač čekat na dokončení činnosti děrovače a může pokračovat v provádění dalšího programu. Před každým výstupem na děrovač musí počítač zjistit, zda je děrovač schopen děrovat danou hodnotu nebo zda je ještě v činnosti. Zjistí-li počítač, že je děrovač ještě v činnosti, tj. signál F=Ø, přestává počítač zpracovávat další program a čeká na vyděrování předcházející hodnoty a pak teprve pokračuje v dalším programu.

```
ØØ1ØØ    1Ø2311    SFS 11B
ØØ1Ø1    Ø241ØØ    JMP *-1
ØØ1Ø2    1Ø2611    OTA 11B
ØØ1Ø3    1Ø3711    STC 11B,C
ØØ1Ø4    ØØØØØØ    NOP
ØØ1Ø5    ØØØØØØ    NOP
ØØ1Ø6    Ø241ØØ    JMP 1ØØB
```



Jako první instrukci použijeme instrukci SFS 11B, tj. dotaz, zda děrovač ukončil předcházející činnost. Pokud je děrovač ještě v činnosti, musíme počkat na signál F=1, k tomu slouží instrukce JMP *-1.

Pokud děrovač ukončil předcházející činnost, provede se instrukce OTA 11B, tj. přenos hodnoty z registru A na desku rozhraní a odtud na vstup děrovače. Poté se provede instrukce STC 11B,C, tj. spustí se děrovač. Touto instrukcí končí spolupráce počítače s děrovačem a počítač se může věnovat provádění další části programu.

f) *Přenos dat ze snímače do počítače s využitím přerušovacího systému*

Při přenosu dat s využitím přerušovacího systému nejprve uvedeme v činnost periferní zařízení a pak provádíme další program. Po ukončení činnosti se periferní zařízení přihlásí signálem F=1 a oznámí, že je volné k další činnosti. V tom případě se přeruší právě prováděný program a následuje skok do podprogramu, který opět uvede v činnost periferní zařízení.

Aby mohlo dojít k přerušení, musí být splněny tyto podmínky:

a) Musí být odblokován přerušovací systém – stiskneme tlačítko INT ENABLE.

b) Dané periferní zařízení musí být odstartováno, tj. C=1.

c) Musí být nahlášen konec operace, tj. F=1.

d) Musí být splněny ještě další podmínky, jako ukončení předcházející operace, po operacích STC, CLC, STF či CLF se musí provádět ještě jedna další fáze atd.

Jestliže jsou tyto podmínky splněny, vloží se do právě prováděného programu instrukce, která je na adrese přerušení – v tomto případě na adrese ØØØØ1Ø.

```
ØØØ1Ø    Ø1411Ø    JSB 11ØB
ØØ1ØØ    1Ø371Ø    STC 1ØB,C
ØØ1Ø1    Ø241Ø1    JMP*
ØØ11Ø    ØØØØØØ    NOP
ØØ111    1Ø251Ø    LIA 1ØB
ØØ112    1Ø371Ø    STC 1ØB,C
ØØ113    12411Ø    JMP 11ØB,I
```

Na adresu ØØØØ1Ø nahrajeme instrukci JSB 11ØB – tj. skok do podprogramu pro obsluhu přerušení. Od adresy ØØØ11Ø napíšeme tento program. Na adresu ØØØ11Ø vložíme instrukci NOP – zde se bude nacházet adresa návratu. Dále provedeme instrukci LIA 1ØB, tj. přenos hodnoty do registru

A, pak STC 1ØB,C, tj. opětovné nastartování snímače. Pokud nebudeme dále zpracovávat načtené hodnoty, ukončíme podprogram instrukcí JMP 11ØB,I – tj. standardní návrat z podprogramu.

Od adresy ØØØ1ØØ nahrajeme hlavní program. Nejprve nastartujeme snímač instrukcí STC 1ØB,C. Potom bychom mohli psát další instrukce, které bychom chtěli provádět. V tomto případě zde použijeme tzv. *dynamický stop* – instrukci JMP*. Při spuštění tlačítkem RUN se tato instrukce stále opakuje. Na první pohled se zdá, že počítač je v režimu HALT, protože všechny indikační žárovky svítí bez blikání, pouze žárovka RUN ukazuje, že počítač je v režimu RUN. Stiskneme-li tlačítko HALT a tiskneme tlačítko SINGLE CYCLE, zdánlivě se nic neděje, opakuje se stále stejná instrukce.

Tento program nemůžeme spouštět pomocí tlačítka SINGLE CYCLE, neboť bychom se dostali na instrukci JMP* a provádění dalších instrukcí by nebylo možné. Musíme proto postupovat takto:

Do snímače děrné pásky dáme větší cívku děrné pásky. Nastavíme adresu ØØØ1ØØ. Stiskneme tlačítko INT ENABLE, čímž zapneme přerušovací systém. Dále již netiskneme tlačítko PRESET – tím bychom opět přerušovací systém vypnuli. O tom, zda je přerušovací systém zapnut nebo vypnut, nás informuje indikační žárovka v tlačítku INT ENABLE. Pokud svítí, je přerušovací systém zapnut. Nyní stiskneme tlačítko RUN. Snímač by měl začít číst děrnou pásku. Snímání zastavíme tlačítkem HALT.

Pokud vypneme přerušovací systém tlačítkem PRESET, nastavíme adresu ØØØ1ØØ a stiskneme tlačítko RUN, páska ve snímači se přesune o jeden znak a počítač bude v dynamickém stopu na instrukci JMP*.

## 5.3. PŘENOS DAT KANÁLEM DMA

### Úvod

Dalším způsobem komunikace periferního zařízení s počítačem je *přímý přístup do paměti* (DMA). Při tomto způsobu se přenášejí celé bloky dat, tj. nikoliv pouze jedna hodnota. Spouštíme-li kanál DMA, musíme zadat některé informace, a to především:

a) požadovaný směr přenosu, buď vstup, nebo výstup;
b) adresu a rozsah vyrovnávací paměti v počítači;
c) adresu periferního zařízení.



### Příprava cvičení

Pro spuštění kanálu DMA jsou nutná tři *řídicí slova*. První řídicí slovo obsahuje následující informace: patnáctý bit udává, že se má vydávat signál STC po přenosu každého slova, čtrnáctý bit udává, zda se mají dva znaky z periferního zařízení zhušťovat nebo rozvinovat, a třináctý bit udává, zda se má po přenosu vydat signál CLC. Bity Ø až 5 udávají výběrový kód (SC) daného zařízení. Druhé řídicí slovo udává na patnáctém bitu, zda jde o přenos do počítače, tj. patnáctý bit je roven 1, nebo jde o výstup z počítače, patnáctý bit je tedy roven Ø. Na dalších bitech je adresa, kam se má uložit první přečtené slovo. Třetí řídicí slovo udává ve dvojkovém doplňku počet slov v bloku dat. Řídicí slova se předávají kanálu způsobem DMA uvedeným v programu.

Ukončení přenosu bloku dat kanálem DMA se signalizuje přerušením (SC Ø6) – pokud chceme využít přerušovací systém. Ukončení přenosu lze testovat dotazem SFS Ø6. Přenos bloku dat můžeme předčasně ukončit programově instrukcí STF Ø6 – tj. nastavením signálu F na jedničku. Posloupnost instrukcí při programování kanálu DMA je pevně určena. Nejdříve se naplní registry kanálu DMA řídicími informacemi, pak se odstartuje periferní zařízení a spustí se kanál DMA. Periferní zařízení dokončí zahájenou operaci a vrací signál F=1. To je pokyn pro kanál DMA, aby přenesl hodnotu do operační paměti a opět spustil periferní zařízení. To probíhá tak dlouho, dokud se nepřenese celý blok. Po přečtení celého bloku činnost kanálu DMA končí.

### Zadání

Vložte do počítače následující program pro komunikaci snímače děrné pásky s pamětí počítače kanálem DMA.

```
ØØØ77    1Ø2Ø77    HLT 77B
ØØ1ØØ    ØØØØØØ    NOP
ØØ1Ø1    1Ø77ØØ    CLC Ø,C
ØØ1Ø2    Ø6Ø12Ø    LDA DMACW
ØØ1Ø3    1Ø26Ø6    OTA 6
ØØ1Ø4    1Ø67Ø2    CLC 2
ØØ1Ø5    Ø6Ø121    LDA ADDRS
ØØ1Ø6    Ø32123    IOR B15
ØØ1Ø7    1Ø26Ø2    OTA 2
```

```
ØØ11Ø   1Ø27Ø2   STC 2
ØØ111   Ø6Ø122   LDA WCNT
ØØ112   1Ø26Ø2   OTA 2
ØØ113   1Ø371Ø   LTC 1ØB,C
ØØ114   1Ø37Ø6   LTC 6,C
ØØ115   1Ø23Ø6   SFS 6
ØØ116   Ø24115   JMP *-1
ØØ117   Ø24Ø77   JMP 77B
ØØ12Ø   16ØØ1Ø   DMACW OCT 16ØØ1Ø
ØØ121   ØØØ124   ADDRS DEF BUFFR
ØØ122   177777   WCNT OCT -1
ØØ123   1ØØØØØ   B15 OCT 1ØØØØØ
ØØ124   ØØØØØØ   BUFFR NOP
```

Popis cvičení

Na adrese ØØØØ77 je instrukce HLT 77B – tam se vracíme po správném ukončení programu. Instrukce na buňce ØØØ1ØØ je rezervována (NOP) pro RET, pokud bychom chtěli z daného programu udělat podprogram. Instrukce CLC Ø,C vypíná všechna periferní zařízení. Instrukcí LDA DMACW přebíráme do registru A první řídicí slovo, které je na adrese DMACW, tj. na adrese ØØØ121. Obsah tohoto slova znamená, že se bude vydávat signál STC po přenosu každého slova, dva znaky se budou zhušťovat do jednoho slova a po skončení přenosu bloku se vyšle signál CLC. Na posledních šesti bitech se nachází výběrový kód daného periferního zařízení, tj. 1ØB. Instrukce OTA 6 přenáší první řídicí slovo do registrů kanálu DMA. Instrukce CLC 2 připravuje registry kanálu DMA pro přenos druhého řídicího slova. Instrukcí LDA ADDRS se přebírá adresa první volné buňky, která je ØØØ124. Instrukcí IOR B15 se k této adrese přidává na patnáctý bit jednička, což znamená, že jde o přenos do počítače. Tato hodnota se instrukcí OTA 2 posílá do registru kanálu DMA. Instrukcí STC 2 se připravuje registr kanálu DMA pro přenos třetího řídicího slova, které se do registru A přenáší instrukcí LDA WCNT. Ve slově WCNT na adrese ØØØ122 je udán počet slov, která se přenášejí. V tomto případě se přenáší jedno slovo. Instrukcí OTA 2 se přenese třetí řídicí slovo do registru kanálu DMA. Instrukcí STC 1ØB,C se spustí snímač, instrukcí STC 6,C se spustí kanál DMA. Další spouštění snímače již zajistí kanál DMA. Instrukcí SFS se testuje dokončení přenosu bloku dat. Pokud není přenos



dokončen, provádí se další testování, po dokončení přenosu se provede skok na instrukci HLT 77B a počítač se zastaví.

Do snímače dáme děrnou pásku a program odzkoušíme pomocí tlačítka SINGLE CYCLE. Bude-li program správně pracovat, můžeme program odzkoušet v režimu RUN. Nastavíme adresu ØØØ1ØØ a stiskneme tlačítko RUN. Nyní můžeme změnit počet přenášených slov, na adresu ØØØ122 nahrajeme např. hodnotu 177ØØØ. Dbáme ale na to, aby ve snímači byla dostatečně velká zásoba děrné pásky. Nastavíme adresu Ø7Ø1ØØ a stiskneme tlačítko RUN. Snímač se rozeběhne. Stiskneme-li tlačítko HALT, počítač se sice zastaví, ale snímač pracuje dále, protože je ovládán přes kanál DMA, a přečte se celý blok dat. Jestliže v tomto případě čteme více slov, přečte se při tisknutí tlačítka SINGLE CYCLE celý blok a ne pouze jeden znak.

Úkoly

1. Pomocí ovládacího panelu vložte do počítače příslušný program.
2. Pomocí tlačítka SINGLE CYCLE nebo podle pokynů u jednotlivých úloh prověřte správnou funkci programu.
3. Povšimněte si rozdílu mezi normálním a zrychleným čtením.
4. Napište program pro přenos dat z počítače do děrovače děrné pásky s použitím kanálu DMA.

## 5.4. DIAGNOSTIKA ARITMETICKÉ A LOGICKÉ JEDNOTKY

Úvod

Úkolem diagnostiky je prověřit správnou funkci instrukcí aritmetické a logické jednotky. Pracujeme na libovolném počítači řady ADT. Zprávy jsou operátorovi předávány prostřednictvím pracovních registrů T, A, B. Operátor může zpětně předávat zprávy pomocí klávesnice nebo registru SW.

Příprava cvičení

Testy pro diagnostiku aritmetické a logické jednotky jsou rozděleny do skupin s hierarchií definovanou podle klesající priority. Jednotlivé skupiny jsou tvořeny řetězci testů tak, aby se pro testování instrukcí použí-

valy jen ověřené kombinace z instrukčního kódu. Plyne z toho nutnost zachovat posloupnost programů. Při zjištění vadného vykonávání instrukce je možné dále pokračovat v testování. Je však nutné analyzovat následující testy a ty, které obsahují vadnou instrukci, nejsou pak směrodatné.

Zpravidla se na začátku testu nepředpokládá žádná správná funkce. Nejdříve se otestují funkce tlačítek. Pomocí tlačítek se z řídicího panelu otestuje základní skupina instrukcí tvořící obslužný program periferního zařízení. Toto periferní zařízení pak umožní vložení složitějšího diagnostického programu.

## Zadání

Podle dalšího popisu ověřte funkci:
1. vybraných tlačítek počítače;
2. vybraných instrukcí.

## Popis cvičení

Po seznámení s jednotlivými tlačítky na řídicím panelu počítače budeme ověřovat funkci registru SW pomocí kontrolního vzorku.

Kontrolní vzorek budeme zavádět přes registry SW. Při testování je nutné nastavit na tomto registru alespoň tyto kontrolní vzorky:

| | | |
|---|---|---|
| nulový | $000000_8$ | $0000000000000000_2$ |
| šachovnicový | $055555_8$ | $0101010101010101_2$ |
| jedničkový | $177777_8$ | $1111111111111111_2$ |

Každé tlačítko registru SW se při tisknutí musí nastavovat střídavě do stavu „1" a do stavu „0". Pro otestování tlačítka LOAD A provedeme následující manipulace:

a) nastavení kontrolního vzorku do registru SW;
b) stisknutí tlačítka LOAD A, vzorek se zobrazí v registru A;
c) další stisknutí tlačítka LOAD A, všechny registry beze změny.

Tlačítko LOAD MEMORY umožňuje zápis do paměti. Jeho správnou funkci ověříme následujícím postupem:

a) nastavení adresy;
b) nastavení kontrolního vzorku;
c) stisknutí tlačítka LOAD MEMORY – vzorek se zobrazí v registru T a zapíše se do paměťového místa adresovaného obsahem registru M;



d) další stisknutí tlačítka LOAD MEMORY – obsah registrů M a P se inkrementují (přičítá se „1") a kontrolní vzorek se zapisuje do následujících paměťových míst.

K ověření správné funkce základních instrukcí počítače ADT byly vybrány instrukce HLT, JMP, LIB, INB. Příslušné testovací programy je nutné před spuštěním vložit do paměti počítače.

a) Instrukce HLT, zastavení programu

```
2ØØ    ØØØØØØ    NOP
2Ø1    ØØØØØØ    NOP
2Ø2    ØØØØØØ    NOP
2Ø3    1Ø2Ø77    HLT 77B
```

b) Instrukce JMP, nepodmíněný skok

```
2ØØ    Ø242Ø1    JMP 2Ø1B
2Ø1    ØØØØØØ    NOP
2Ø2    Ø242ØØ    JMP *-2
2Ø3    1Ø2ØØ1    HLT Ø1B
```

c) Instrukce LIB, čtení obsahu paměťového místa do registru B

```
2ØØ    1Ø65Ø1    LIB Ø1B
2Ø1    Ø242ØØ    JMB *-1
2Ø2    1Ø2ØØ1    HLT Ø1B
```

d) Instrukce INB, inkrementace registru B

```
2ØØ    1Ø65Ø1    LIB Ø1B
2Ø1    ØØ6ØØ4    INB
2Ø2    Ø242ØØ    JMP *-2
2Ø3    1Ø2ØØ1    HLT Ø1B
```

## Úkoly

1. Ověřte činnost všech tlačítek řídicího panelu počítače.
2. Mimo zadané instrukce vyberte dalších pět – sestavte testovací programy a ověřte činnost instrukcí.
3. Proveďte analýzu programů a objasněte případy chybného vykonávání instrukcí.

## 5.5. DIAGNOSTIKA OPERAČNÍ PAMĚTI

### Úvod

Operační paměti je nutné věnovat zvlášť mimořádnou pozornost. K její správné funkci je nutné, aby bylo možné správně adresovat všechna její adresová místa a mimo to aby na každé z míst bylo možné zapsat libovolný vzorek (a číst). Diagnostické paměťové testy lze rozčlenit na dvě základní skupiny – diagnostické adresové testy a vzorkové testy.

Existují dva *adresové testy*, horní a dolní. Horní adresový test se umísťuje na konec paměti a kontroluje všechny adresy v paměti uložené pod ním. Naopak dolní adresový test je uložen na začátku paměti a kontroluje všechna vyšší místa. Oba testy kontrolují adresovací obvody a vyvolají zastavení testu při zjištění chyby. Způsob signalizace chyby se liší stupněm složitosti diagnostického programu.

*Vzorkové testy* jsou opět dva – horní a dolní. Pomocí vzorkových testů se ověřuje správné uchování dat v operační paměti a kontroluje se přenos dat mezi registrem T a pamětí. Při chybě dojde k zastavení testu.

Všechny tyto testy mohou pracovat automaticky, tzn. že po otestování horní části paměti se test programově přesune z dolní části do horní a provádí se testování dolní části paměti, která byla předtím obsazena testem.

### Zadání

Pro zvolený typ počítače ADT se seznamte s dodanými paměťovými testy. Zpravidla jsou k dispozici tyto testy:
TEST 1 – šachovnicový test se střídáním slov;
TEST 2 – šachovnicový test s nulovými bity;
TEST 3 – šachovnicový test s jedničkovými bity.

### Popis cvičení

a) Zaveďte adresový test (horní, dolní), spusťte jej, sledujte chování testu.

b) Zaveďte vzorkový test (horní, dolní), spusťte jej, sledujte chování testu.



### Úkoly

1. Seznamte se se způsobem vyhodnocování chybovosti během diagnostického testu.
2. Pokuste se o sestavení vlastního krátkého paměťového vzorkového testu.

# 6. Periferní zařízení – cvičení

V této části učebnice jsou popsány návody ke cvičení z připojování nejpoužívanějších periferních zařízení k počítači.

Každé cvičení obsahuje zadání, ve kterém je vytčen jeho hlavní cíl. V úvodu je stručná charakteristika funkce periferního zařízení, popis připojení a postup při testování. Do seznamu použitých přístrojů jsou zahrnuty potřebné technické prostředky pro cvičení. Jako vzorový byl vybrán počítač ADT 4400 vyráběný v roce 1985. Postup cvičení je osnovou pro konkrétní cvičení, které je možné upravit podle vybavení počítače. Při použití jiného typu počítače je nutné přepsat testovací programy do odpovídajícího jazyka symbolických adres.

Testování připojení periferních zařízení k počítači se provádí pouze programovými prostředky. Cvičení je možné doplnit i měřením signálů osciloskopem nebo logickou sondou. U jednoduchých testovacích programů je nutné programy, které jsou uvedeny v jazyku symbolických adres, přeložit do strojového kódu počítače. Překlad by měl být kontrolou, zda je student na cvičení připraven. U složitějších programů je uveden i přeložený program, protože chyba ve strojovém kódu se obtížně hledá. V těchto cvičeních je kladen důraz na pochopení principu ovládání složitých periferních zařízení s řadičem.

Do omezeného rozsahu této knihy nebylo možné zahrnout úplný návod. Proto jsou nezbytnou pomůckou návody a výkresové dokumentace dodávané výrobcem zařízení. Obsluha a údržba složitých periferních zařízení jsou náplní dlouhodobých kursů. Je tedy nutné chápat tato cvičení jako úvod do problematiky testování a provozu periferních zařízení počítačů.

## 6.1. DĚROVAČ DĚRNÉ PÁSKY

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou děrovače děrné pásky. Řiďte se daným postupem a výsledky



zpracujte ve zprávě o cvičení. Děrnou pásku získanou ve cvičení uchovejte pro cvičení se snímačem děrné pásky.

### Úvod

Děrovač děrné pásky je v současné době nejzákladnějším výstupním zařízením minipočítačů ADT pro záznam informací na děrnou pásku.

Děrovač lze ovládat v režimu OFF – LINE tlačítky SP a WS. Při zmáčknutém tlačítku SP se děruje vodicí stopa. Stiskneme-li současně tlačítko WS, děrují se otvory ve všech stopách. Tím lze ověřit provozuschopnost děrovače ještě před připojením k počítači.

Na šířku papírové pásky lze vyděrovat buď 5, nebo 8 stop. Každé stopě přísluší jeden informační signál, který je vysílán z počítače. Rozhraní mezi počítačem a děrovačem je paralelní a všechny signály se předávají na úrovni TTL. Deska rozhraní obsahuje pouze výstupní osmibitovou paměť a synchronizační obvod. Po vodičích označených BIT0 až BIT7 se předávají data. Vodič SO signalizuje, že zdroj dat (počítač) je připraven vysílat data. Signálem SC se potvrzují data na vodičích BIT0 až BIT7 a zároveň odstartuje děrování znaku. Po vyděrování znaku odpoví děrovač signálem AC, který jednak vynuluje signál SC a současně povolí počítači vyslání dalšího znaku.

Abychom prověřili, je-li děrovač připojen k počítači, musíme nejdříve zkontrolovat správnou posloupnost synchronizačních signálů SC a AC a potom otestovat datové signály vhodnými kombinacemi bitů.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-082 a výkres desky, děrovač děrné pásky DT-105s, speciální pravítko z příslušenství a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-082 a s návodem obsluhy děrovače. Naučte se zakládat novou pásku do zásobníku děrovače a v režimu OFF – LINE vyděrujte tlačítky SP a WS střídavě prázdnou a plně vyděrovanou pásku.

2. V domácí přípravě přeložte testovací program do strojového kódu a zaveďte ho do operační paměti počítače. Přesvědčte se, zda je děrovač připojen k počítači na správné vstupní/výstupní adrese.

3. Do čítače instrukcí nastavte počáteční adresu testovacího programu. Tlačítkem INSTR.STEP krokujte program po jednotlivých instrukcích. V každém kroku sledujte registr A a čítač instrukcí. Podle komentáře u instrukce kontrolujte vykonávání programu. Pokud se motor děrovače po instrukci STC SC, C nerozběhne, je závada v rozhraní nebo v děrovači. Závadu musíte odstranit. Po dosažení instrukce JMP OPAK vypněte děrovač nebo odpojte kabel k počítači a krokujte výstup další slabiky. Testovací program po dosažení instrukcí SFS SC a JMP *−1 je bude neustále vykonávat. Tímto ověříme synchronizační obvody na desce rozhraní a v děrovači.

4. Připravte děrovač opět k děrování a tlačítkem SP vyděrujte asi 30 cm prázdné pásky. Do čítače instrukcí nastavte opět počáteční adresu testovacího programu a odstartuje počítač tlačítkem RUN. Děrovač postupně vyděruje všechny kombinace znaků, které vznikly binárním čítáním. Zkontrolujte obsah pásky a dále speciálním pravítkem zkontrolujte rozteče vyděrovaných otvorů. Děrnou pásku si uchovejte pro cvičení se snímačem děrné pásky.

### Testovací program

```
      ORG 100B
SC    EQU  xx        V/V ADRESA DESKY ROZHRANI
START CLA            VYNULUJ REGISTR A
      CLC SC         VYNULUJ BIT C NA DESCE ROZHRANI
OPAK  OTA SC         ZAPIS REGISTR A DO DESKY ROZHRANI
      STC SC,C       START DEROVANI ZNAKU
      SFS SC         JE OPERACE DOKONCENA ?
      JMP *-1        NE, OPAKUJ TESTOVANI
      INA            ANO, IKREMENTUJ REGISTR A
      CPA B400       JE UZ VSE VYDEROVANO ?
      RSS            ANO, PRESKOC
      JMP OPAK       NE, DERUJ DALSI ZNAK
      HLT 77B        ZASTAVENI DEROVANI
      JMP START      OPAKUJ TESTOVACI PROGRAM
B400  OCT 400        POCET VYDEROVANYCH ZNAKU
```

## 6.2. SNÍMAČ DĚRNÉ PÁSKY

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou snímače děrné pásky. Pracujte podle daného postupu a výsledky zpracujte ve zprávě o cvičení. Jako testovací pásku použijte výstupní pásku ze cvičení s děrovačem.



### Úvod

Snímač děrné pásky je nejdůležitějším vstupním zařízením minipočítačů ADT. Slouží k zavedení dat a programů archivovaných na děrné pásce do operační paměti počítače.

Otvory v děrné pásce jsou snímány fotoelektricky. Každé stopě přísluší jeden informační signál, který je vysílán do počítače. Snímač nelze vyzkoušet v režimu OFF−LINE tlačítky, ale lze sestrojit jednoduchý přípravek, kterým se ovládají signály START a STOP. Rozhraní snímače a počítače je paralelní. Výrobce dodává snímače a jejich rozhraní v několika variantách. Každý datový signál je veden dvěma vodiči v přímém a negovaném tvaru. Deska rozhraní obsahuje pouze vstupní osmibitovou vyrovnávací paměť a synchronizační obvod. Signálem START se uvolní brzda snímače a po příchodu odpovědi signálem FLAG se signál START vynuluje a signál STOP se nastaví na jedničku. Signál FLAG je přímo odvozen od vodicí stopy. Vodicí stopa je děrována menšími otvory, aby se získal potřebný předstih pro zabrzdění pásky. Signály z fotonek jsou přes tvarovací obvody přímo vedeny do vstupní paměti desky rozhraní, kam se zapíší signálem FLAG.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-081 a výkres, snímač FS-1501 nebo FS-751 a návod k obsluze.

### Postup cvičení

1. Seznamte se s výkresovou dokumentací desky rozhraní 03-081 a s návodem obsluhy snímače děrné pásky.

2. V domácí přípravě přeložte testovací program do strojového kódu a z řídicího panelu ho zaveďte do operační paměti počítače. Zkontrolujte adresu desky rozhraní snímače v počítači.

3. Do snímače založte děrnou pásku získanou v cvičení s děrovačem tak, aby světelný proužek osvětloval pásku těsně před prvním nenulovým znakem. Čítač instrukcí nastavte počáteční adresou testovacího programu. Tlačítkem INSTR.STEP krokujte testovací program. V každém kroku sledujte registry A a B a čítač instrukcí. Po vykonání instrukce STC SC,C musí snímač načíst jednu slabiku (znak). Pokud se tak nestane, zkontrolujte

připojení k počítači a napájení snímače. Po vykonání instrukce LIB SC zkontrolujte obsah registru B se znakem vyděrovaným na pásce. Po dosažení instrukce JMP OPAK odklopte raménko snímače. Dalším krokováním přejde program do nekonečné smyčky. Tímto prověříte synchronizační obvody desky rozhraní a snímače.

4. Založte děrnou pásku do snímače stejně jako v bodě 3 a do čítače instrukcí nastavte počáteční adresu testovacího programu. Stiskněte tlačítko RUN. Snímač zahájí čtení děrné pásky. Jestliže snímač přečetl děrnou pásku správně, počítač se zastaví na instrukci HLT 77B. Dojde-li k chybě, počítač se zastaví na instrukci HLT 55B a registr A obsahuje očekávaná data a v registru B jsou načtená data.

5. Při trvalém testování snímače lze děrnou pásku slepit v nekonečnou smyčku a vynulováním instrukce HLT 77B lze program neustále opakovat.

### Testovací program

```
      ORG 100B
SC    EQU xx          V/V ADRESA DESKY ROZHRANI
START CLA             VYNULUJ REGISTR A
      CLC SC          VYNULUJ BIT C NA DESCE ROZHRANI
OPAK  STC SC,C        START CTENI ZNAKU
      SFS SC          JE DOKONCENA OPERACE ?
      JMP *-1         NE, OPAKUJ TESTOVANI
      LIB SC          ANO, PREVEZMI ZNAK
      INA             INKREMENTUJ REGISTR A
      CPB 0           JE NACTENY ZNAK = OCEKAVANY ?
      RSS             ANO, PRESKOC
      HLT 55B         NE, CHYBA
      CPB B377        JE UZ VSE NACTENO ?
      RSS             ANO, PRESKOC
      JMP OPAK        NE, JDI NA CTENI DALSIHO ZNAKU
      HLT 77B         KONEC CTENI PASKY
      JMP START       OPAKUJ TESTOVACI PROGRAM
B377  OCT 377         POCET OCEKAVANYCH ZNAKU NA PASCE
```

## 6.3. SNÍMAČ DĚRNÝCH ŠTÍTKŮ

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou snímače děrných štítků. Pracujte podle postupu cvičení a výsledky zpracujte do zprávy o cvičení.



### Úvod

Snímač děrných štítků se používá pro vstup informací do počítače z děrných štítků. Děrné štítky se děrují na pořizovačích děrných štítků.

Snímač je k počítači připojen paralelně a všechny signály se předávají na úrovni TTL. Deska rozhraní obsahuje vstupní vyrovnávací paměť, do které se zapisují čtená data a stavové signály. V bitu 15 se zobrazuje součtová chyba z bitů 13 a 14. Signál COMMAND (SC) odstartuje čtení sloupce štítků. Po načtení se data zapíší do vstupní paměti na desce rozhraní signálem FLAG (AC), který zároveň informuje počítač o ukončení čtení sloupce štítků (jednoho znaku).

Kontrolu snímače provedeme tak, že celý štítek načteme do operační paměti počítače a porovnáme se skutečným obsahem.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-088 a výkres desky, snímač děrných štítků a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-088 a s návodem obsluhy snímače děrných štítků.

2. V domácí přípravě přeložte testovací program do strojového kódu a zaveďte ho do operační paměti počítače. Přesvědčte se, zda je snímač připojen k počítači na správné vstupní/výstupní adrese.

3. Do zásobníku ve snímači děrných štítků vložte testovací štítek. Můžete použít štítek s libovolným obsahem. Obsah štítku si poznamenejte před vložením do snímače. Testovací program nelze projít po jednotlivých instrukcích, protože po odstartování prvního sloupce štítku následuje čtení dalších. Jestliže se včas neodeberou data ze snímače, dojde k jejich ztrátě. Odstartujte testovací program v počítači tlačítkem RUN. Program načte obsah štítku do operační paměti od adresy PAM. Porovnejte obsah v paměti s daty vyděrovanými na testovaném štítku. Jestliže nedojde k přenosu dat ze štítku do operační paměti, zkontrolujte překlad programu a jeho zavedení do operační paměti počítače.

Testovací program

```
      ORG 100B
SC    EQU xx          V/V ADRESA DESKY ROZHRANI
START LDA M80         PREVEZMI  ZAPORNY  POCET SLOUPCU NA STITKU
      STA CITAC        A NASTAV CITAC
      LDA APAM        PREVEZMI  POCATECNI ADRESU PAMETI STITKU
      STA UK           A NASTAV UKAZATEL
      CLC SC          VYNULUJ BIT C NA DESCE ROZHRANI
OPAK  STC SC,C        START CTENI ZNAKU
      SFS SC          JE DOKONENA OPERACE ?
      JMP *-1         NE, OPAKUJ TESTOVANI
      LIA SC          ANO, PREVEZMI ZNAK
      STA UK,I        ULOZ SLOUPEC STITKU
      ISZ UK          ZVYS ADRESU UKLADANI
      ISZ CITAC       ZAPOCITEJ SLOUPEC A TESTUJ KONEC
      JMP OPAK        NE, JDI NA CTENI DALSIHO ZNAKU
      HLT 77B         KONEC CTENI STITKU
      JMP START       OPAKUJ TESTOVACI PROGRAM
M80   DEC -80
CITAC BSS 1           CITAC POCTU NACTENYCH SLOUPCU
APAM  DEF PAM         POCATECNI ADRESA PAMETI STITKU
PAM   BSS 80          PAMET STITKU
```

## 6.4. ELEKTRICKÝ PSACÍ STROJ

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou elektrického psacího stroje. Pracujte podle daného postupu a výsledky zpracujte do zprávy o cvičení.

### Úvod

Elektrický psací stroj byl ještě nedávno nezbytným periferním zařízením minipočítače. Sdružuje několik funkcí najednou. Především se používá ke konverzaci s operátorem a k tisku protokolů. Jeho velkou nevýhodou je malá rychlost tisku.

Znaky se přenášejí v kódu ISO-7. Deska rozhraní obsahuje vstupní a výstupní osmibitové vyrovnávací paměti. Synchronizační signály COMMAND a FLAG jsou doplněny ještě signály IN a OUT, které ovládají směr přenosu dat. Při výstupu znaku do psacího stroje musí být znak nastaven ve výstupní paměti, nastaven směr přenosu OUT a teprve potom se signálem COMMAND zahájí tisk znaku. Signál FLAG, který informuje o ukončení tisku, povolí počítači výstup dalšího znaku.



Jestliže chceme přečíst znak z klávesnice, musíme nejdříve nastavit směr IN a pak odstartovat operaci čtení. Po stisknutí některé klávesy psacího stroje se přenese odpovídající kód znaku do desky rozhraní, kde je signálem FLAG zapsán do vstupní paměti, a zároveň signál FLAG nuluje signál COMMAND a informuje počítač o ukončení přenosu.

U psacího stroje nejdříve prověříme výstup znaků a potom vstup z klávesnice. Ověřený výstup nám pomůže zkontrolovat klávesnici výpisem znaku.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-083 a výkres desky, elektrický psací stroj CONSUL s elektronikou a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-083 a s návodem obsluhy elektrického psacího stroje CONSUL. Naučte se vkládat do psacího stroje papír.
2. V domácí přípravě přeložte testovací program a z řídicího panelu ho zaveďte do operační paměti počítače.
3. Odkrokujte několik cyklů první části testovacího programu po jednotlivých instrukcích. Sledováním čítače instrukcí a registrů kontrolujte průběh testovacího programu. V každém cyklu po vykonání instrukce STC SC,C psací stroj vytiskne jeden znak. Po ověření výstupu znaku odpojte psací stroj od počítače. Dalším krokováním zkontrolujte signál FLAG. V této fázi musí program neustále vykonávat instrukce, kterými se testuje signál FLAG. Po kontrole připojení nastavte do čítače instrukcí opět počáteční adresu testu a stiskněte RUN.
4. Kontrolu klávesnice proveďte pomocí druhé části testovacího programu. Program opět krokujte po instrukcích. Po dosažení čekací smyčky stiskněte klávesu psacího stroje. Jakmile se kód klávesy přenese do registru A, zkontrolujte ho podle tabulky ISO-7.
5. Testování výstupu znaků změňte tak, aby se tiskly znaky malé abecedy.

### Testovací program

```
SC    EQU xx          V/V ADRESA DESKY ROZHRANI
STR1  LDA B40         NASTAV POCATECNI ZNAK
      CLC SC          NULUJ BIT C NA DESCE ROZHRANI
OPAK1 JSB VYSTZ       VYSTUP ZNAKU
      INA             INKREMENTUJ KOD ZNAKU
      CPA B140        JE POSLEDNI ZNAK ?
      RSS             ANO, PRESKOC
      JMP OPAK1       NE, OPAKUJ VYSTUP
      LDA LF
      JSB VYSTZ       PREJDI NA NOVY RADEK
      LDA CR
      JSB VYSTZ       PREJDI NA ZACATEK RADKU
      HLT 77B
      JMP STR1
*
VYSTZ NOP             VYSTUP JEDNOHO ZNAKU
      LDB SMVY        NA DESCE ROZHRANI NASTAV
      OTB SC           SMER VYSTUPU ZNAKU
      AND B177        PONECH POUZE KOD ZNAKU
      OTA SC          ZAPIS ZNAKU DO DESKY ROZHRANI
      STC SC,C        START TISKU
      SFS SC          JE VYSTUPNI OPERACE DOKONCENA ?
      JMP *-1         NE, OPAKUJ TESTOVANI
      JMP VYSTZ,I
*
STR2  CLC SC          NULUJ BIT C NA DESCE ROZHRANI
OPAK2 LDA SMVS
      OTA SC          NASTAV SMER VSTUPU ZNAKU
      STC SC,C        START CTENI KLAVESY
      SFS SC          JE VSTUPNI OPERACE DOKONCENA ?
      JMP *-1         NE, OPAKUJ TESTOVANI
      LIA SC          ANO, PREVEZMI ZNAK
      JSB VYSTZ       KONTROLNI VYSTUP ZNAKU
      JMP OPAK2       NEUSTALE OPAKUJ
*
B40   OCT 40          KOD MEZERY
B140  OCT 140         KOD POSLEDNIHO ZNAKU +1
SMVY  OCT 100000      RIDICI SLOVO PRO VYSTUP ZNAKU
SMVS  OCT 140000      RIDICI SLOVO PRO VSTUP ZNAKU
LF    OCT 12          KOD NOVEHO RADKU
CR    OCT 15          KOD NAVRATU VOZIKU
```

## 6.5. BODOVÁ TISKÁRNA

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou bodové tiskárny. Pracujte podle daného postupu a výsledky zpracujte ve zprávě o cvičení.



### Úvod

Bodová tiskárna je standardním výstupním zařízením minipočítačů pro tisk dat a protokolů. Pro tisk lze použít papír v roli nebo tabelační papír.

Máme-li tiskárnu s transportním zařízením, lze ovládat posun papíru v režimu OFF-LINE tlačítky, které jsou v pravé části transportního zařízení.

Rozhraní mezi počítačem a tiskárnou je paralelní a všechny signály se předávají na úrovni TTL. K desce rozhraní lze připojit nejen tiskárnu, ale i klávesnici. Ve cvičení se budeme zabývat pouze připojením tiskárny. Deska rozhraní obsahuje sedmibitovou výstupní paměť, generátor parity, osmibitovou vstupní paměť pro připojení klávesnice, obvod vyhodnocení parity vstupních dat a synchronizační obvod. Po vodičích označených SI-1 až SI-7 se předávají data, která se signálem SC zapíší do vstupní vyrovnávací paměti tiskárny. Zároveň s nastavením signálu SC se odstartuje tisk. Po vodičích SI-8 a SI-9 se předávají signály parity. V testovacím programu se parita nepoužívá. Po vytištění znaku odpoví tiskárna signálem AC, který vynuluje signál COMMAND a informuje počítač o ukončení přenosu znaku do tiskárny.

Bodová tiskárna se prověří vytištěním všech znaků kódu ISO-7.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-089 a výkres desky, bodová tiskárna Consul 2111 a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-089 a s návodem obsluhy bodové tiskárny. Naučte se zakládat nový papír do tiskárny. U transportního zařízení seřiďte pilotní pásku se začátkem nové stránky.

2. V domácí přípravě přeložte testovací program a z řídicího panelu ho zaveďte do operační paměti počítače. Připojte tiskárnu k počítači tlačítkem K.

3. Do čítače instrukcí nastavte počáteční adresu testovacího programu a odkrokujte několik cyklů programu. Po každé instrukci STC SC,C musí tiskárna vytisknout jeden znak. Pokud se tak nestane, zjistěte závadu a odstraňte ji. Při odpojené tiskárně zůstane program v čekací smyčce – čeká

na signál FLAG. Kontrolou správné posloupnosti signálů COMMAND a FLAG jsme ověřili synchronizační obvody na desce rozhraní a v tiskárně.

4. Po kontrole spojení mezi tiskárnou a počítačem odstartujte znovu program od začátku tlačítkem RUN. Zkontrolujte všechny vytisknuté znaky podle tabulky kódu ISO-7.

5. Jestliže tiskárna má generátor znaků malé nebo ruské abecedy, upravte program pro tisk těchto znaků.

### Testovací program

```
      ORG 100B
SC    EQU xx          V/V ADRESA DESKY ROZHRANI
START LDA B40         NASTAV REGISTR A POCATECNIM ZNAKEM
      CLC SC          VYNULOVANI BIT C NA DESCE ROZHRANI
OPAK  OTA SC          ZAPIS REGISTRU A DO DESKY ROZHRANI
      STC SC,C        START TISKU ZNAKU
      SFS SC          JE OPERACE DOKONCENA ?
      JMP *-1         NE, OPAKUJ TESTOVANI
      INA             ANO, IKREMENTUJ REGISTR A
      CPA B140        JE UZ VSE VYTISTENO ?
      RSS             ANO, PRESKOC
      JMP OPAK        NE, TISKNI DALSI ZNAK
      HLT 77B         ZASTAVENI TISKU
      JMP START       OPAKUJ TESTOVACI PROGRAM
B40   OCT 40          PRVNI TISKNUTY ZNAK
B140  OCT 140         POSLEDNI+1 TISKNUTY ZNAK
```

## 6.6. ŘÁDKOVÁ TISKÁRNA

### Zadání

Úkolem cvičení je seznámit se s ovládáním, testováním a programovou obsluhou řádkové tiskárny. Pracujte podle daného postupu a výsledky zpracujte ve zprávě o cvičení.

### Úvod

Řádková tiskárna se používá pro obsáhlé tisky protokolů. U tiskárny lze ovládat v režimu OFF-LINE řádkování a stránkování papíru.

Rozhraní mezi počítačem a tiskárnou je paralelní a všechny signály se předávají na úrovni TTL. Deska rozhraní 03-086 obsahuje pouze výstupní vyrovnávací osmibitovou paměť a synchronizační obvod. Po vodičích BIT0 až BIT7 se předávají data. Signálem COMMAND se potvrzuje plat-



nost dat na vodičích BIT0 až BIT7 a současně se startuje tisk znaku. Po vytištění znaku odpoví tiskárna signálem FLAG, který povolí počítači vyslání dalšího znaku. Deska rozhraní povolí vstup signálu FLAG pouze, je-li tiskárna ve stavu READY.

K prověření připojení tiskárny k počítači musíme nejdříve zkontrolovat správnou posloupnost synchronizačních signálů COMMAND a FLAG a teprve potom otestovat tiskárnu tiskem všech znaků v každém sloupci.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-086 a výkres desky, řádková tiskárna Videoton a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-086 a s návodem obsluhy řádkové tiskárny. Naučte se zakládat nový papír do tiskárny. Seřiďte začátek stránky tabelačního papíru s pilotní páskou tiskárny.

2. V domácí přípravě přeložte testovací program a z řídicího panelu ho zaveďte do operační paměti počítače.

3. Odkrokujte několik cyklů testovacího programu. Po každé instrukci STC SC,C se musí vytisknout jeden znak. Po odpojení tiskárny od počítače dojde k neustálému testování signálu FLAG.

4. Po kontrole synchronizace spolupráce počítače a tiskárny odstartujte program od začátku tlačítkem RUN. Tiskárna začne tisknout znaky podle tabulky ISO-7. Každý následující řádek bude mít obsah posunutý o jeden znak doleva.

### Testovací program

```
      ORG 100B
SC    EQU xx          V/V ADRESA DESKY ROZHRANI
START LDA B40         PREVEZMI KOD MEZERY
      STA PZ           A NASTAV JI DO POCATECNIHO ZNAKU
      CLC SC          VYNULOVANI BIT C NA DESCE ROZHRANI
RADEK LDA M80         PREVEZMI ZAPORNE POCET ZNAKU NA RADEK
      STA CITAC        A NASTAV CITAC
      LDA PZ          NASTAV DO REGISTRU A POCATECNI ZNAK
OPAK  JSB VYSTZ       VYPIS ZNAK NA TISKARNE
      INA             ZVYS KOD ZNAKU O 1
      ISZ CITAC       JE DOKONCEN RADEK ?
      JMP OPAK        NE, OPAKUJ VYSTUP ZNAKU
      LDA PZ          PREVEZMI KOD POCATECNIHO ZNAKU
```

```
      INA             ZVYS KOD O 1
      STA PZ          A ULOZ ZPET
      CPA B140        JE UZ POSLEDNI ZNAK ?
      JMP KONEC       ANO, JDI NA KONEC TESTU
      LDA CR
      JSB VYSTZ       PREJDI NA NOVY RADEK
      LDA LF
      JSB VYSTZ       PREJDI NA ZACATEK RADKU
      JMP RADEK       TISKNI DALSI POSUNUTY RADEK
KONEC HLT 77B         ZASTEVENI NA KONCI TESTU
      JMP START
*
VYSTZ NOP             PODPROGRAM VYSTUPU ZNAKU
      OTA SC          ZAPIS REGISTRU A DO DESKY ROZHRANI
      STC SC,C        START TISKU A NULUJ BIT F NA DESCE ROZHRANI
      SFS SC          JE VYSTUPNI OPERACE DOKONCENA ?
      JMP *-1         NE, CEKEJ NA DOKONCENI OPERACE
      JMP VYSTZ,I     ANO, NAVRAT Z PROCEDURY
*
CR    OCT 15          ZACATEK RADKU
LF    OCT 12          NOVY RADEK
B40   OCT 40          PRVNI TISKNUTY ZNAK
B140  OCT 140         POSLEDNI+1 TISKNUTY ZNAK
B137  OCT 137         MASKA ZNAKU
M80   DEC -80         POCET ZNAKU NA RADEK
PZ    BSS 1           POCATECNI ZNAK NA RADKU
CITAC BSS 1           CITAC VYTISTENYCH ZNAKU NA RADKU
```

## 6.7. ABECEDNĚ ČÍSLICOVÝ OBRAZOVKOVÝ DISPLEJ

### Zadání

Úkolem zadání je seznámit se s ovládáním a programovou obsluhou abecedně číslicového obrazovkového displeje. Pracujte podle daného postupu a výsledky zpracujte do zprávy o cvičení.

### Úvod

Abecedně číslicový obrazovkový displej je nejdůležitějším periferním zařízením minipočítače. Zajišťuje přímý styk mezi člověkem a počítačem.

Displej lze provozovat v režimu LOCAL, kdy je klávesnice přímo propojena s obrazovkou. Obsah obrazovky se pak přenáší do počítače tlačítkem SEND. Častější použití je v režimu REMOTE, kdy jsou klávesnice a obrazovka dvě samostatná zařízení.

Znaky jsou přenášeny v kódu ISO-7. Deska rozhraní obsahuje vstupní a výstupní osmibitové vyrovnávací paměti. Přenášená data jsou sedmibitová. Osmý (tzv. paritní) bit slouží k jejich zabezpečení. Synchronizační



signály COMMAND a FLAG jsou určeny zvlášť pro vstup a výstup znaku. Přepínají se automaticky podle směru přenosu. Displej hlásí svůj stav pomocí 7 signálů, které jsou popsány v návodu. Stavové bity se předávají v bitech 10 až 15 při čtení dat z desky rozhraní.

### Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-181 a výkres desky, abecedně číslicový obrazovkový displej SM7202 a návod k obsluze.

### Postup cvičení

1. Seznamte se s dokumentací desky rozhraní 03-181 a s návodem obsluhy displeje SM7202. V režimu LOCAL zobrazte na obrazovce displeje všechny klávesy displeje.
2. V domácí přípravě přeložte testovací program a z řídicího panelu ho zaveďte do operační paměti počítače. Displej připojte k počítači klávesou REMOTE.
3. Nejdříve prověříme výstup znaků na obrazovku displeje. Do čítače instrukcí nastavte adresu VYZN a tlačítkem INSTR.STEP krokujte program. Po vykonání instrukce STC SC,C se na obrazovce objeví znak. Prvním znakem je mezera, proto dojde k posunutí kurzoru. Dále se budou zobrazovat znaky podle tabulky kódu ISO 7. Po několika opakováních výstupu znaku odpojte displej od počítače tlačítkem LOCAL. Následujícím krokováním se musí program uvést v nekonečnou smyčku, která testuje nastavení bitu F na desce rozhraní.
4. Po kontrole spojení displeje s počítačem nastavte znovu počáteční adresu VYZN do čítače instrukcí a odstartujte počítač tlačítkem RUN. Na obrazovce se budou zobrazovat řádky složené ze všech znaků. Každý následující řádek bude mít obsah posunutý doleva.
5. Kontrolou přenosu z klávesnice displeje do počítače proveďte druhou část testovacího programu. Do čítače instrukcí nastavte adresu CTIZN a odstartujte počítač tlačítkem RUN. Po stisknutí vybraného tlačítka klávesnice se znak načte do registru A a podprogramem VYSTZ se zobrazí na obrazovce.

## Testovací program

```
        ORG 100B
SC      EQU xx          V/V ADRESA DESKY ROZHRANI
VYZN    LDA B40         PREVEZMI KOD MEZERY
        STA PZ           A ULOZ DO POCATECNIHO ZNAKU
        CLC SC          VYNULOVANI BITU C NA DESCE ROZHRANI
RADEK   LDA M80         PREVEZMI ZAPORNE DELKU RADKU
        STA CITAC        A NASTAV CITAC ZNAKU
        LDA PZ          PREVEZMI KOD POCATECNIHO ZNAKU
OPAK1   JSB VYSTZ       VYPIS ZNAK
        INA             ZVYS KOD ZNAKU O 1
        CPA B140        JE POSLEDNI ZNAK ?
        LDA B40         ANO, NASTAV MEZERU
        ISZ CITAC       JE DOKONCEN RADEK ?
        JMP OPAK1       NE, OPAKUJ VYSTUP ZNAKU
        LDA PZ          PREVEZMI KOD POCATECNIHO ZNAKU
        INA             ZVYS KOD O 1
        CPA B140        JE UZ POSLEDNI ZNAK ?
        LDA B40         ANO, NASTAV OPET MEZERU,
        STA PZ          NAHREJ DO POCATECNIHO ZNAKU
        LDA CR
        JSB VYSTZ       PREJDI NA NOVY RADEK
        LDA LF
        JSB VYSTZ       PREJDI NA ZACATEK RADKU
        JMP RADEK
*
VYSTZ   NOP             PODPROGRAM VYSTUPU ZNAKU
        OTA SC          ZAPIS REGISTRU A DO DESKY ROZHRANI
        STC SC,C        START PRENOSU A NULOVANI BITU F
        SFS SC          JE DOKONCENA VYSTUPNI OPERACE ?
        JMP *-1         NE, OPAKUJ TESTOVANI
        JMP VYSTZ,I     ANO, NAVRAT Z PRODPROGRAMU
*
CTIZN   CLC SC          NULUJ BIT C NA DESCE ROZHRANI
OPAK2   SFS SC          JE DOKONCENA VSTUPNI OPERACE ?
        JMP *-1         NE, OPAKUJ TESTOVANI
        LIA SC          PREVEZMI ZNAK Z DESKY ROZHRANI
        JSB VYSTZ       ZOBRAZ ZNAK NA OBRAZOVKU
        JMP OPAK2       OPAKUJ
*
CR      OCT 15          ZACATEK RADKU
LF      OCT 12          NOVY RADEK
B40     OCT 40          PRVNI TISKNUTY ZNAK
B140    OCT 140         POSLEDNI+1 TISKNUTY ZNAK
B137    OCT 137         MASKA ZNAKU
M80     DEC -80         POCET ZNAKU NA RADEK
PZ      BSS 1           POCATECNI ZNAK NA RADKU
CITAC   BSS 1           CITAC VYTISTENYCH ZNAKU NA RADKU
```

# 6.8. DISKETOVÁ PAMĚŤ

## Zadání

Seznamte se s programovým ovládáním disketové paměti. Pracujte podle daného postupu.



## Úvod

Disketová paměť slouží jako vnější paměť počítače. Nahrazuje klasická periferní zařízení, jako jsou snímač a děrovač děrné pásky.

Jako paměťové médium se používá *disketa (pružný disk)* o průměru 200 mm se záznamem dvojí frekvencí. Organizace záznamu odpovídá normě ISO, č. 5654. Diskety jsou přenosné na zařízení JSEP, SMEP a na zařízení pro sběr a zpracování dat.

Jednotka pružných disků se skládá z řadiče a dvou jednotek pamětí. K jednomu řadiči lze připojit až čtyři paměti. Připojení k počítači je realizováno pouze jednou deskou rozhraní, kterou se řadiči předávají data a příkazy.

V tomto cvičení se pouze seznámíme s programovou obsluhou. K důkladnému prověření slouží test dodávaný výrobcem.

## Použité přístroje a pomůcky

Počítač typu ADT 4000, deska rozhraní 03-170 MICRO a výkres desky, disketa (pružný disk) MPD-1 a návod k obsluze, instalaci a údržbě.

## Postup cvičení

1. Prostudujte testovací program a návod k obsluze pružného disku MPD-1.
2. Do jednotky vložte testovací pružný disk. Do operační paměti zaveďte testovací program a pečlivě ho překontrolujte. Pomocí programu KROK vystavte univerzální hlavu v jednotce pružného disku. Od adresy ZP připravte zkušební data pro zápis a od adresy CP vynulujte vstupní paměť v maximální délce 64 slov. Do čítače instrukcí zaveďte adresu ZAPIS a odstartujte program tlačítkem RUN. Po každém zastavení na instrukci HLT zkontrolujte stavové slovo (SW), a je-li bez chyby, pokračujte dál.
3. Po úspěšném zápisu odstartujte program CTENI tlačítkem RUN. Program končí instrukcí HLT 77B. Po načtení zkontrolujte obsah operační paměti od adresy 2000B. Data se musí shodovat se zapsanými.
4. Po vlastní modifikaci slov N, ADRST zopakujte body 2 a 3.

## Testovací program

```
                ASMB,A,L
100                ORG 100B
020          SC    EQU 20B      V/V ADRESA DESKY ROZHRANI
*     TESTOVANI KROKOVANI VYSTAVENI HLAVY
100 060115  KROK  LDA CKRVP    KOD PRIKAZU KROKUJ VPRED
101 030120        IOR DISK      PRICTI CISLO DISKETOVE JEDNOTKY
102 030117        IOR N         PRIDEJ POCET KROKU
103 102620        OTA SC        A ZAPIS DO DESKY ROZHRANI
104 103720        STC SC,C     START PRIKAZU
105 010122        AND HEXTR    PONECH CISLO JEDNOTKY
106 030121        IOR ADRST     PRIDEJ ADRESU CILOVE STOPY
107 102620        OTA SC        A PRENES DO DESKY ROZHRANI
110 102320        SFS SC       JE OPERACE DOKONCENA ?
111 024110        JMP *-1      NE, OPAKUJ TESTOVANI
112 106720        CLC SC       ANO, UKONCI PRIKAZ
113 102520        LIA SC       PREVEZMI STAV
114 102055        HLT 55B
115 001400  CKRVP OCT 1400      PRIKAZ KROKUJ VPRED
116 003400  CKKVZ OCT 3400      PRIKAZ KROKUJ VZAD
117 000012  N     DEC 10       POCET KROKU
120 000000  DISK  DEC 0        CISLO DISKETY
121 000012  ADRST DEC 10       ADRESA STOPY
122 177400  HEXTR OCT 177400   MASKA PRO HORNI SLABIKU
```

```
*     ZAPIS BLOKU DAT
*
123 060220  ZAPIS LDA ZVPV     PREVEZMI ADRESU BLOKU DAT
124 070216       STA UKVP      A PRENES DO UKAZATELE
125 060215       LDA DM64     PREVEZMI ZAPORNOU DELKU DAT
126 070217       STA CITAC     A PRENES DO CITACE
127 060156       LDA CPD      PREVEZMI PRIKAZ, PRENES DATA
130 030120       IOR DISK      PRIDEJ CISLO DISKU
131 102620       OTA SC        A ZAPIS NA DESKU ROZHRANI
132 103720       STC SC,C     START PRIKAZU
133 102320  OPAK1 SFS SC       MOHU ZAPSAT ?
134 024133       JMP *-1      NE, OPAKUJ TESTOVANI
135 160216       LDA UKVP,I   ANO, PREVEZMI SLOVO Z PAMETI
136 103620       OTA SC,C     ZAPIS DO DESKY ROZHRANI
137 034216       ISZ UKVP     POSUN UKAZATEL
140 034217       ISZ CITAC    JE PRENESEN BLOK ?
141 024133       JMP OPAK1    NE, ZAPIS DALSI SLOVO
142 106720       CLC SC       UKONCI PRIKAZ
143 102520       LIA SC       PREVEZMI STAV
144 102001       HLT 1        ZKONTROLUJ STAV
145 060157       LDA CZD      PREVEZMI PRIKAZ, ZAPIS DATA
146 030120       IOR DISK      PRIDEJ CISLO DISKETY
147 102620       OTA SC        A PRENES DO DESKY ROZHRANI
150 103720       STC SC,C     START PRIKAZU
151 102320       SFS SC       JE DOKONCEN PRIKAZ ?
152 024151       JMP *-1      NE, OPAKUJ TESTOVANI
153 106720       CLC SC       KONEC PRIKAZU
154 102520       LIA SC       CTI STAV STAV DO REGISTRU A
155 102066       HLT 66B      KONEC ZAPISU
156 000000  CPD  OCT 0        KOD PRIKAZU PRENES DATA
157 000400  CZD  OCT 400      KOD PRIKAZU ZAPIS DATA
```



```
*     CTENI BLOKU DAT
*
160 060321  CTENI LDA CVPV     PREVEZMI  ADRESU BLOKU DAT
161 070216       STA UKVP      A PRENES DO UKAZATELE
162 060215       LDA DM64     PREVEZMI  ZAPORNOU DELKU DAT
163 070217       STA CITAC     A PRENES DO CITAC
164 060214       LDA CCD      PREVEZMI  PRIKAZ CTI DATA,
165 030120       IOR DISK      PRIDEJ CISLO DISKETY
166 102620       OTA SC        A PRENES DO DESKY ROZHRANI
167 103720       STC SC,C     START PRIKAZU
170 102320       SFS SC       JE DOKONCEN PRIKAZ
171 024170       JMP *-1      NE, OPAKUJ TESTOVANI
172 106720       CLC SC       KONEC PRIKAZU
173 102520       LIA SC       CTI STAV
174 102002       HLT 2        KONEC CTENI DAT
175 060156       LDA CPD      PREVEZMI  PRIKAZ PRENES DATA,
176 030120       IOR DISK      PRIDEJ CISLO DISKU
177 102620       OTA SC        A PRENES NA DESKU ROZHRANI
200 103720       STC SC,C     START PRIKAZU
201 102320  OPAK2 SFS SC       MOHU CIST?
202 024201       JMP *-1      NE,CEKEJ
203 102520       LIA SC       PREVEZMI SLOVO
204 170216       STA UKVP,I   ULOZ SLOVO DO PAMETI
205 034216       ISZ UKVP     POSUN UKAZATEL
206 034217       ISZ CITAC    JE PRENESEN BLOK?
207 024201       JMP OPAK2    NE, ZAPIS DALSI SLOVO
210 106720       CLC SC       UKONCI PRIKAZ
211 102520       LIA SC       PREVEZMI STAV
212 102077       HLT 77B      KONEC CTENI, ZNOVA NASTAV
213 024100       JMP KROK     JDI NA ZACATEK TEST
*
214 001000  CCD  OCT 1000     KOD CTI DATA
215 177700  DM64 DEC -64      ZAPORNA DELKA BLOKU
216 000000  UKVP NOP          UKAZATEL DO VYROVNAVACICH PAMETI
217 000000  CITAC NOP         CITAC PRENESENYCH SLOV
220 000221  ZVPV DEF *+1      POC.ADRESA ZP
221 000000  ZP   BSS 64       ZAPIS. DATA, NAPLN PRED ZAPISEM
321 000322  CVPV DEF *+1      POCATECNI ADRESA VP
322 000000  CP   BSS 64       CTENA DATA,ZKONTR. PO CTENI S ZP
                 END
```

## 6.9. MAGNETICKÁ PÁSKOVÁ PAMĚŤ

### Zadání

Úkolem cvičení je seznámit se s ovládáním magnetické páskové paměti a s její jednoduchou programovou obsluhou.

### Úvod

Magnetickou páskovou paměť zařazujeme do skupiny periferních zařízení, kterým říkáme *vnější velkokapacitní paměti.* Výhodou použití záznamu na magnetickou pásku je velká kapacita. Nevýhodou je malá rychlost přístupu k uloženým souborům dat.

Data jsou na magnetické pásce zaznamenána v záznamech. *Záznam* je složen ze šestnáctibitových slov. Délka záznamu není pevná, ale volitelná a odpovídá např. jednomu textovému řádku na terminálu. Záznamy tvoří *soubor;* každý soubor je zakončen *zónovou značkou* – tj. značkou konce souboru EOF (End Of File).

Magnetická pásková jednotka se k počítači nepřipojuje přímo, ale je připojena na tzv. *řadič magnetických páskových pamětí,* který počítači velmi zjednodušuje řízení. Jeden řadič může ovládat až čtyři magnetické páskové jednotky. Řadič je spojen s počítačem datovým kanálem a řídicím kanálem. Každý kanál vyžaduje samostatnou desku rozhraní.

Přenos dat mezi počítačem a magnetickou páskovou jednotkou je velmi rychlý. S úspěchem se využívá přenos dat pomocí přímého přístupu do paměti (DMA). V našem případě budeme používat programovou obsluhu přenosu dat, protože počítač ADT je dostatečně rychlý.

## Použité přístroje a pomůcky

Počítač typu ADT 4000, desky rozhraní 03-151 a 03-152, magnetická pásková jednotka PT105 nebo PT305 s řadičem a návod k obsluze.

## Postup cvičení

1. Seznamte se s návodem obsluhy magnetické páskové paměti. V domácí přípravě prostudujte testovací program. Máte-li možnost, napište program v jazyku symbolických adres a přeložte ho.

2. Připojte řadič magnetické páskové jednotky k počítači a do operační paměti zaveďte testovací program. Do magnetické páskové jednotky vložte testovací pásku.

3. Programem POVEL zadávejte tyto příkazy: nulování, výběr jednotky 0, 2× piš zónovou značku EOF, převinutí, zóna vpřed, zóna vzad a další příkazy podle vlastního výběru. Do registru A zapište číselný kód příkazu a program odstartujte od adresy POVEL. Pro opakování příkazu stačí pouze stisknout tlačítko RUN. Po odstartování programu POVEL sledujte, zda magnetická pásková jednotka vykonává zvolený povel.

4. Vynulujte instrukce HLT 1 a JMP POVEL+1 v podprogramu POVEL. Programem ZAPIS nahrajte na magnetickou pásku data. Po zastavení na instrukci HLT 66B můžeme několikrát opakovat zápis se změněnými daty pouze stisknutím tlačítka RUN. Data pro zápis jsou umístěna v paměti od adresy ZB.



5. Programem CTENI přečtěte z magnetické pásky data zapsaná programem ZAPIS. Po zastavení na instrukci HLT 77B zkontrolujte data načtená do operační paměti od adresy CB. Další záznam přečtěte pouze stlačením tlačítka RUN.

## Testovací program

```
                ASMB,A,L
100               ORG 100B
*
*     PRIKAZY RADICE MAGNETICKYCH PAMETI
*
100 001400  MP0   OCT 1400      VYBER JEDNOTKY 0
101 002400  MP1   OCT 2400      VYBER JEDNOTKY 1
102 004400  MP2   OCT 4400      VYBER JEDNOTKY 2
103 010400  MP3   OCT 10400     VYBER JEDNOTKY 3
104 000110  CLR   OCT 110       NULOVANI
105 000031  WCC   OCT 31        ZAPIS
106 000023  RRF   OCT 23        CTENI
107 000041  BSR   OCT 41        BLOK VZAD
110 000101  REW   OCT 101       PREVINUTI
111 000015  GAP   OCT 15        MAZANI MEZERY
112 000003  FSR   OCT 3         BLOK VPRED
113 000105  RWO   OCT 105       VYMENA
114 000211  WFM   OCT 211       PIS ZONOVOU ZNACKU
115 000203  FSF   OCT 203       ZONA VPRED
116 000241  BSF   OCT 241       ZONA VZAD
117 000215  GFM   OCT 215       PIS ZON.ZNACKU + MEZERU
120 003437  MSTS  OCT 3437      MASKA CHYBOVYCH BITU STAVU
121 000100  MBOT  OCT 100       POCATEK PASKY
*     P R O M E N N E     A    K O N S T A N T Y
122 000000  P     BSS 1         UKAZATEL DO BLOKU DAT
123 000000  CIT   BSS 1         CITAC PRENESENYCH ZNAKU
124 000125  ACB   DEF *+1        ADRESA PAMETI DAT PRO CTENI
125 000000  CB    BSS 128       CTENA DATA
325 000326  AZB   DEF *+1        ADRESA PAMETI DAT PRO ZAPIS
326 000000  ZB    BSS 128       ZAPISOVANA DATA
526 177600  DELKA DEC -128      DELKA BLOKU DAT VE SLOVECH
527 000005  CEK   DEC 5         POCET CEKACICH SMYCEK
530 000000  CCC   BSS 1         CITAC PRO CEKANI NA RADIC
000         A     EQU 0         REGISTR A
001         B     EQU 1         REGISTR B
021         COM   EQU 21B       V/V ADRESA RIDICIHO KANALU
020         DAT   EQU 20B       V/V ADRESA DATOVEHO KANALU

531 000000  POVEL NOP           VYSTUP POVELU
532 102625        OTA COM
533 103725        STC COM,C
534 102325        SFS COM
535 024534        JMP *-1
536 102001        HLT 1         VYNULOVAT PRI CTENI A ZAPISU
537 024532        JMP POVEL+1   VYNULOVAT PRI CTENI A ZAPISU
540 124531        JMP POVEL,I
*
*    DSTS   -   TEST NA SPRAVNE STAVOVE SLOVO
*
541 000000  DSTS  NOP
542 102525  C04   LIA COM       PREVEZMI STAV Z RADICE
```

```
543 070001         STA B
544 010120         AND MSTS       PONECH JEN CHYBY PASKY
545 002003         SZA,RSS
546 124541         JMP DSTS,I     NEJSOU CHYBY, NAVRAT
547 060541         LDA DSTS       DO REGISTRU A PREVEZMI ADRESU CHYBY
550 102011         HLT 11B        DOSLO K CHYBE !
551 124541         JMP DSTS,I

*      ZAPIS   - PROVEDE ZAPIS BLOKU DAT NA PASKU
*
552 060325  ZAPIS LDA AZB        PREVEZMI  ADRESU POCATKU DAT
553 070122        STA P          A NASTAV UKAZATEL
554 064526        LDB DELKA      PREVEZMI  DELKU BLOKU
555 074123        STB CIT        A NASTAV CITAC
556 060105        LDA WCC        PREVEZMI POVEL PIS
557 102625        OTA COM        A PRENES HO DO DESKY ROZHRANI
560 014541        JSB DSTS       TESTUJ STAV RADICE
561 103724        STC DAT,C      START
562 103725        STC COM,C            PRENOSU
563 102324  D02   SFS DAT        DATAVY KANAL POZADUJE SLOVO ?
564 024563        JMP *-1        NE, OPAKUJ TESTOVANI
565 160122        LDA P,I        PRENES SLOVO
566 103624        OTA DAT,C                DO DESKY SPOJENI
567 034122        ISZ P          ZVYS UKAZATEL DO BLOKU DAT
570 034123        ISZ CIT        SNIZ POCET PRENESENYCH SLOV
571 024563        JMP D02        POKUD NENI KONEC, CTI DAL
572 107724        CLC DAT,C      ZRUS ZAPIS
573 102325        SFS COM        JE UZ KONEC PRIKAZU ?
574 024573        JMP *-1        NE, OPAKUJ TESTOVANI
575 014541        JSB DSTS       ANO, TESTUJ STAV RADICE
576 102066        HLT 66B        ZASTAVENI NA KONCI ZAPISU

*      CTENI   -   PROVEDE CTENI Z MAGNETICKE PASKY
*
577 060124  CTENI LDA ACB        PREVEZMI ADRESU PAMETI PRO CTENI
600 070122        STA P          A ULOZ DO UKAZATELE
601 064526        LDB DELKA      PREVEZMI DELKU BLOKU
602 074123        STB CIT        A ULOZ DO CITACE
603 060106        LDA RRF        PRIPRAV
604 102625        OTA COM              PRIKAZ CTI
605 014541        JSB DSTS       TESTUJ RADIC
606 103724        STC DAT,C      START
607 103725        STC COM,C            PRENOSU
610 060527  LCT   LDA CEK        PROVED NASTAVENI
611 070530        STA CCC          DOBY CEKANI
612 102324  D06   SFS DAT        JE SLOVO DAT PRIPRAVENO ?
613 002001        RSS            NE
614 024621        JMP D07        ANO
615 034530        ISZ CCC        INKREMENTUJ CITAC CASOVEHO VYPADKU
616 024612        JMP D06        NEJNI-LI VYPADEK, OPAKUJ TESTOVANI
617 102021        HLT 21B        JE CASOVY VYPADEK
620 024626        JMP D08
621 103524  D07   LIA DAT,C      ANO, CTI SLOVO
622 170122        STA P,I        ULOZ DO BLOKU
623 034122        ISZ P          ZVYS UKAZATEL DO BLOKU
624 034123        ISZ CIT        SNIZ CITAC PRENOSU
625 024610        JMP LCT        JESTE NENI VSE, OPAKUJ CTENI
626 107724  D08   CLC DAT,C      ANO, ZRUS CTENI
627 102325        SFS COM          JE KONEC PRIKAZU?
630 024627        JMP *-1          NE,CEKEJ
631 014541        JSB DSTS       TESTUJ STAV RADICE
632 102077        HLT 77B        ZASTAVENI NA KONCI CTENI
633 024552        JMP ZAPIS
                  END
```



## 6.10. KAZETOVÁ DISKOVÁ PAMĚŤ

### Zadání

Seznamte se s programovým ovládáním kazetové diskové paměti. Pracujte podle daného postupu.

### Úvod

Kazetová disková paměť se připojuje k minipočítačům ADT, jestliže chceme mít rychlý přístup k velkým souborům dat.

Jednotka kazetové diskové paměti KDP-723 má dva kazetové disky, pevný a výměnný. Na každém ze čtyř povrchů těchto disků jsou umístěny 203 soustředné kružnice – *stopy*. Stopy umístěné na disku nad sebou se nazývají *válce*. Stopa je rozdělena na 24 kruhových výsečí – *sektorů*. Každý sektor obsahuje *adresové slovo, datové pole* a *slovo cyklické kontroly*. Do počítače se přenáší pouze datové pole. Adresové slovo obsahuje číslo stopy, hlavy a případně indikaci vadné nebo chráněné stopy. Datové pole obsahuje 128 šestnáctibitových slov.

Tak jako u magnetické páskové paměti je i zde mezi kazetovými diskovými jednotkami a počítačem zapojen *řadič* těchto jednotek. Konstrukčně je řadič umístěn na čtyřech deskách rozhraní. Na jeden řadič lze připojit až čtyři kazetové diskové jednotky. Řadič je spojen s počítačem datovým kanálem a řídicím kanálem.

Datovým kanálem se přenášejí data v obou směrech, stavové slovo z řadiče do počítače a adresové slovo z počítače do řadiče.

Po řídicím kanálu se přenáší z počítače do řadiče kód příkazu, dva bity adresy kazetové diskové jednotky a dva bity pro zápis indikace chráněný, popř. vadný válec. Z řadiče jsou do počítače předávány po řídicím kanále signály ATTENTION, jež určují kazetovou páskovou jednotku, která přerušuje chod počítače.

Data jsou po datovém kanále předávána kanálem přímého přístupu do paměti (DMA). Přenos ostatních informací je řízen na obou kanálech programem.

K dokonalému prověření kazetové diskové paměti slouží testovací program dodávaný výrobcem. Program, který je uveden v cvičení, je jen ukázkou, jakým způsobem se pracuje s kazetovou diskovou pamětí.

## Použité přístroje a pomůcky

Počítač typu ADT 4000, řadič RKD-723 a popis řadiče, kazetová disková paměť KDP-723 a technický popis.

## Postup cvičení

1. Prostudujte testovací program, technický popis kazetové paměti a řadiče.
2. Do kazetové diskové jednotky vložte testovací kazetový disk. Do operační paměti zaveďte testovací program a pečlivě ho překontrolujte. Od adresy 1000B připravte zkušební data pro zápis a od adresy 2000B vynulujte obsah paměti. Do čítače instrukcí nastavte adresu ZAPIS a odstartujte program tlačítkem RUN. Pokud se program zastaví na instrukci HLT 77B, proběhl zápis v pořádku. Dojde-li k zastavení na instrukci HLT 55B, nastala chyba. V registru A je uloženo stavové slovo, které musíte dekódovat a příčinu chyby odstranit.
3. Po úspěšném zápisu odstartujte program CTENI tlačítkem RUN. Program by se měl zastavit instrukcí HLT 70B. Pokud se tak nestane, postupujte stejně jako v bodě 2. Po načtení zkontrolujte obsah operační paměti od adresy 2000B. Data v operační paměti se musí shodovat s daty zapsanými na kazetovém disku.
4. Po vlastní modifikaci slov CYLNO (číslo stopy), HDSEC (číslo hlavy a sektoru) proveďte další zápis a čtení podle bodů 2 a 3.

## Testovací program

```
            ASMB,A,L
00100               ORG 100B
00100 000144 CYLNO OCT 000144 CISLO STOPY (=100)
00101 000405 HDSEC OCT 000405 CISLO HLAVY (=1) A SEKTORU (=5)
00102 120010 CW1A  OCT 120010 RIDICI SLOVO DMA,V/V ADRESA
00103 101000 CW2A  OCT 101000 POCATECNI ADRESA PRO DMA-ZAPIS
00104 101200 CW2B  OCT 101200 POCATECNI ADRESA PRO DMA-CTENI
00105 177600 CW3A  DEC -128   ZAPORNA DELKA BLOKU
00106 010000 WRCMD OCT 010000 RIDICI SLOVO ZAPIS DATA
00107 020000 RDCMD OCT 020000 RIDICI SLOVO CTI DATA
00110 030000 SKCMD OCT 030000 RIDICI SLOVO VYHLEDEJ ZAZNAM
00111 000000 STCMD OCT 000000 RIDICI SLOVO TEST ZARIZENI
00112 000000 DN    DEC 0      CISLO JEDNOTKY KAZET.DISKU
00012        DC    EQU 10     V/V ADRESA PRIKAZ.KANALU
00013        CC    EQU 11     V/V ADRESA RIDICIHO KANALU
   ZAPIS BLOKU DAT Z ADRESY 1000B A DELKY 128 SLOV
00113 014164 ZAPIS JSB SEEK   VYSTAVENI NA POZAD.ADRESU
00114 106713       CLC CC     VYNULUJ BIT C V RIDICIM KANALU
00115 060102       LDA CW1A   PRVEZMI PRVNI RIDICI SLOVO PRO DMA
```



```
00116 102606       OTA 6      A ZAPIS DO DMA
00117 106702       CLC 2      PRIPRAV DMA PRO DRUHE SLOVO
00120 060103       LDA CW2A   PREVEZMI DRUHE RIDICI SLOVO PRO DMA
00121 102602       OTA 2      A ZAPIS DO DMA
00122 102702       STC 2      PRIPRAV  DMA PRO TRETI SLOVO
00123 060105       LDA CW3A   PREVEZMI TRETI RIDICI SLOVO PRO DMA
00124 102602       OTA 2      A  ZAPIS DO DMA
00125 060106       LDA WRCMD  PRVEZMI RIDICI SLOVO ZAPIS DATA,
00126 030112       IOR DN     PRIDEJ CISLO JEDNOTKY
00127 102613       OTA CC     A ZAPIS DO RIDICIHO KANALU
00130 102112       STF DC     PRIPRAV DATOVY KANAL PRO ZAPIS
00131 103706       STC 6,C    PROVED START DMA
00132 103713       STC CC,C   PROVED START PRIKAZU ZAPIS DATA
00133 102313       SFS CC     JE KONEC OPERACE ?
00134 024133       JMP *-1    NE, OPAKUJ TESTOVANI
00135 014207       JSB STATS  ANO, PREVEZMI STAV RADICE
00136 102055       HLT 55B    CHYBA, V REGISTRU A STAV RADICE
00137 102077       HLT 77B    KONEC ZAPISU

   CTENI BLOKU DAT NA ADRESU 2000B O DELCE 128 SLOV
*
00140 014164 CTI   JSB SEEK   VYSTAV NA POZADOVANOU ADRESU
00141 106713       CLC CC     VYNULUJ BIT C V RIDICIM KANALU
00142 060102       LDA CW1A   PREVEZMI PRVNI RIDICI SLOVO DMA
00143 102606       OTA 6      A ZAPIS DO DMA
00144 106702       CLC 2      PRIPRAV DMA PRO DRUHE RIDICI SLOVO
00145 060104       LDA CW2B   PREVEZMI DRUHE RIDICI SLOVO DMA
00146 102602       OTA 2      A ZAPIS DO DMA
00147 102702       STC 2      PRIPRAV DMA PRO TRETI SLOVO
00150 102602       OTA 2      A ZAPIS DO DMA
00151 060107       LDA RDCMD  PREVEZMI RIDICI SLOVO CTI DATA,
00152 030112       IOR DN     PRIDEJ CISLO JEDNOTKY
00153 102613       OTA CC     A ZAPIS PRIKAZ DO RIDICIHO KANALU
00154 103712       STC DC,C   PRIPRAVA DATOVEHO KANALU PRO CTENI
00155 103706       STC 6,C    START DMA
00156 103713       STC CC,C   START PRIKAZU CTI DATA
00157 102313       SFS CC     JE KONEC OPERACE ?
00160 024157       JMP *-1    NE, OPAKUJ TESTOVANI
00161 014207       JSB STATS  ANO, PREVEZMI STAV
00162 102044       HLT  B     CHYBA, V REGISTRU A STAV RADICE
00163 102070       HLT 70B    KONEC CTENI, POROVNEJ DATA SE ZAPSANYMI

*   PODPROGRAM  VYHLEDEJ ZAZNAM
*
00164 000000 SEEK  NOP
00165 106713       CLC CC     VYNULUJ BIT C V RIDICIM KANALU
00166 060100       LDA CYLNO  PREVEZMI ADRESU POZADOVANEHO VALCE
00167 102612       OTA DC     A PRENES DO DATOVEHO KANALU
00170 103712       STC DC,C   ZAPIS PRVNI ADRESOVE SLOVO
00171 060110       LDA SKCMD  PREVEZMI PRIKAZ VYHLEDEJ ZAZNAM,
00172 030112       IOR DN     PRIDEJ CISLO JEDNOTKY
00173 102613       OTA CC     A ZAPIS DO RIDICIHO KANALU
00174 103713       STC CC,C   PROVED START PRIKAZU VYHLEDEJ ZAZNAM
00175 102312       SFS DC     JE ADRESA VALCE PRENESENA ?
00176 024175       JMP *-1    NE, OPAKUJ TESTOVANI
00177 060101       LDA HDSEC  PREVEZMI ADRESU HLAVY/SEKTORU
00200 102612       OTA DC     A PRENES DO DATOVEHO KANALU
00201 103712       STC DC,C   ZAPIS DRUHE ADRESOVE SLOVO
00202 102313       SFS CC     JE KONEC PRIKAZU ?
00203 024202       JMP *-1    NE, OPAKUJ TESTOVANI
00204 014207       JSB STATS  ANO, PREVEZMI STAV RADICE
00205 102033       HLT 33B    CHYBA, V REGISTRU A STAV RADICE
00206 124164       JMP SEEK,I NAVRAT
```

```
*   PODPROGRAM TEST ZARIZENI
00207 000000 STATS NOP
00210 106713       CLC CC     VYNULUJ BIT C V RIDICIM KANALU
00211 060111       LDA STCMD  PREVEZMI  PRIKAZ TEST ZARIZENI,
00212 030112       IOR DN      PRIDEJ CISLO JEDNOTKY
00213 102613       OTA CC      A ZAPIS DO RIDICI JEDNOTKY
00214 103713       STC CC,C   START PRIKAZU
00215 102312       SFS DC     JE STAVOVE SLOVO PRIPRAVENO ?
00216 024215       JMP *-1    NE, OPAKUJ TESTOVANI
00217 002011       SLA,RSS    CHYBA ?
00220 034207       ISZ STATS  NE, NAVRAT O ADRESU DAL
00221 124207       JMP STATS,I   NAVRAT
                   END
```

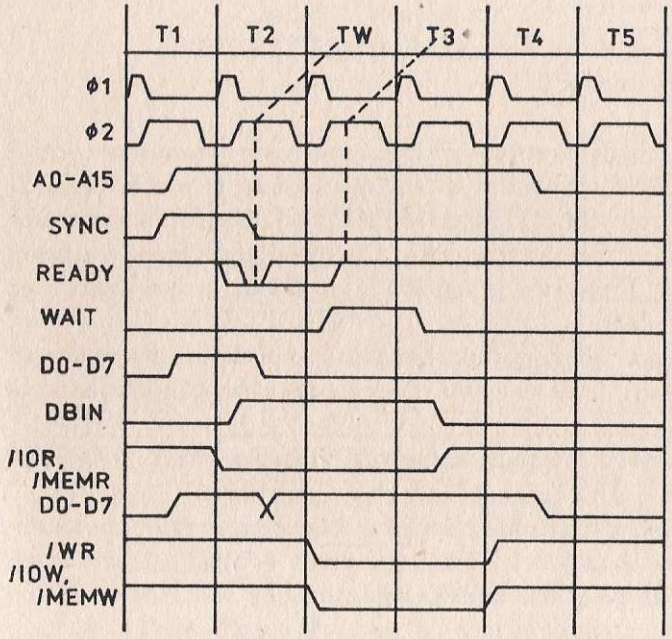


# 7. Mikropočítač — cvičení

## 7.1. ZÁKLADNÍ MĚŘENÍ NA MIKROPOČÍTAČI TEMS 80-03

### 7.1.1. Úvod

V této úloze si ukážeme průběhy hodinových a řídicích signálů na základních obvodech mikropočítače TEMS 80-03. Tyto signály budeme měřit na mikroprocesoru MHB8080A, řadiči MH8228 a generátoru hodinových signálů MH8224. Ideálním měřicím přístrojem pro tuto úlohu je logický analyzátor, na kterém si můžeme všechny signály zobrazit současně. V případě, že tento přístroj není k dispozici, vystačíme i s dvoukanálovým osciloskopem a přípravkem na krokování strojových cyklů mikroprocesoru, který bude popsán dále.




*Obr. 11.* Časové průběhy hodinových řídicích signálů mikropočítače

Průběhy základních hodinových a řídicích signálů jsou zobrazeny na obr. 11. Frekvence signálů Φ1 a Φ2 je 2 048 MHz. Každá instrukce mikroprocesoru MHB8080A je složena ze strojových cyklů M1, M2, ... Během každého tohoto cyklu se provede jeden zápis/čtení z/do operační paměti nebo periferního zařízení. Každý tento cyklus je složen z hodinových taktů (dob) T1, T2, ... Na začátku každého cyklu, v době T1, je generován mikroprocesorem signál SYNC, ze kterého po logickém součinu s Φ1 v MH8224 vznikne signál /STSTB. Ten je přiveden na vstup obvodu MH8228 a definuje časový interval, kdy na datové sběrnici mezi mikroprocesorem MHB8080A a řadičem MH8228 je řídicí slovo. Obsah tohoto slova definuje činnost, která bude vykonávána v daném strojovém cyklu, tj. zda půjde o čtení nebo zápis a zda to bude komunikace s operační pamětí nebo periferním zařízením. Další průběh řídicích signálů je už závislý na typu vykonávané operace.

V následujícím výkladu budou značeny logické hodnoty tímto způsobem:

L – úroveň napětí 0 až 0,8 V (logická nula),

H – úroveň napětí 2 až 5 V (logická jednička).

### 7.1.2. Přípravek pro krokování strojových cyklů mikroprocesoru

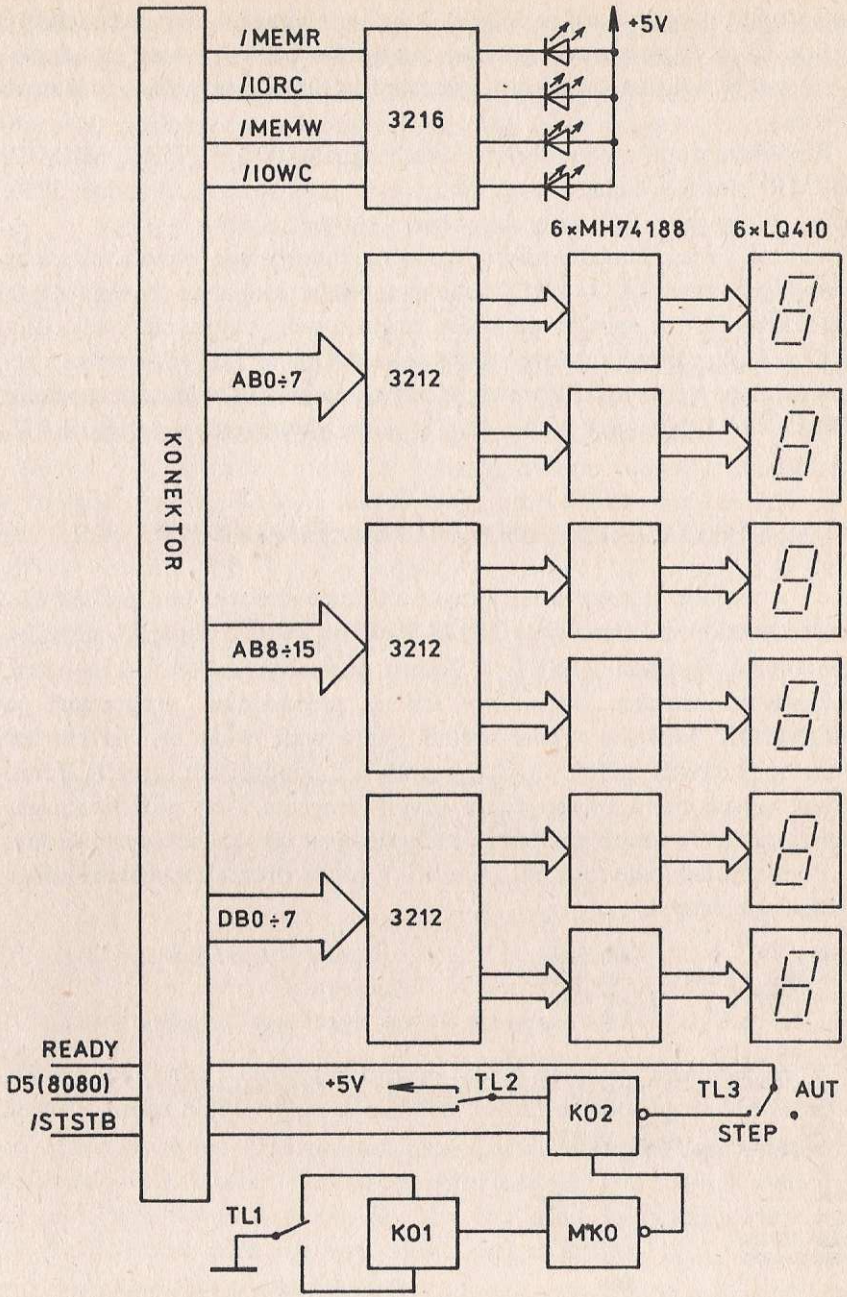
Princip krokování strojových cyklů spočívá ve využití stavu WAIT mikroprocesoru 8080, vyvolaném úrovní logické nuly na vodiči READY. Je-li na začátku strojového cyklu signál READY = L, přejde mikroprocesor do stavu WAIT a na sběrnicích se objeví signály odpovídající příslušnému strojovému cyklu. Tento stav je stabilní až do změny úrovně signálu na READY = H.

Pokud se zabezpečí, aby na začátku každého strojového cyklu byla úroveň signálu READY = L, bude pracovat mikroprocesor po jednotlivých strojových cyklech.

Přitom tu okolnost, že signály na sběrnicích jsou ve stavu WAIT stabilní, lze vhodně využít k jejich jednoduchému zobrazení.

Na obr. 12 je blokové schéma přípravku. Přípravek se připojuje na mikropočítač pomocí konektoru FRB s devadesáti kontakty. Konektorem jsou přiváděny datové a adresové signály na obvody 3212, které oddělují systémové sběrnice mikropočítače od pevných pamětí MH74188. Tyto paměti jsou naprogramovány tak, aby převáděly čtyřbitová dvojková čísla






*Obr. 12.* Blokové schéma přípravku krokování mikroprocesoru

na vstupu do kódu displeje LQ410, který je zobrazuje jako šestnáctkové číslice. Čtyři displeje slouží pro zobrazení informace přítomné na adresových vodičích a dva slouží pro zobrazení informace přítomné na datové sběrnici.

K zobrazení stavu jednotlivých řídicích signálů (IOWR, IORC, MEMW, MEMR) slouží světelné emisní diody, které jsou spínány obvodem 3216, jehož vstupy jsou připojeny na sběrnici řídicích signálů.

Vlastní řízení činnosti mikroprocesoru zabezpečuje obvod sestavený z klopných obvodů KO1 a KO2 a monostabilního klopného obvodu MKO. Tlačítkem TL1 se spouští jednotlivé strojové nebo instrukční cykly. Tlačítkem TL2 se přepíná režim krokování a tlačítkem TL3 se krokovací režim vyřazuje z činnosti. Přípravek se připojí na konektor J4 mikropočítače TEMS 80-03. Napájecí vodiče se připojí na vhodný zdroj napětí +5 V.

### 7.1.3. Programy pro sledování časových průběhů

Pomocí popsaného přípravku můžeme sledovat buď průběh činnosti vlastního mikropočítače TEMS 80-03 po zapnutí napájení, nebo po vynulování signálem RESET. V těchto případech začíná mikropočítač vykonávat instrukce od adresy 0H a prováděným programem je MONITOR. Můžeme si také sestavit jednoduché programy, na kterých budeme provádět měření časových průběhů jednotlivých signálů. Tento druhý způsob má tu výhodu, že lze vytvořit program, který probíhá v uzavřeném cyklu. To umožňuje řídicí signály sledovat také pomocí osciloskopu.

První příklad bude čtení dat z paměti. V tomto programu probíhá pouze čtení dat z paměti.

```
2000                    ORG 2000H
2000 3E55       CT1:    MVI A,55H
2002 C30020             JMP CT1
                        END
```

V následujícím příkladu bude využito instrukce zápisu do paměti. Všimněte si, že při použití řadiče MH8228 je vždy v cyklu zápisu alespoň jeden takt $T_w$ (doba WAIT).

```
2000                    ORG 2000H
2000 3E55       CT2:    MVI A,55H
2002 325020             STA 2050H
2005 C30020             JMP CT2
                        END
```



Obdobně můžeme sestavit krátké programy, kde je použito čtení nebo zápis do periferního zařízení. U všech těchto programů pomocí osciloskopu prověřte závislost mezi jednotlivými řídicími signály a pomocí popsaného přípravku posloupnost jednotlivých činností ve strojových cyklech.

## 7.2. VSTUP A VÝSTUP DAT S OBVODEM MHB8255A

### 7.2.1. Úvod

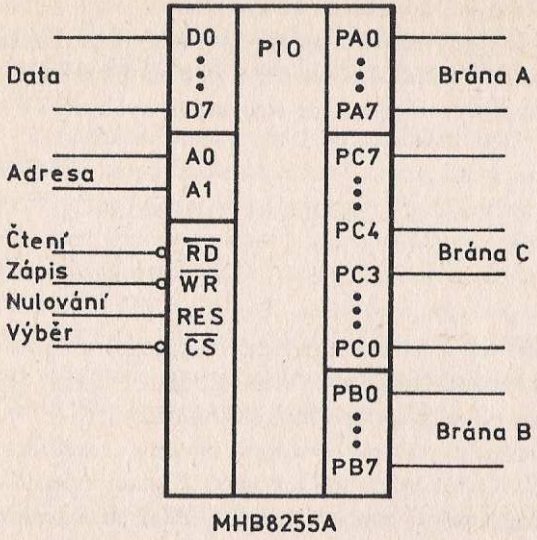
Pro paralelní přenos dat mezi mikropočítačem a periferním zařízením je výhodné používat obvod typu 8255 (MHB8255A). Na obr. 13 je schematická značka tohoto obvodu spolu s funkční tabulkou.

Obvod 8255 obsahuje paralelní vstupní/výstupní obvody, rozdělené na tři *brány (porty)* A, B, C. Každá brána má 8 vodičů. Funkci jednotlivých vodičů brány lze naprogramovat zápisem *řídicího slova* do řídicího registru obvodu 8255. Význam jednotlivých bitů v řídicím slově je následující:

Bit D0 – brána C, bity 0 až 3 jsou: 1 = vstup
0 = výstup
D1 – brána B je: 1 = vstup
0 = výstup
D2 – výběr režimu brány B: 0 = režim 0
1 = režim 1
D3, D4 – výběr režimu brány A: 00 = režim 0
01 = režim 1
1X = režim 2
D5 – brána A je: 1 = vstup
0 = výstup
D6 – brána C, bity 4 až 7 jsou: 1 = vstup
0 = výstup
D7 – 1 = příznak řídicího slova nastavení režimu

V *režimu* 0 obvod 8255 provádí jednoduché vstupní/výstupní operace přes každou ze tří bran. Je možné využít následující možnosti obvodu:

- dvě osmibitové brány (A, B) a dvě čtyřbitové brány (C);
- každá brána může být buď vstupní, nebo výstupní;
- brány pro výstup obsahují registry;




MHB8255A

| A1 | A0 | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | Vstupní operace (čtení) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | Brána A → data |
| 0 | 1 | 0 | 1 | 0 | Brána B → data |
| 1 | 0 | 0 | 1 | 0 | Brána C → data |
| | | | | | Výstupní operace (zápis) |
| 0 | 0 | 1 | 0 | 0 | Data → brána A |
| 0 | 1 | 1 | 0 | 0 | Data → brána B |
| 1 | 0 | 1 | 0 | 0 | Data → brána C |
| 1 | 1 | 1 | 0 | 0 | Data → řízení |

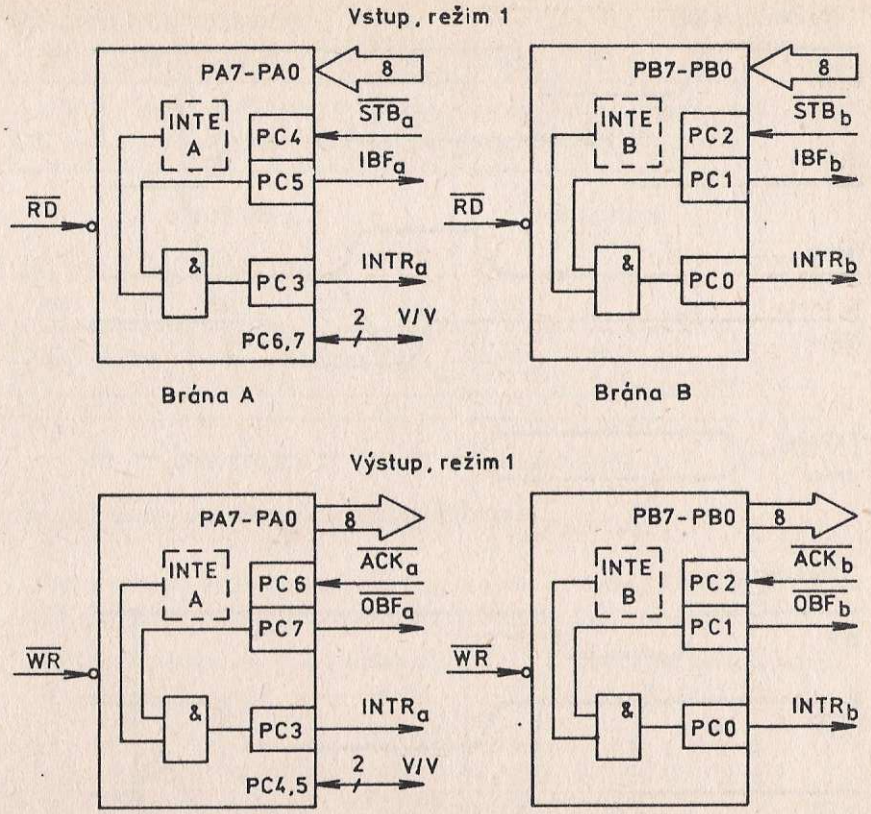
*Obr. 13.* Programovatelný obvod MHB8255A

– brány pro vstup registry neobsahují;
– je možné 16 různých konfigurací vstupu/výstupu.

V *režimu 1* je obvodem 8255 realizován přenos dat do/z dané brány v konfiguraci s řídicími signály pro řízení styku s periferním zařízením. Brány A a B slouží k přenosu dat a vodiče brány C slouží k řízení přenosu. Je možné využívat následující možnosti obvodu:

– dvě množiny signálů (A, B);
– každá množina obsahuje osmibitovou datovou bránu a čtyři řídicí signály;





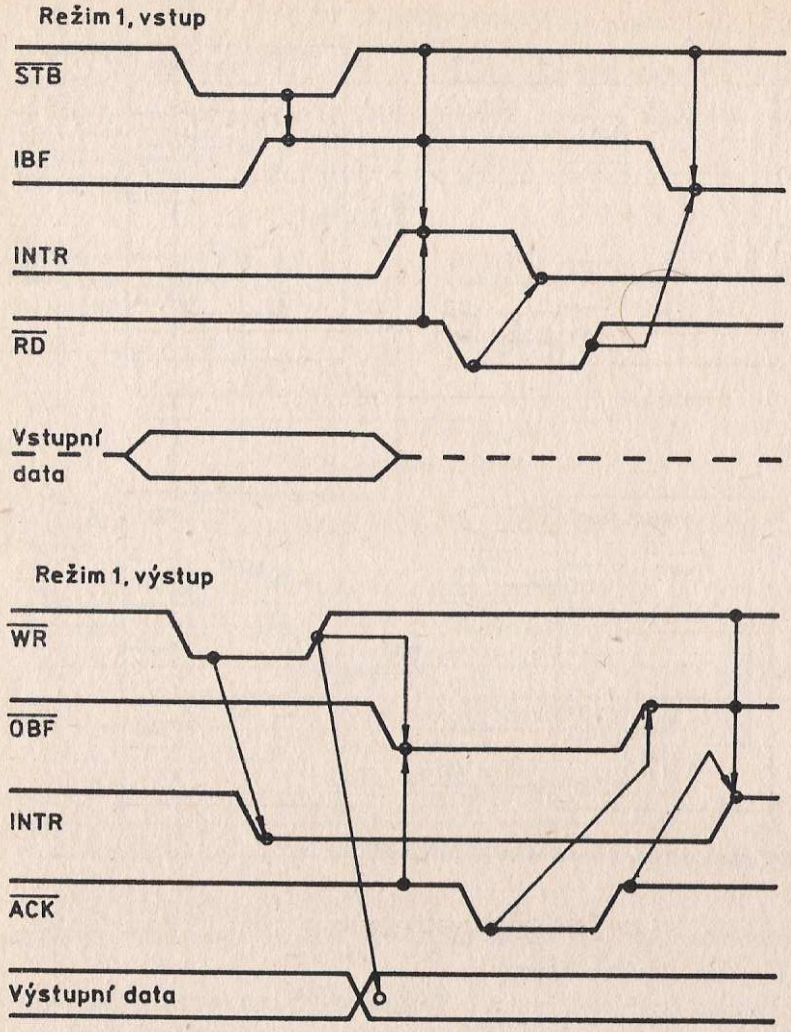

*Obr. 14.* Funkce signálů obvodu MHB8255A v režimu 1

– osmibitová datová brána může být buď vstupní, nebo výstupní, vstup i výstup obsahuje registry;
– příslušná čtyřbitová brána je použita pro řízení přenosu osmibitové datové brány.

Na obr. 14 je vyznačena funkce jednotlivých signálů v režimu 1. Jednotlivé signály mají tento význam:

STB – přechod tohoto signálu z H do L zapíše vstupní data do vstupního registru brány;

IBF – úroveň H tohoto signálu indikuje, že byla data zapsána do vstupního registru. Signál IBF je nastaven na úroveň H přechodem signálu STB do stavu L a nulován náběžnou hranou vstupu /RD;



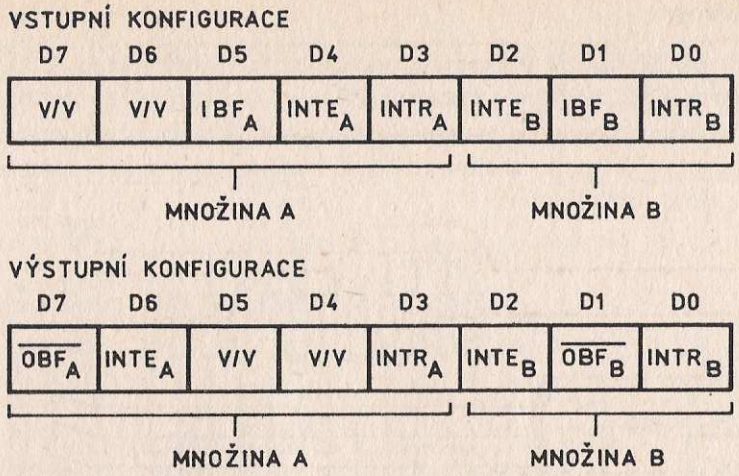

*Obr. 15.* Průběh řídicích signálů obvodu MHB8255A v režimu 1

INTR – tento signál slouží k řízení žádosti o přerušení;

/OBF – tento signál je nastaven na úroveň H náběžnou hranou signálu /WR a nulován signálem ACK = L;

/ACK – úroveň L tohoto signálu informuje, že připojené zařízení si výstupní data převzalo.






*Obr. 16.* Formát stavového slova obvodu MHB8255A

Pro uvolnění systému žádosti o přerušení je nutné nastavit hodnotu INTE pro daný režim a bránu do stavu H, a to tímto způsobem:

| Datová brána | Vstup/výstup | INTE ovládáno bitem |
|---|---|---|
| A | vstup | PC4 |
| A | výstup | PC6 |
| B | vstup | PC2 |
| B | výstup | PC2 |

Průběh řídicích signálů v režimu 1 je na obr. 15. Hodnoty bitů brány C můžeme měnit také pomocí řídicího slova nastavení/nulování bitu C. Jednotlivé bity tohoto slova mají tento význam:

D0 – nastavení/nulování bitu 1 = nastavení
0 = nulování

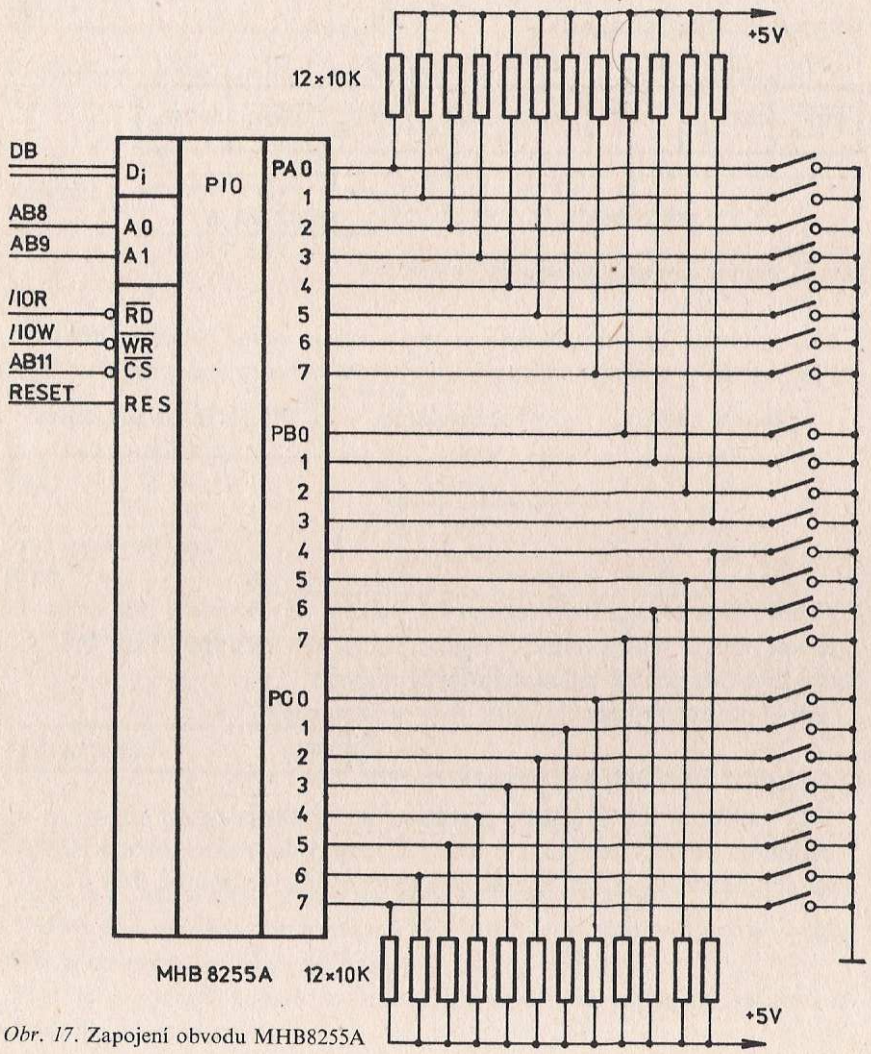
D1 až D3 – adresa bitu brány C

D7 – příznak řídicího slova pro nulování/nastavení = 0

Hodnota čtená v režimu 1 z brány C odpovídá stavu obvodu 8255A. Význam jednotlivých bitů tohoto přečteného slova je na obr. 16.

### 7.2.2. Zapojení

Obvod MHB8255A zapojíme podle obr. 17. Vodiče bran A, B, C můžeme použít jako vstupní i výstupní. Při jejich použití k výstupu jsou příslušné spínače rozpojeny. U vstupní brány pomocí spínačů nastavujeme vstupní hodnotu. Výstupní hodnoty měříme buď pomocí logické sondy, voltmetrem nebo osciloskopem.




*Obr. 17.* Zapojení obvodu MHB8255A



### 7.2.3. Programy

Následující program nastaví obvod MHB82255A do režimu 0 a všechny brány budou výstupní. Výstupní hodnoty jsou uloženy na adresách 2050H až 2052H.

```
                ;  mod 0,vystup
2000                    ORG 2000H
2000 3E80       ZAC:    MVI A,80H       ;ridici slovo
2002 D3F7               OUT CNTRL       ;zapis ho do 8255A
2004 3A5020             LDA AHOD        ;hodnota brany A
2007 D3F4               OUT PORTA
2009 3A5120             LDA BHOD        ;hodnota brany B
200C D3F5               OUT PORTB
200E 3A5220             LDA CHOD        ;hodnota brany C
2011 D3F6               OUT PORTC
2013 76                 HLT
                ;
2050                    ORG 2050H
2050 55         AHOD:   DB 55H          ;hodnota do brany A
2051 AA         BHOD:   DB 0AAH         ;hodnota do brany B
2052 33         CHOD:   DB 33H          ;hodnota do brany C
00F4            PORTA   EQU 0F4H
00F5            PORTB   EQU 0F5H
00F6            PORTC   EQU 0F6H
00F7            CNTRL   EQU 0F7H
                ;
                        END
```

V dalším programu použijeme obvod 8255 pro vstup dat z bran A a B. Hodnoty budou nastavovány spínači. Přečtené hodnoty zobrazíme na displeji mikropočítače TEMS 80-03, v jeho adresové části. Nejprve zobrazíme hodnotu brány A s příznakem „A" a po dvou sekundách hodnotu brány B s příznakem „B" atd.

```
                ;  mod 0,vstup
2000                    ORG 2000H
2000 3E9B               MVI A,9BH       ;ridici slovo
2002 D3F7               OUT CNTRL       ;zapis ho do 8255A
2004 DBF4       ZOBR:   IN PORTA        ;precti branu A
2006 5F                 MOV E,A         ;presun do reg E
2007 16A0               MVI D,0A0H      ;priznak brany A
2009 CD1302             CALL AKTDS      ;zobraz
200C 110000             LXI D,0         ;cekej 2 s
200F CD1802             CALL DELAY
2012 CD1802             CALL DELAY
2015 DBF5               IN PORTB        ;precti branu B
2017 5F                 MOV E,A         ;presun do reg E
2018 16B0               MVI D,0B0H      ;priznak brany B
201A CD1302             CALL AKTDS      ;zobraz
201D 110000             LXI D,0         ;cekej 2 s
2020 CD1802             CALL DELAY
2023 CD1802             CALL DELAY
2026 C30420             JMP ZOBR        ;pokracuj branou A
                ;
00F4            PORTA   EQU 0F4H
00F5            PORTB   EQU 0F5H
```

```
00F6            PORTC   EQU 0F6H
00F7            CNTRL   EQU 0F7H
0213            AKTDS   EQU 213H
0218            DELAY   EQU 218H
                ;
                        END
```

Dané programy lze modifikovat pro čtení brány C i pro práci v režimu 1 a řídicí signály simulovat spínači. Potom odezvu na řídicí signály STB, /ACK sledujeme na příslušných výstupech IBF, /OBF, INTR.

## 7.3. PŘIPOJENÍ KLÁVESNICE K MIKROPOČÍTAČI

### 7.3.1. Úvod

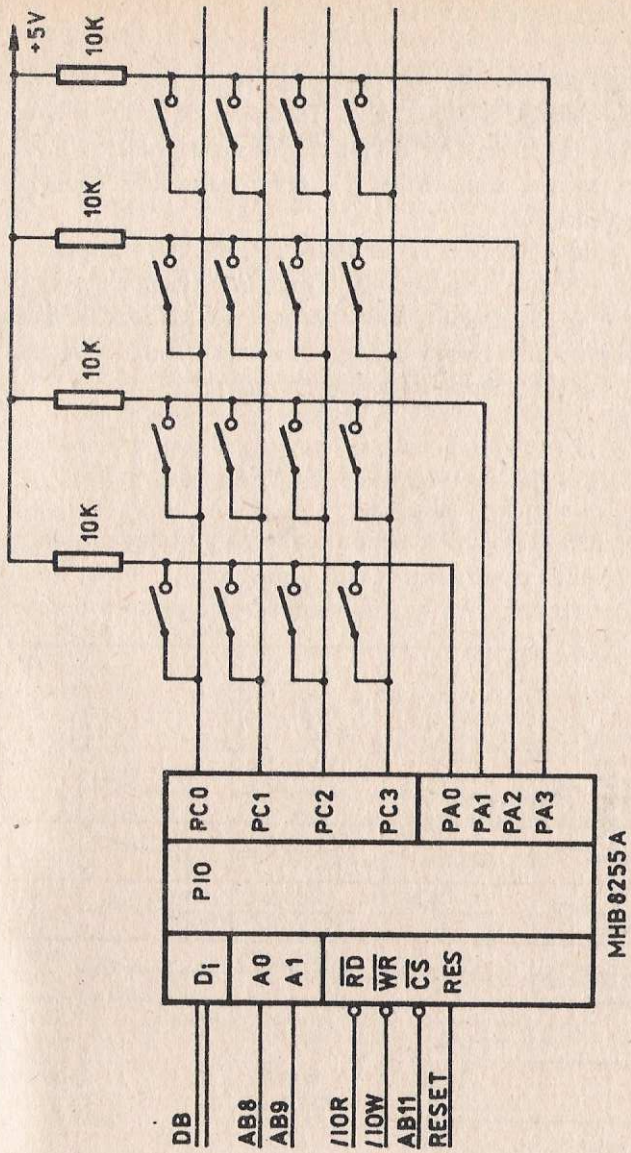
Klávesnice je jedním ze základních periferních zařízení, se kterými může mikropočítač spolupracovat. Je složena z jednotlivých tlačítek a každému z nich je přiřazen určitý význam. Stisknutím tlačítka zadáváme do mikropočítače vybranou informaci. Klávesnice může být realizována několika způsoby, z nichž vybereme dvě následující možnosti:

– klávesnice je pouze množinou tlačítek bez dekódovacích obvodů a detekce typu stisknutého tlačítka je realizována pomocí programu v mikropočítači;

– klávesnice obsahuje kódovací obvody a obvody styku s mikropočítačem, který potom pouze čte kód stisknuté klávesy. Obslužný program je krátký.

Samotná klávesnice je chápána mikropočítačem jako matice tlačítek o $m$ řádcích a $n$ sloupcích. K systému mikropočítače je potom každý řádek nebo sloupec připojen jedním vodičem. V takto uspořádané klávesnici je maximální počet tlačítek roven hodnotě $m \times n$. Aby součet počtu řádků a sloupců připojených k mikropočítači byl minimální, je výhodné použít čtvercového uspořádání matice tlačítek. Pokud klávesnici vybavíme elektronikou pro vyhodnocení stisknutého tlačítka a každému tlačítku přiřadíme určitou dvojkovou hodnotu, zmenší se značně počet vodičů, kterými je klávesnice připojena k mikropočítači. Jestliže máme např. klávesnici se 64 tlačítky, je při čtvercovém uspořádání třeba $8 + 8 = 16$ vodičů k připojení klávesnice bez elektroniky. Pokud ovšem obsahuje elektroniku pro dvojkové kódování tlačítek, je třeba pouze 6 vodičů pro připojení k mikropočítači, neboť $2^6 = 64$.






*Obr. 18.* Připojení klávesnice k obvodu MHB8255A

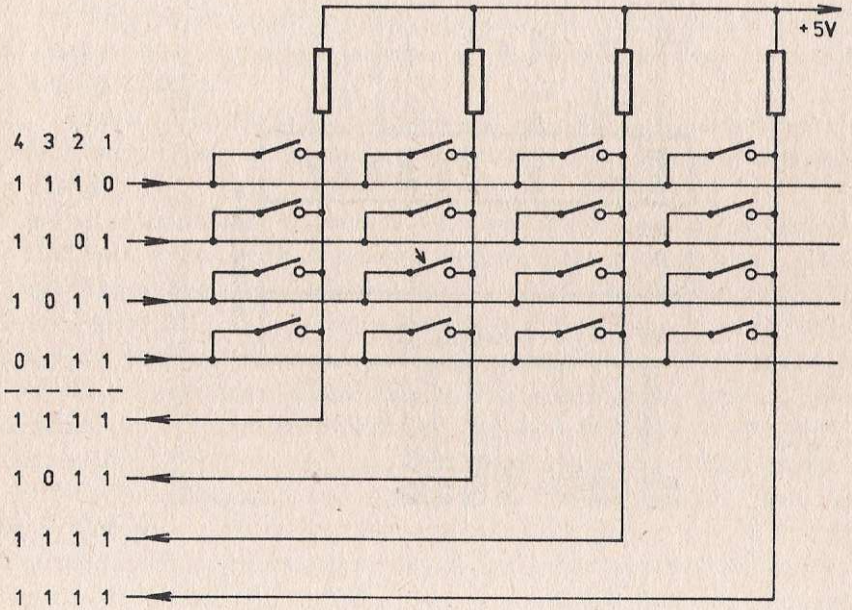
### 7.3.2. Připojení samotné klávesnice

Klávesnici můžeme k mikropočítači připojit pomocí programovatelného paralelního obvodu MHB 8255A. Tento obvod je ke sběrnici připojen datovými vodiči D0 až D7, dvěma adresovými vodiči A0, A1, řídicími signály pro čtení a zápis /IOR a /IOW, nulovacím signálem RESET a signálem výběru /CS.

Příklad připojení samotné klávesnice se šestnácti tlačítky (4 sloupce × × 4 řádky) k mikropočítači je na obr. 18. Tímto způsobem lze připojit klávesnici o rozměru až 8 sloupců × 8 řádků = 64 tlačítek. Obvod MHB8255A bude naprogramován do režimu 0, brána C bude výstupní, brána A vstupní, tzn. že řídicí slovo bude mít hodnotu 92H.

### 7.3.3. Program obsluhy samotné klávesnice

Jestliže máme klávesnici připojenou podle obr. 18, lze pro dekódování stisknutého tlačítka použít algoritmus s postupující nulou. Tento algoritmus je naznačen na obr. 19 a spočívá v tom, že na jednotlivé řádky




*Obr. 19.* Zjišťování stisknutého tlačítka programem



postupně vysíláme vzorky dat, kde všechny bity kromě jednoho mají hodnotu 1. Přitom čteme hodnoty vodičů sloupců. Je-li stisknuto některé tlačítko, získáme v jednom bitu příslušného sloupce hodnotu 0.

V následujícím programu je ukázána konkrétní obsluha obvodu MHB8255A a dekódování stisknutého tlačítka. Tento program se skládá ze dvou částí:

– hlavního programu, začíná návěštím HLPR1;
– podprogramu pro vlastní čtení klávesnice – CTIKL.

V hlavním programu se nejprve zapíše řídicí slovo 92H do obvodu MHB8255A. Potom se cyklicky postupně volá podprogram CTIKL a testuje příznak CY, zda bylo stisknuto tlačítko klávesnice. Pokud ano, zobrazí se jeho kód na displeji. Pokud ne, displej se vymaže.

Podprogram CTIKL provádí vlastní testování matice klávesnice. Velikost klávesnice lze nastavit pomocí proměnných POCET (počet tlačítek) a SLOUP (počet sloupců). Pokud jsme našli stisknuté tlačítko, potom příznak CY = 1 a registr D obsahuje kód tlačítka, jinak příznak CY = 0 a registr D je nulový.

```
                ; cyklicke cteni klavesnice
2000                    ORG 2000H
2000 3E92       HLPR1:  MVI A,92H       ;nastaveni obvodu 8255A
2002 D3F7               OUT CNTRL       ;A,B vstup, C vystup
2004 CD8020     CTI:    CALL CTIKL      ;cti klavesnici
2007 0E02               MVI C,2         ;priprav zobrazeni v poli dat
2009 D21220             JNC NIC         ;kdyz zadny znak
200C CD1302             CALL AKTDS      ;zobraz precteny znak
200F C30420             JMP CTI         ;opakuj cteni klavesnice
2012 211920     NIC:    LXI H,NILL      ;priprav adresu prazdnych znaku
2015 E7                 RST 4           ;vymaz displej
2016 C30420             JMP CTI         ;cti znovu klavesnici
                ;
2019 14         NILL:   DB 14H,14H      ;prazdne znaky
201A 14
00F7            CNTRL   EQU 0F7H
0213            AKTDS   EQU 213H
                ;
2080                    ORG 2080H
2080 26FE       CTIKL:  MVI H,0FEH      ;pocatecni vzorek dat radku
2082 1610               MVI D,POCET     ;pocet tlacitek do pocitadla
2084 7C         DLSRD:  MOV A,H         ;vzorek dat radku
2085 D3F6               OUT PORTC       ;zapis do radku
2087 07                 RLC             ;priprav pristi vzorek
2088 67                 MOV H,A         ;a uloz si ho
2089 DBF4               IN PORTA        ;cti sloupce
208B 0E04               MVI C,SLOUP     ;pocet sloupcu
208D 0F         DLSL:   RRC             ;test na nulu v danem sloupci
208E D29E20             JNC SLP         ;skok pokud je nula
2091 15                 DCR D           ;sniz pocet tlacitek
2092 0D                 DCR C           ;uz vsechny sloupce?
2093 C28D20             JNZ DLSL        ;testuj dalsi sloupec,zbyva-li
2096 79                 MOV A,C         ;zjisti zda jeste zbyvaji radky
2097 B7                 ORA A
```

```
2098 C28420          JNZ DLSRD       ;jdi na dalsi radek,zbyva-li
209B 37              STC
209C 3F              CMC             ;priznak nestisknuteho tlacitka
209D C9              RET             ;navrat
209E 37      SLP:    STC             ;priznak stisknuteho tlacitka
209F C9              RET             ;navrat
             ;
0010         POCET   EQU 16          ;pocet tlacitek
0004         SLOUP   EQU 4           ;pocet sloupcu
00F4         PORTA   EQU 0F4H        ;adresa brany A
00F6         PORTC   EQU 0F6H        ;adresa brany C
             ;
                     END
```

Takto sestavený program se pro praktické aplikace obvykle nehodí, neboť je jím mikropočítač plně zaměstnán. Většinou potřebujeme, aby se na základě stisknutí tlačítka vykonala daná činnost. K tomu potřebujeme mít jistotu, že dané tlačítko bylo stisknuto. Toto lze programově zajistit tak, že při testování klávesnice dáme podmínku, že dané tlačítko považujeme za stisknuté, když bylo stisknuto po určitou minimální dobu. V následujícím programu se nejprve testuje, zda po dobu minimálně 32 vzorkování není stisknuto žádné tlačítko klávesnice. Potom se testuje, zda po dobu minimálně 32 vzorkování je naopak nějaké tlačítko stisknuto. Pokud je toto splněno, považujeme dané tlačítko za stisknuté.

```
             ;  vstup znaku z klavesnice
2020                 ORG 2020H
2020 3E92    HLPR2:  MVI A,92H       ;nastaveni obvodu 8255A
2022 D3F7            OUT CNTRL       ;A,B vstup, C vystup
2024 0620    KMIT1:  MVI B,20H       ;uloz pocet vzorku
2026 CD8020  VZRK1:  CALL CTIKL      ;cti klavesnici
2029 DA2420          JC KMIT1        ;jeste nejake tlacitko stisknute?
202C 05              DCR B           ;ne, vzorkuj dal
202D C22620          JNZ VZRK1       ;pokud nejsi hotov
2030 0620    KMIT2:  MVI B,20H       ;nastav pocitadlo vzorku
2032 CD8020  VZRK2:  CALL CTIKL      ;cti klavesnici
2035 C23020          JNZ KMIT2       ;cekej na priznak
2038 05              DCR B           ;mame uz 32 krat priznak za sebou?
2039 C23220          JNZ VZRK2       ;vzorkuj dal kdyz ne
203C CF              RST 1           ;navrat do monitoru
             ;
00F7         CNTRL   EQU 0F7H
2080         CTIKL   EQU 2080H       ; stejne jako u predchoziho programu
             ;
                     END
```

Dále můžeme program pro obsluhu klávesnice upravit tak, aby při stisknutí libovolného tlačítka vykonal požadovanou funkci, např. rozsvícení nápisu na displeji.



### 7.3.4. Klávesnice mikropočítače TEMS 80-03

Vestavěná klávesnice mikropočítače TEMS 80-03 je příkladem klávesnice s obslužnými elektronickými obvody. U této klávesnice je příznak čtení znaku generován v obslužných obvodech.

Součástí monitoru je podprogram pro čtení kódu stisknutého tlačítka klávesy. Název tohoto programu je RDKBD. Podprogram RDKBD čeká nejprve ve smyčce na příznak stisknutého tlačítka. Z kódu přečteného znaku se odstraní příznak stisknutí tlačítka a uloží se do zásobníku. Nyní se čeká na ukončení impulsu STB, kterým je definována čekací doba uklidnění stisknutého tlačítka. Následuje vyzvednutí kódu znaku ze zásobníku a návrat do hlavního programu. Pro potlačení vlivu kmitání stisknutého tlačítka je použit monostabilní klopný obvod, který generuje při stisknutí impuls dlouhý 15 ms.

```
0010                 ORG 10H
0010 DB3C    RDKBD:  IN KPORT        ;cteni klavesnice
0012 17              RAL             ;posuv vlevo
0013 D21600          JNC RDKB1       ;skok na pokracovani
0016 3F      RDKB1:  CMC             ;ano,nuluj priznak
0017 1F              RAR             ;obnov znak
0018 F5              PUSH PSW        ;a uloz ho
0019 DB3C    RDKB2:  IN KPORT        ;cekej na konec STB
001B 17              RAL
001C DA1900          JC RDKB2
001F F1              POP PSW         ;vyzvedni znak
0020 C9              RET             ;navrat
             ;
003C         KPORT   EQU 3CH
             ;
                     END
```

Podprogram RDKBD můžeme používat v programech obdobně jako CTIKL. Volat jej můžeme buď instrukcí CALL RDKBD, nebo pomocí instrukce RST 2.

## 7.4. PŘIPOJENÍ SEDMISEGMENTOVÉHO DISPLEJE

### 7.4.1. Úvod

Jedním z možných způsobů komunikace mikropočítače s obsluhou je zobrazení stavů pomocí světelných emisních diod (LED), které mohou být samostatné nebo součástí sedmisegmentového displeje. Dioda svítí, pokud jí prochází proud v přímém směru o velikosti $I_{AK} = 0{,}5$ mA

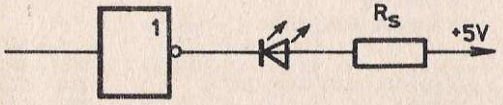
až 30 mA. Úbytek napětí na diodě bývá $U_{LED}$ = 1,6 V až 3 V. Konkrétní hodnoty pro každý typ světelné emisní diody je nutné zjistit v katalogu výrobce.

### 7.4.2. Zapojení

Obvyklé připojení světelné emisní diody (LED) k výstupu logického členu je na obr. 20. Při tomto zapojení se dioda rozsvítí, je-li na výstupu logického členu úroveň L. Hodnotu odporu určíme následujícím způsobem:

$$R_s = \frac{U_{CC} - U_{LED} - U_L}{I_{LED}}$$

kde $U_{CC}$ je napětí zdroje (+5 V),
$U_{LED}$ – napětí světelné emisní diody v přímém směru,
$U_L$ – výstupní napětí logického členu na úrovni L,
$I_{LED}$ – zvolený proud procházející světelnou emisní diodou.




*Obr. 20.* Připojení světelné emisní diody k logickému členu TTL

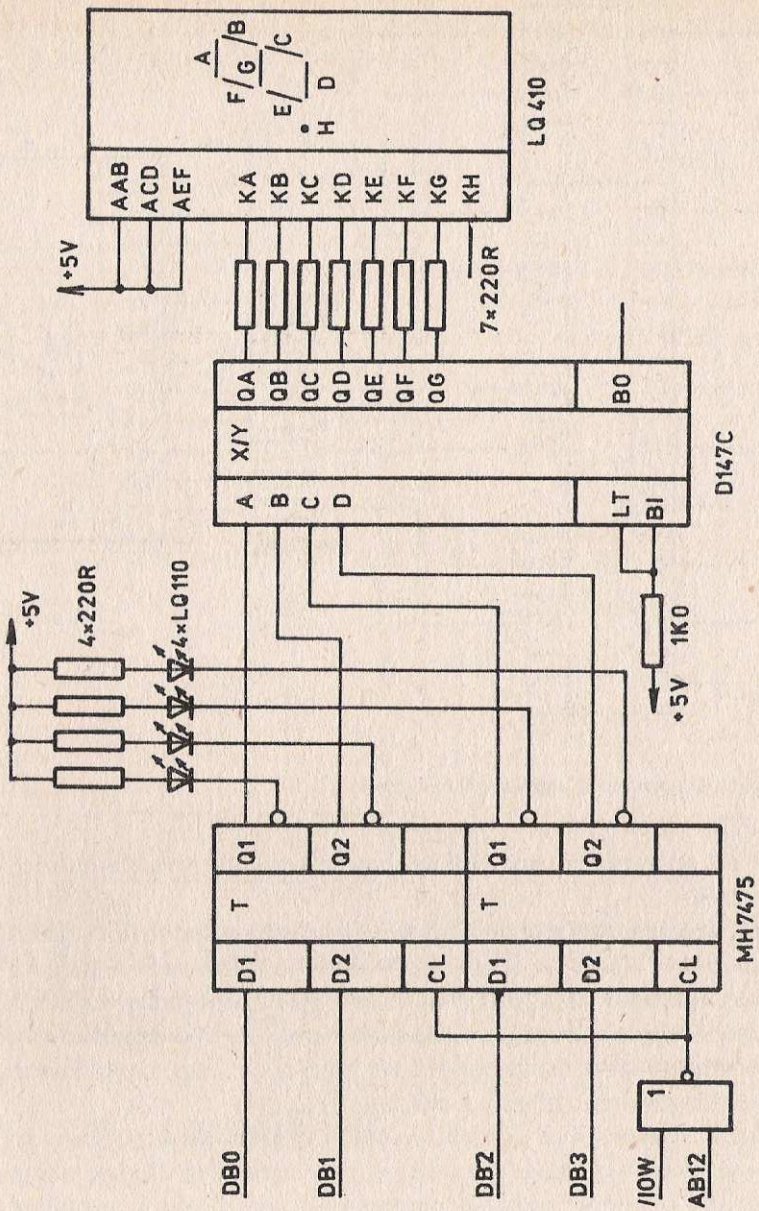
Při volbě proudu $I_{LED}$ musíme vzít v úvahu zatížitelnost výstupu daného budicího logického členu ve stavu L podle katalogu výrobce. Na obr. 21 je konkrétní zapojení čtyř světelných emisních diod a jednoho sedmisegmentového displeje. Pro zapamatování zobrazované veličiny je použit obvod MH7475. Pro dekódování hodnoty zobrazované na displeji je použit převodník D147C z kódu BCD na sedmisegmentový kód. Světelnými emisními diodami prochází proud, jehož maximální velikost zvolíme vzhledem k zatížení výstupů MH7475 16 mA. Z této hodnoty určíme i velikost odporů. Platí

$$R = \frac{U_{CC} - U_{LED} - U_L}{I_{LED}} = \frac{5 - 1{,}6 - 0{,}5}{0{,}016} = 181\ \Omega$$

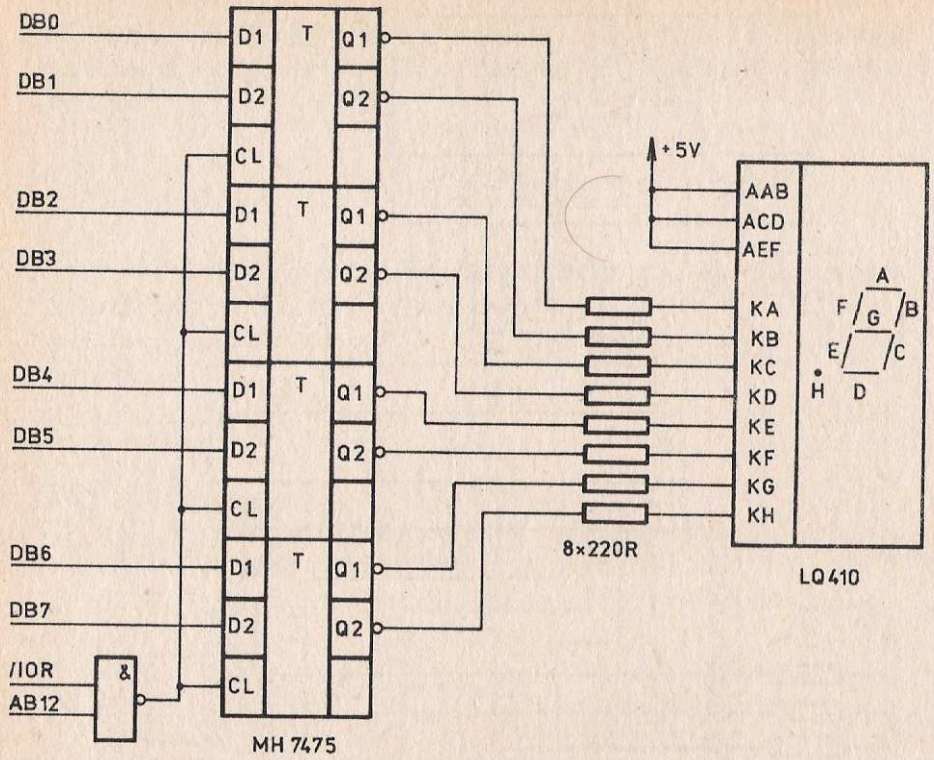
Nejbližší hodnota odporu je 220 Ω.

Vstupy LT (light test) a BI (blanking in) převodníku D147C jsou připojeny na úroveň H, takže nejsou aktivní. Úroveň L na vstupu LT způsobí rozsvícení všech segmentů nezávisle na vstupech A až D. Vstup BI






*Obr. 21.* Zapojení sedmisegmentového displeje s převodníkem



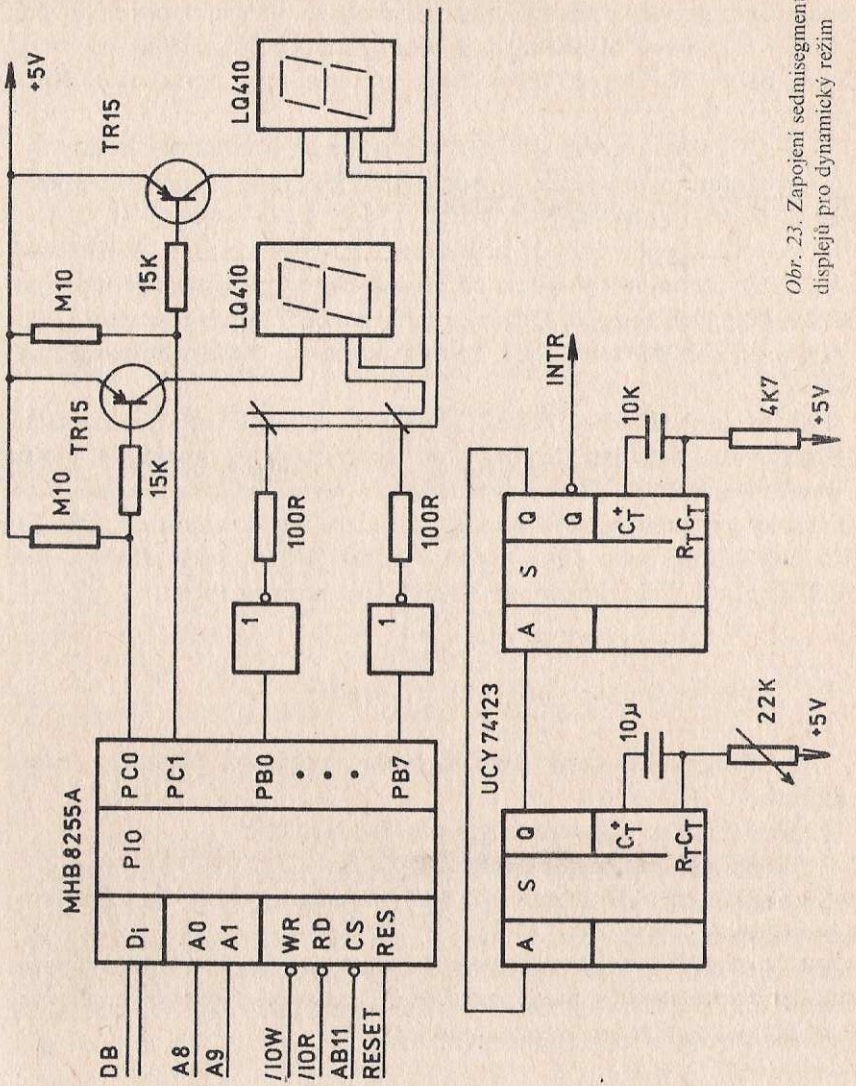

*Obr. 22.* Přímé připojení sedmisegmentového displeje

a výstup B0 se používají pro potlačení zobrazení nul na začátku vícemístného čísla.

Pomocí daného převodníku zobrazíme číslice 0 až 9 a šest dalších znaků. Sedmisegmentový displej je schopen zobrazit více informací – šestnáctkové číslice, abecedu a další znaky. Chceme-li plně využit možnosti sedmisegmentového displeje, použijeme zapojení z obr. 22. Pro zapamatování zobrazovaného stavu je nutný osmibitový registr, do kterého zapisujeme hodnoty podle převodní tabulky z obr. 24.

Máme-li k dispozici více displejů, je často výhodné použít *dynamický režim zobrazování*. Využívá se setrvačnosti oka. Jednotlivé displeje nesvítí stále, ale jsou vhodnou rychlostí přepínány a rozsvěcovány postupně. Toto přepínání lze realizovat buď programově, nebo obvodovými prostředky. Schéma zapojení obvodu přepínání jednotlivých displejů pomocí






*Obr. 23.* Zapojení sedmisegmentových displejů pro dynamický režim

programového vybavení je na obr. 23. Pro zapamatování zobrazované informace a zapnutí daného displeje se použil obvod MHB8255A. Pro generování intervalu přepnutí je využito monostabilních klopných obvodů UCY74123. Obvod MHB8255A je naprogramován do režimu 0 pro výstupní bránu B. Obvod MHB8255A můžeme nyní obsluhovat těmito způsoby:

– programově generujeme interval přepínání (programový čítač);

– programově testujeme obvod MHB8255A, zda již došlo k překlopení připojených obvodů UCY74123;

– povolení přijetí žádosti o přerušení mikroprocesorem MHB8080A a okamžik přerušení odvodíme od příslušného výstupu dvojice monostabilních klopných obvodů. Displeje přepneme při obsluze přerušení.

Třetí způsob má tu výhodu, že mikroprocesor může vykonávat další činnosti.

Příklad obvodového řešení přepínání najdeme v mikropočítači TEMS 80-03. Sedmisegmentový kód zobrazovaných znaků je uložen v pomocné paměti RAM (6 slov po 8 bitech), ale z hlediska mikroprocesoru se displej jeví jako paměť šesti slov, do které lze jen zapisovat. Displeje jsou adresovány jako části paměti. Adresy jednotlivých displejů jsou 3C00H (pravá krajní pozice) až 3C05H (levá krajní pozice).

### 7.4.3. Programy pro obsluhu displejů

Program obsluhy displejů může vykonávat některé z těchto činností:

1. převod dat na kód sedmisegmentového displeje;
2. zaslání dat do registru displeje;
3. cyklické provádění činnosti 1 a 2 pro jednotlivé displeje v dynamickém režimu.

Použijeme-li zapojení podle obr. 21, stačí, aby program zajistil pouze činnost 2. Následující program zobrazí postupně v jednosekundových intervalech všech 16 znaků převodníku D147C.

```
            ;  zobrazeni
2000                ORG 2000H
2000 AF             XRA A           ;pocatecni vynulovani reg A
2001 D3EC   UKAZ:   OUT MH75        ;zapis znak do MH7475
2003 3C             INR A           ;priprav dalsi znak
2004 F5             PUSH PSW        ;uloz si ho do zasobniku
2005 110000         LXI D,0
```



```
2008 CD1802         CALL DELAY      ;cekej 1 s
2008 F1             POP PSW         ;vyzvedni znak ze zasobniku
200C C30120         JMP UKAZ        ;a zobraz ho
            ;
00EC        MH75    EQU 0ECH
0218        DELAY   EQU 218H
            ;
                    END
```

V praxi se může vyskytnout potřeba zobrazit dané hodnoty na požádání. Můžeme si to ukázat na následujícím programu, kde se zobrazí nižší 4 bity paměťového místa s adresou DATA na dobu 1 s po stisknutí tlačítka INT.

```
            ;  zobrazeni po INT
2000                ORG 2000H
2000 FB             EI              ;povoleni prijeti zadosti o preruseni
2001 00     SEM:    NOP             ;simulace hlavniho programu
2002 C30120         JMP SEM         ;cekaci smyckou
            ;
2005 05     DATA:   DB 5            ;zobrazovany znak
            ;
2010                ORG 2010H
2010 F5     DISP:   PUSH PSW        ;uschovej obsah A a F
2011 3A0520         LDA DATA        ;vezmi znak
2014 D3EC           OUT MH75        ;a zobraz ho
2016 110000         LXI D,0         ;pockej 1 s
2019 CD1802         CALL DELAY
201C 3EFF           MVI A,0FFH
201E D3EC           OUT MH75        ;zhasni segmentovku
2020 F1             POP PSW         ;obnov obsah A a F
2021 FB             EI              ;povol nove preruseni
2022 C9             RET             ;navrat do hlavniho programu
            ;
23CC                ORG 23CCH
23CC C31020         JMP DISP        ;skok pri preruseni
            ;
00EC        MH75    EQU 0ECH
0218        DELAY   EQU 218H
            ;
                    END
```

Není-li displej vybaven převodníkem a je zapojen podle obr. 22, je nutné převod na kód displeje realizovat programově. K tomuto účelu výhodně využijeme převodní tabulku na obr. 24. Následující program ukazuje zobrazení zvoleného alfanumerického znaku, který jsme zadali z klávesnice TEMS 80-03 šestnáctkovou hodnotou v kódu ASCII. Převod mezi kódem ASCII a tvarem zobrazovaného znaku se provede pomocí tabulky. Tabulka je realizována pomocí pole, kde každý prvek má délku jednu slabiku (byte). Hodnoty, které jsou uloženy v jednotlivých prvcích pole, jsou kódy jednotlivých znaků pro sedmisegmentový displej. Indexy prvků pole jsou hodnoty jednotlivých znaků v kódu ASCII, tzn. že jde o tabulku s přímým přístupem.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 39 | 3A |
| 3F | 06 | 5B | 4F | 66 | 6D | 7D | 07 | 7F | 6F |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A |
| 77 | 7C | 39 | 5E | 79 | 71 | 3D | 76 | 04 | 1E |
| A | b | C | d | E | F | G | H | I | J |

| K | L | M | N | O | P | Q | R | S | T |
|---|---|---|---|---|---|---|---|---|---|
| 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 |
| 70 | 38 | 37 | 54 | 5C | 73 | 67 | 50 | 6D | 78 |
| K | L | M | n | o | P | q | r | S | t |

| U | V | W | X | Y | Z | ZNAK |
|---|---|---|---|---|---|---|
| 55 | 56 | 57 | 58 | 59 | 5A | KÓD ASCII |
| 1C | 62 | 3E | 64 | 6E | 5B | KÓD DISPLEJE |
| U | V | U | X | Y | Z | ZOBRAZENÍ |

*Obr. 24.* Převodní tabulka

Pokuste se vytvořit program pro ovládání zobrazované informace podle zapojení na obr. 23. Využijte jednu z uvedených možností řízení přepínání jednotlivých displejů.



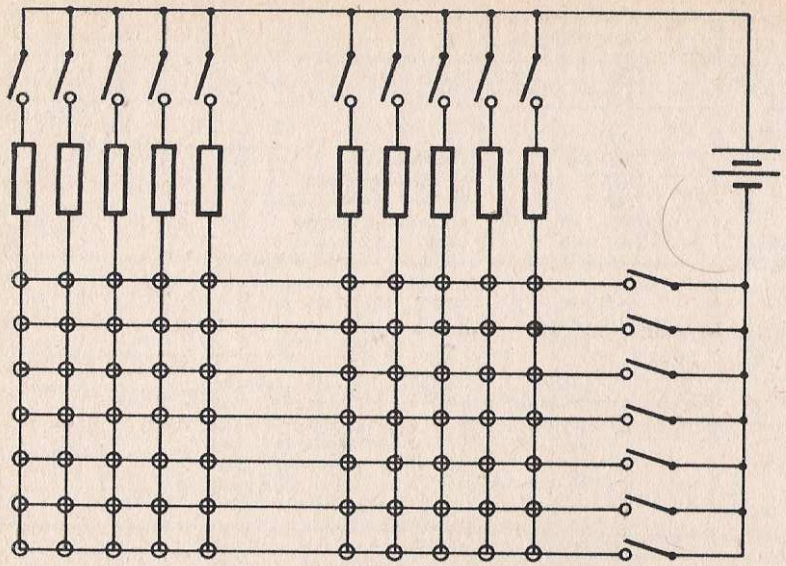
```
                ; zobrazeni alfanumerickych znaku
2000                    ORG 2000H
2000 D7         START:  RST 2           ;cti klavesnici
2001 07                 RLC             ;posun znak do horni
2002 07                 RLC             ;poloviny registru
2003 07                 RLC
2004 07                 RLC
2005 47                 MOV B,A         ;uschovej ho
2006 D7                 RST 2           ;cti druhy znak
2007 B0                 ORA B           ;pripoj prvni znak
2008 6F                 MOV L,A         ;relativni adresa
2009 2600               MVI H,0         ;kodu v tabulce
200B 110021             LXI D,ZACTAB    ;zacatek prevodni tabulky
200E 19                 DAD D           ;absolutni adresa
200F 7E                 MOV A,M         ;vezmi kod z tabulky
2010 D3EC               OUT MH75
2012 C30020             JMP START       ;jdi pro dalsi znak
                ;
2100                    ORG 2100H       ;zacatek prevodni tabulky
2100 00         ZACTAB: DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2108 00                 DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2110 00                 DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2118 00                 DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2120 00                 DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2128 00                 DB 0,0,0,0,0,0,0,0   ; netisknutelny znak
2130 3F                 DB 3FH,6,5BH,4FH,66H,6DH,7DH,7       ; 0 az 7
2138 7F                 DB 7FH,6FH,0,0,0,0,0,0               ; 8,9
2140 00                 DB 0,77H,7CH,39H,5EH,79H,71H,3DH     ;znaky
2148 76                 DB 76H,4,1EH,70H,38H,37H,54H,5CH     ;abecedy
2150 73                 DB 73H,67H,50H,6DH,78H,1CH,62H,3EH
2158 64                 DB 64H,6EH,5BH,0,0,0,0,0
00EC            MH75    EQU 0ECH
                ;
                        END
```

## 7.5. GENEROVÁNÍ ZNAKŮ POMOCÍ MATICE BODŮ

### 7.5.1. Úvod

Displej pro zobrazení alfanumerických znaků můžeme také vytvořit z matice bodů, např. 7 × 5 nebo 7 × 9 bodů. Každý bod této matice je tvořen jednou světelnou emisní diodou. Pro zobrazování na těchto displejích se používá *multiplexní (dynamický) režim*, tzn. že světelné emisní diody, které tvoří daný znak, nesvítí všechny najednou, ale postupně se rozsvěcují diody v jednotlivých řádcích (*vertikální rozklad*) nebo v jednotlivých sloupcích (*horizontální rozklad*).

Na obr. 25 je základní schéma zapojení takovéhoto displeje. Pokud k řízení zobrazení použijeme horizontální rozklad, zapneme v prvním taktu spínač prvního sloupce prvního znaku a podle zobrazovaného znaku příslušné spínače řádků. Ve druhém taktu tyto spínače vypneme a naopak



*Obr. 25.* Základní zapojení bodového displeje

zapneme spínač druhého sloupce prvního znaku a příslušné spínače řádků. Takto postupujeme pro všechny sloupce všech znaků displeje. Jedna dioda bude rozsvícena $1/s$ doby jedné obsluhy celého displeje ($s$ je celkový počet sloupců displeje).

Při vertikálním řízení zobrazení v prvním taktu sepneme spínač prvního řádku a příslušné spínače sloupců všech znaků displeje. Ve druhém taktu vypneme tyto spínače a pokračujeme druhým řádkem atd. U tohoto způsobu je doba, po kterou je rozsvícena jedna dioda, rovna $1/r$ doby jedné obsluhy daného displeje ($r$ je celkový počet řádků displeje).

Vzhledem k dynamickému režimu displeje je třeba zvýšit střední proud procházející diodou na hodnotu $I_{av}$

$$I_{max} = sI_{av} \qquad \text{popř.} \qquad I_{max} = rI_{av}$$

Displej s pěti znaky s maticí 7 × 5 bodů a mezním proudem 100 mA procházejícím diodou bude mít tyto hodnoty $I_{av}$:
horizontální rozklad

$$I_{av} = \frac{I_{max}}{s} = \frac{100}{5 \,.\, 5} = 4 \text{ mA}$$

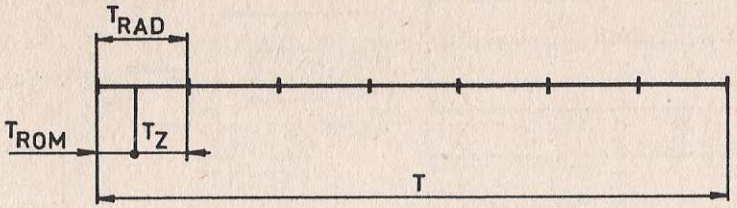


vertikální rozklad

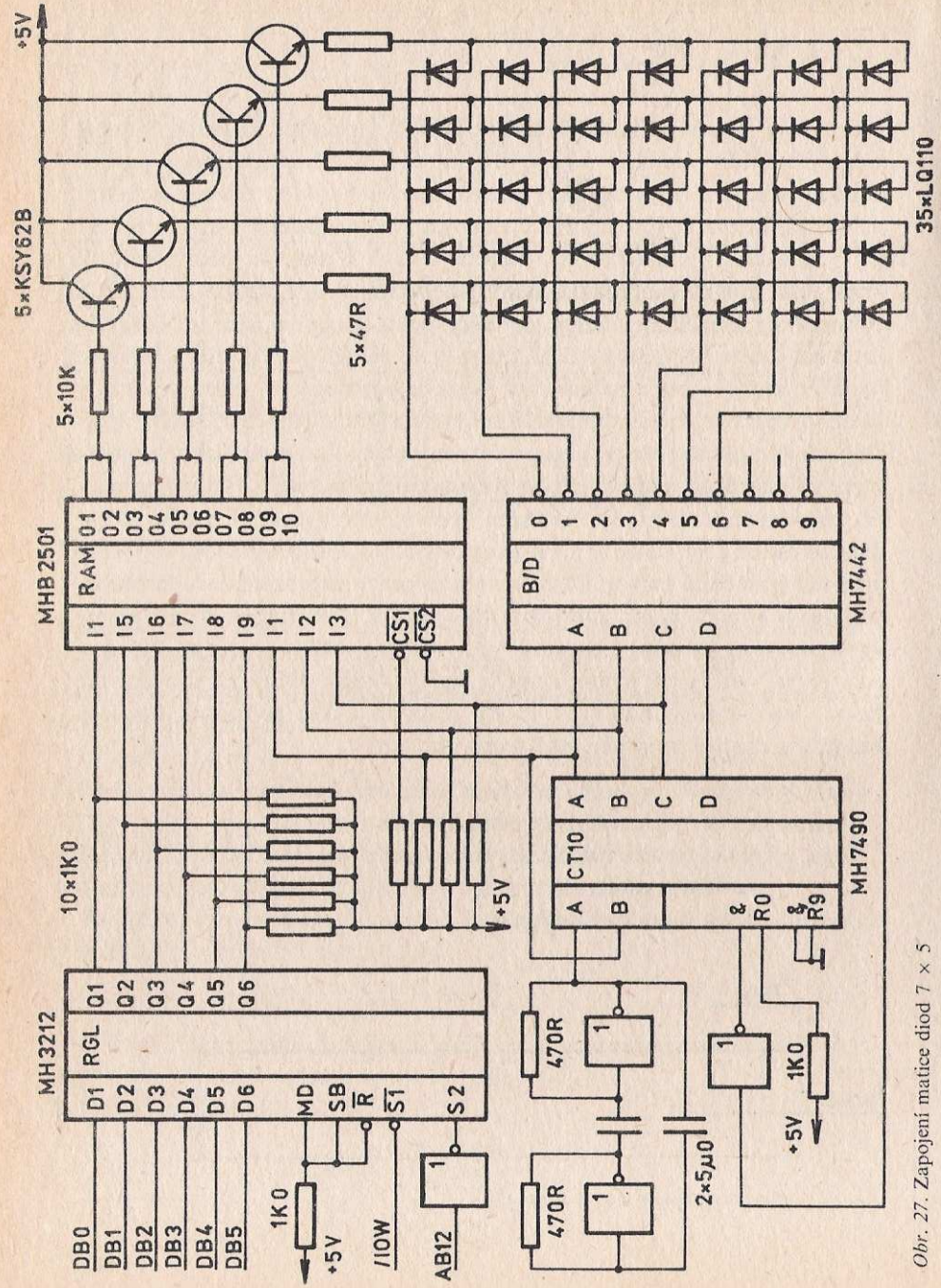
$$I_{av} = \frac{I_{max}}{r} = \frac{100}{7} = 14 \text{ mA}$$

Pro větší počet znaků je výhodné použít vertikální rozklad. Činnost spínačů lze řídit pomocí znakové paměti typu ROM, např. integrovaného obvodu MHB2501. Tato paměť má standardně 6 vstupů definujících kód ASCII znaku (vstupy I4 až I9) a 3 vstupy definující řádek znaku (vstupy I1 až I3, 8 řádků). Kapacita této paměti je 64 × 8 slov po pěti bitech, tj. 2 560 bitů. Pro zapamatování řádku jednoho znaku je použita pětibitová vyrovnávací paměť. Zobrazení jednoho řádku se potom provádí následujícím způsobem: Na vstupy I1 až I3 znakové paměti se přivede číslo řádku (0 až 7), který se bude zobrazovat. Nyní se postupně přivedou na vstupy I4 až I9 znakové paměti kódy jednotlivých zobrazovaných znaků. Na výstupech této paměti budou potom informace o daném řádku těchto znaků, které se zapíší do pětibitové vyrovnávací paměti znaků. To se provádí při zhasnutém displeji. Po naplnění vyrovnávacích pamětí zapne spínač daného řádku a příslušné spínače sloupců podle zapsané hodnoty ve vyrovnávacích pamětech znaků. Potom začne příprava zobrazení dalšího řádku, tzn. že se vypne spínač zobrazovaného řádku a postupuje se od plnění vyrovnávacích pamětí, postupně pro všechny řádky displeje. Nyní si odvodíme výraz pro střídu impulsu každé diody. Obrázek 26 ukazuje časové rozložení jednotlivých kroků jednoho cyklu zobrazení displeje. Jednotlivé časové intervaly mají tento význam:

$T$ je perioda; $T = 1/f_s$, kde $f_s$ je snímková frekvence,
$T_{RAD}$ doba vyhrazená pro jeden řádek,
$T_{ROM}$ doba pro přepis výstupů znakové paměti do vyrovnávacích pamětí znaků,
$T_Z$ doba rozsvícení řádku.




*Obr. 26.* Časové rozložení ovládání displeje




*Obr. 27.* Zapojení matice diod 7 × 5

Nyní určíme velikost jednotlivých časových intervalů. Platí

$$T_{ROM} = Nt_{ROM}$$

kde $N$ je počet znaků displeje,

$t_{ROM}$ vybavovací doba znakové paměti a zápis do vyrovnávací paměti znaku

$$T_{RAD} = T/7$$
$$T_Z = T_{RAD} - T_{ROM} = T/7 - Nt_{ROM} = (T - 7Nt_{ROM})/7$$

Poměr doby rozsvícení jednoho řádku a doby vyhrazené pro obsluhu jednoho řádku je dán vztahem

$$d = T_Z/T = (1 - 7f_sNt_{ROM})/7$$

Maximální proud procházející řádkovými spínači je dán vztahem

$$I_{RAD} = 5NI_{max}$$

### 7.5.2. Zapojení

Pro danou úlohu se spokojíme s jednou maticí složenou ze $7 \times 5$ světelných emisních diod. Celkové schéma zapojení je na obr. 27. Výběr jednoho ze sedmi řádků displeje se provádí pomocí oscilátoru řádkové frekvence, dvojkového čítače a dekodéru. Nejnižší tři bity čítače (A, B, C) představují adresu řádku. Spolu se šesti bity kódu znaku tvoří adresu paměti znaků pro výběr potřebného pětibitového slova pro rozsvícení diod v adresovaném řádku. Úroveň H na výstupu paměti znaků odpovídá rozsvícení diody. Vlastní buzení diod je provedeno tranzistory NPN.

### 7.5.3. Programy

První program zobrazuje všechny znaky obsažené v paměti znaků MHB2501 s kódem 0 až 3FH. Každý znak se zobrazuje po dobu 2 s.

```
                ;  sekvence znaku MHB2501
2000                    ORG 2000H
2000 AF                 XRA A           ;pocatecni stav registru A
2001 D3EF       ZOBR:   OUT MH12        ;zapis do MH3212 a zobraz
2003 F5                 PUSH PSW        ;uklid si kod zobr. znaku
2004 110000             LXI D,0         ;cekaci konstanta
2007 CD1802             CALL DELAY      ;cekej 1 s
```

```
200A CD1802          CALL DELAY      ;cekej 1 s
200D F1              POP PSW         ;kod zobrazovaneho znaku
200E 3C              INR A           ;kod nasledujiciho znaku
200F C30120          JMP ZOBR        ;a zobraz ho
                     ;
00EF            MH12    EQU 0EFH
0218            DELAY   EQU 218H
                ;
                        END
```

Nyní si ukážeme program obdobný programu z čl. 7.4. Jde o zobrazení znaku, který je zadán pomocí klávesnice mikropočítače. Znak zadáváme pomocí šestnáctkového čísla, které odpovídá hodnotě znaku v kódu ASCII.

```
                ;  zobrazeni kodu ASCII
2000                    ORG 2000H
2000 CD5020     ASC:    CALL GETB       ;precti zadanou hodnotu
2003 D3EF               OUT MH12        ;zobraz ji
2005 C30020             JMP ASC
                ;
2050                    ORG 2050H
                ;podprogram pro nacteni hodnoty
2050 D7         GETB:   RST 2           ;cti klavesnici
2051 07                 RLC             ;posun o 4 bity
2052 07                 RLC
2053 07                 RLC
2054 07                 RLC
2055 47                 MOV B,A         ;uloz si mezivysledek
2056 D7                 RST 2           ;cti klavesnici
2057 B0                 ORA B           ;vysledna hodnota
2058 C9                 RET
                ;
00EF            MH12    EQU 0EFH
                ;
                        END
```

V následujícím programu budeme zadané znaky nejen zobrazovat, ale i ukládat do zásobníku, abychom je mohli později opět zobrazit a případně některé z nich změnit.

```
                ;  ukladani zpravy
2000                    ORG 2000H
2000 310022             LXI SP,2200H    ;nastav ukazatel zasobniku
2003 CD5020     UKL:    CALL GETB       ;cti znak
2006 D3EF               OUT MH12        ;zobraz ho
2008 F5                 PUSH PSW        ;a uloz do zasobniku
2009 33                 INX SP          ;posun ukazatel
200A C30320             JMP UKL         ;dalsi znak
                ;
00EF            MH12    EQU 0EFH
2050            GETB    EQU 2050H
                ;
                        END
```

Poslední hodnotou, kterou zadáme, bude hodnota 0. Podle ní v následujícím programu pro opakované zobrazení znaků ze zásobníku poznáme konec zprávy. Jednotlivé znaky se budou zobrazovat po dobu 2 s.



```
                ;  zobrazeni zpravy
201F                    ORG 201FH
201F FB                 EI              ;povoleni prijeti zadosti o preruseni
2020 210022             LXI H,2200H     ;zacatek zpravy
2023 2B         VYBER:  DCX H
2024 7E                 MOV A,M
2025 B7                 ORA A           ;zda uz neni konec zpravy
2026 CA3720             JZ KONEC
2029 D3EF               OUT MH12        ;zobraz znak
202B 110000             LXI D,0         ;cekej 2 s
202E CD1802             CALL DELAY
2031 CD1802             CALL DELAY
2034 C32320             JMP VYBER       ;jdi pro dalsi znak
2037 3E20       KONEC:  MVI A,20H       ;zhasni displej zobrazenim
2039 D3EF               OUT MH12        ;mezery
203B 76                 HLT
                ;
00EF            MH12    EQU 0EFH
0218            DELAY   EQU 218H
                ;
                        END
```

Je-li v zobrazované zprávě chyba, je možné v době, kdy se na displeji zobrazuje chybný znak, stisknout tlačítko INT a tento znak opravit. Stisknutí tlačítka INT vyvolá přerušení úrovně 7, tzn. že se provede instrukce RST 7, během níž se skočí na adresu 38H. Tato adresa je součástí monitoru a je na ní instrukce skoku na adresu 23CCH.

```
                ;  oprava ulozeneho textu
2040                    ORG 2040H
2040 F5         EDIT:   PUSH PSW        ;uloz obsah reg A
2041 CD5020             CALL GETB       ;nacti novy kod znaku
2044 77                 MOV M,A         ;uloz ho na misto zobrazovaneho
2045 D3EF               OUT MH12        ;a zobraz ho
2047 F1                 POP PSW         ;obnov reg A
2048 FB                 EI              ;povoleni prijeti zadosti o preruseni
2049 C9                 RET             ;a navrat
                ;
23CC                    ORG 23CCH
23CC C34020             JMP EDIT
                ;
2050            GETB    EQU 2050H
00EF            MH12    EQU 0EFH
                ;
                        END
```

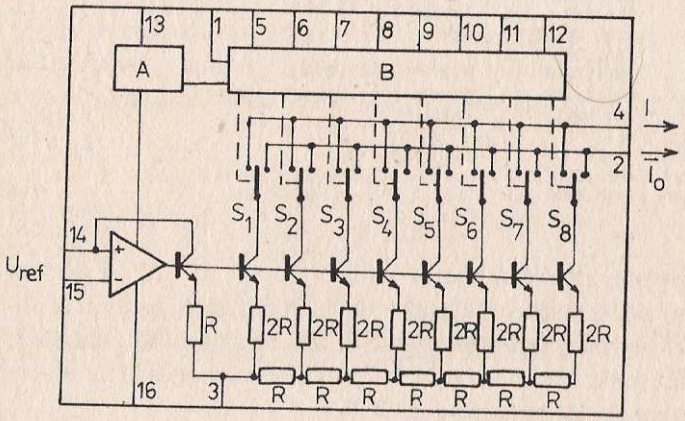
## 7.6. GENERÁTOR ANALOGOVÝCH SIGNÁLŮ

### 7.6.1. Teoretický rozbor

Při řízení různých procesů pomocí mikropočítače se často vyskytuje požadavek na analogové zobrazení výsledných veličin, např. ve formě napětí. Tento problém se obvykle řeší připojením číslicově analogového

(D/A) převodníku, který vytváří výstupní analogový signál ze vstupního referenčního napětí v závislosti na dvojkovém čísle na svém vstupu.

Vstupní referenční napětí může být konstantní nebo i proměnné (pak mluvíme o násobicím převodníku D/A, neboť vstupní analogová hodnota je násobena hodnotou číslicovou).



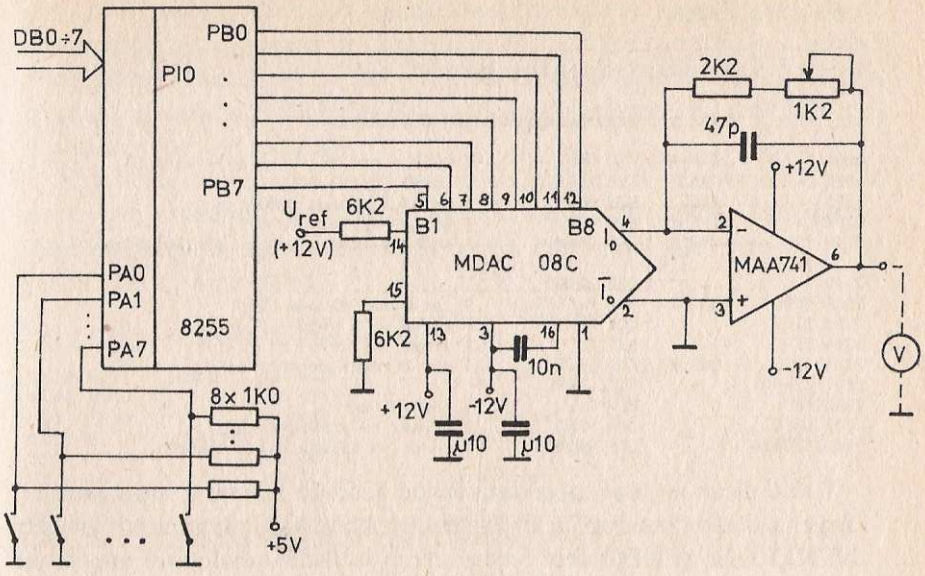

*Obr. 28.* Vnitřní zapojení číslicově analogového převodníku MDAC 08

Pro měření bude použito konstantní referenční napětí, jako převodník D/A využijeme obvod TESLA MDAC 08C – osmibitový převodník D/A, jehož blokové schéma je na obr. 28. Převodník je třeba doplnit vnějším zdrojem referenčního napětí a výstupním zesilovačem.

Základní částí převodníku D/A jsou přepínané tranzistorové spínače kalibrovaných, binárně dělených proudů, odvozených příčkovým článkem $R/2R$ ze zdroje referenčního napětí pomocí zpětné vazby přes referenční tranzistor. Dílčí proudy velikosti $I_{ref}/2^i$ jsou připojovány tranzistorovými spínači na přímou (výstup 2) nebo komplementární (výstup 4) proudovou sběrnici. Výstupní hodnoty proudu se tedy mohou pohybovat v rozsahu 0 až $255*I_{ref}/256$ s krokem $I_{ref}/256$. Pro optimální činnost převodníku doporučuje výrobce hodnotu $I_{ref} = 2$ mA. Kompenzační kondenzátor (zapojený mezi svorky 3 a 16) s kapacitou 10 nF má zabránit rozkmitání vnitřního operačního zesilovače, dále se doporučuje připojit svorky vstupního napětí přes kondenzátor s kapacitou 0,1 μF na zem.

Výstupní proud na přímém (svorka 2) nebo komplementárním výstupu (svorka 4) lze na napětí převést buď pasívně (pomocí rezistoru), nebo aktivně (pomocí operačního zesilovače se zpětnovazebním rezistorem).






*Obr. 29.* Zapojení pro měření převodu D/A

## 7.6.2. Zapojení

Převodník D/A připojíme k datové sběrnici mikroprocesoru přes bránu B programovatelného paralelního obvodu MHB8255 v režimu 0. Jako zdroj referenčního proudu (s optimální hodnotou 2 mA) můžeme zvolit zdroj napětí +12 V spojený s rezistorem o odporu 6,2 kΩ nebo zdroj napětí +5 V spojený s rezistorem o odporu 2,7 kΩ. Stejný rezistor připojíme i na druhou svorku referenčního napětí. Vzhledem k tomu, že obvod 8255 má na datovém výstupu logické úrovně TTL, můžeme vstup 1 (nastavuje prahové napětí spínačů) uzemnit.

Výstupní proud převodníku D/A převedeme na napětí pomocí výstupního operačního zesilovače TESLA MAA 741. Použijeme kompenzační kondenzátor s kapacitou 47 pF mezi vstupní a výstupní svorkou, zpětnou vazbu vytvoříme z pevného rezistoru a potenciometru, kterým budeme regulovat vlastně převodní konstantu celého převodníku (schéma zapojení je na obr. 29).

Na bránu A připojíme 8 spínačů v pouzdře DIL, jejichž pomocí budeme ovlivňovat parametry průběhů. Na vstupní bránu A takto přivádíme buďto napětí +5 V (přes rezistor s odporem 1 kΩ), nebo ji uzemňujeme.

## 7.6.3. Měření

### Generování konstantního napětí

```
                ;       Generovani konstantniho napeti

0090            MOBIA   EQU 90H         ; 8255 - mod 0, vstup A, vystup B
00F7            PR55    EQU 0F7H        ; ridici brana 8255
00F5            PB55    EQU 0F5H        ; brana B obvodu 8255
00F4            PA55    EQU 0F4H        ; brana A obvodu 8255
0248            GTHEX   EQU 248H        ; vstup 2,popr. 4 znaku z klavesnice

2000                    ORG 2000H
2000 3E90       CONST:  MVI A,MOBIA     ; nastaveni obvodu 8255
2002 D3F7               OUT PR55        ; modus 0, A vstup, B vystup
2004 EF                 RST 5           ; vymaz displej
2005 0682       UROVEN: MVI B,82H       ; cti 2 znaky do dat. pole
2007 CD4802             CALL GTHEX
200A 7B                 MOV A,E         ; zadane znaky
200B D3F5               OUT PB55        ; predej na prevodnik D/A
200D C30520             JMP UROVEN      ; cekej na zadani dalsi hodnoty
```

V této úloze nejprve nastavíme obvod 8255 do režimu 0, smažeme celý displej a očekáváme zadání dvou znaků z klávesnice ukončených znakem (NEXT) nebo (EXEC), které nám určují výsledné analogové napětí, jež měříme voltmetrem. Celý převodník je možné „kalibrovat" pomocí zpětnovazebního potenciometru tak, aby pro maximální zadané číslo (OFFH) bylo výstupní napětí např. 5 V. Takto můžeme určit i přesnost převodu D/A – $U_{výst}/256$ = rozdíl mezi 2 sousedními hladinami (pro +5 V asi 19,5 mV).

### Generování signálu proměnné délky s pilovým průběhem

```
                ; Generovani piloveho prubehu

2010 3E90       PILA:   MVI A,MOBIA     ; nastav 8255 do modu 0
2012 D3F7               OUT PR55        ; A vstup, B vystup
2014 0600               MVI B,0
2016 78         KROK:   MOV A,B         ; citac prubehu piloveho napeti
2017 D3F5               OUT PB55        ; vysli na prevodnik D/A
2019 DBF4               IN PA55         ; nacti zadanou delku hladiny
201B 3D         CEKEJ:  DCR A           ;  z brany A
201C C21B20             JNZ CEKEJ       ; pozdrz na tuto dobu
201F 04                 INR B           ; zvys citac
2020 C31620             JMP KROK
```

V této úloze se pomocí předchozího zapojení generuje pilové napětí (s omezenou přesností, neboť diskrétní hodnoty hladin jsou dány přesností převodníku D/A). Délka těchto hladin je určována obsluhou (nastavením spínačů DIL). Snadno lze spočítat rozmezí periody pilového průběhu, které lze tímto způsobem generovat. Pro daný program je to přibližně 6,88 ms (pro $N = 1$) až 485 ms (pro $N = 0$, tj. 256 cyklů). Sestavte vzorec



pro určení periody v závislosti na nastavené konstantě, vypočtěte konstantu pro periodu 50 ms a přesnost výpočtu i celého průběhu kontrolujte s pomocí osciloskopu. Jaké největší frekvence lze dosáhnout zjednodušením programu (odstranění čekací smyčky) s danou diskrétností hladin? (320 Hz) Proveďte toto zjednodušení.

Navrhněte modifikaci programu, která zabezpečí generování signálu s pilovým průběhem v určeném intervalu napětí. Navrhněte dále úpravu zapojení, která by zabezpečila tutéž úlohu. Porovnejte obě metody, určete jejich výhody a nevýhody.

```
                ; Trojuhelnikove napeti

2023 78         KROK0:  MOV A,B         ; citac na prevodnik D/A
2024 D3F5               OUT PB55
2026 DBF4               IN PA55         ; nacti delku
2028 3D         CEKEJ0: DCR A           ; podrz hladinu
2029 C22820             JNZ CEKEJ0
202C 04                 INR B           ; zvys citac
202D C22320             JNZ KROK0       ; uz konec nabez. hrany ?
2030 05                 DCR B           ; zmensi na 0FFH
2031 78         KROK1:  MOV A,B         ; sestupna strana
2032 D3F5               OUT PB55        ; citac na prevodnik
2034 DBF4               IN PA55         ; delka hladiny
2036 3D         CEKEJ1: DCR A
2037 C23620             JNZ CEKEJ1      ; podrz ji
203A 05                 DCR B           ; sniz citac
203B C23120             JNZ KROK1       ; uz konec sestup. hrany ?
203E C32320             JMP KROK0       ; ano - cele znovu
```

Malou obměnou programu lze generovat *trojúhelníkové napětí*. Tělo generující procedury bude mít tuto podobu:

Přepište tuto úpravu do paměti a opět experimentálně ověřte rozmezí frekvencí průběhu. Zvažte, jak by se maximálně frekvence změnila, kdyby se signál generoval v každé druhé, čtvrté atd. úrovni. Jak se změní kvalita generovaného signálu?

### Generování signálů s obecnějším průběhem

V předchozích úlohách jsme generovali signály s jednoduchou časovou závislostí. Ovšem často se vyskytuje požadavek generovat průběhy mnohem složitější. Problém nastává při výpočtu hodnot dané funkční závislosti, čímž se značně zpomalí generování a klesá frekvence průběhu. Proto je nutné volit kompromis mezi přesností a rychlostí generovaného signálu. Aby nebylo nutné vypočítávat hodnoty v každém bodě, zvolíme pro vyjádření funkčních hodnot tabulku. Jako základní program pro generování periodických průběhů v $M$ bodech použijeme sekvenci TABSEK.

```
TABSEK: LXI H,TABPCE    ; adresa tabulky funkce
        MVI C,POCM      ; pocet vzorku
CYKL:   MOV A,M         ; hodnota z tabulky
        INX H           ; posun ukazatel
        OUT PB55        ; vysli na prevodnik
        DCR C           ; sniz citac
        JNZ CYKL        ; a opakuj
KONEC:                  ; atd.
```

Periodu průběhu je možné ovlivňovat pomocí počtu bodů $M$ v cyklu, vkládáním instrukcí, čekacích smyček aj. Perioda základního cyklu je určena vzorcem

$$T_p = 37MT_h$$

kde $T_h$ je perioda hodinových impulsů $T_h = 0{,}488\ \mu s$.
Zvolíme-li tabulku pro 256 bodů

```
TABPIL:  DB 0, 1, 2, 3
         DB 4, 5, 6, 7
         .......
         DB OFCH, OFDH, OFEH, OFFH
```

dostaneme pilový průběh jako u předchozích programů, minimální perioda ale může být až 4,6 ms, tedy nižší. Je ale možné i periodu zkrátit zmenšením počtu bodů.

Největší problémy této metody jsou tyto:

1. sestavení tabulky funkčních hodnot;
2. velká potřeba paměti pro složitější průběhy.

Pro ukázku budeme generovat signál se sinusovým průběhem. Protože s naším převodníkem nelze generovat záporné napětí, přidáme k sinusovému průběhu stejnosměrnou složku rovnou polovině rozsahu (80H) převodníku D/A.

```
                ; Generovani sinusoveho prubehu

2041 3E90   GENSIN: MVI A,MODIA
2043 D3F7           OUT PR55        ; iniciace obvodu 8255
2045 217420         LXI H,TABSIN    ; zacatek tabulky funkcnich hodnot
2048 0E1E           MVI C,30        ; pocet hodnot v tabulce
204A 1680           MVI D,80H       ; stejnosmerna slozka
204C 41     SIN:    MOV B,C
204D 7A     SIN0:   MOV A,D         ; ss slozka
204E 86             ADD M           ; pricti hodnotu SIN
204F D3F5           OUT PB55        ; vysli na prevodnik
2051 23             INX H           ; posun v tabulce smerem vpred
2052 05             DCR B           ; citac hodnot
2053 C24D20         JNZ SIN0        ; uz cela ctvrtperioda ?
2056 41             MOV B,C         ; prepis znovu citac
2057 7A     SIN1:   MOV A,D         ; ss slozka
```



```
2058 86             ADD M           ; pricti hodnotu SIN
2059 D3F5           OUT PB55        ; vysli na prevodnik
205B 2B             DCX H           ; posun v tabulce smerem vzad
205C 05             DCR B           ; pocitadlo hodnot
205D C25720         JNZ SIN1        ; uz druha ctvrtperioda ?
2060 41             MOV B,C
2061 7A     SIN2:   MOV A,D         ; ss slozka
2062 96             SUB M           ; odecti hodnotu SIN
2063 D3F5           OUT PB55        ; vysli na prevodnik
2065 23             INX H           ; posun v tabulce smerem vpred
2066 05             DCR B           ; pocitadlo hodnot
2067 C26120         JNZ SIN2        ; uz treti ctvrtperioda ?
206A 41             MOV B,C
206B 7A     SIN3:   MOV A,D         ; ss slozka
206C 96             SUB M           ; odecti hodnotu SIN
206D D3F5           OUT PB55        ; vysli na prevodnik
206F 2B             DCX H           ; posun v tabulce smerem vzad
2070 05             DCR B           ; pocitadlo hodnot
2071 C26B20         JNZ SIN3        ; uz ctvrta ctvrtperioda ?

2074 00070D TABSIN: DB 00H,07H,0DH,014H,01AH
2077 141A
2079 21272E         DB 021H,027H,02EH,034H,03AH
207C 343A
207E 40454B         DB 040H,045H,04BH,050H,055H
2081 5055
2083 5A5E63         DB 05AH,05EH,063H,067H,06BH
2086 676B
2088 6E7174         DB 06EH,071H,074H,077H,079H
208B 7779
208D 7B7C7D         DB 07BH,07CH,07DH,07EH,07FH
2090 7E7F
2092 7F             DB 07FH

                    END
```

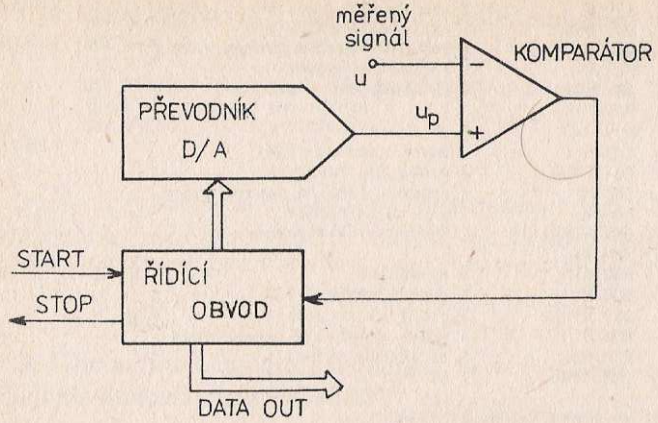
Pro první přiblížení zvolíme vzorkovací interval 3°. Tím dostáváme počet 120 hodnot na periodu, při rozdělení na čtvrtperiody dostáváme tabulku s 30 diskrétními hodnotami.

Snadno odvodíme vztah pro maximální frekvenci sinusového průběhu daného hodnotami $N$ v periodě generované programem. Platí

$$f = \frac{1}{(168N + 30)\ T_h}$$

Vypočtěte krok (tj. počet bodů $N$) tak, aby výsledná sinusoida měla frekvenci přibližně 300 Hz. Zkuste sestavit program v jazyce BASIC, který by generoval tabulku hodnot pro požadovanou frekvenci přímo v šestnáctkovém kódu.

Sestavte a ověřte program pro generování signálu s pravoúhlým průběhem s pevnou úrovní a periodou určenou stavem spínačů. Určete minimální a maximální frekvenci průběhu (asi 260 Hz až 32 kHz).




*Obr. 30.* Blokové schéma převodníku A/D

## 7.7. PŘEVOD ANALOGOVÝCH SIGNÁLŮ NA ČÍSLICOVÉ

### 7.7.1. Teoretický rozbor

V mnoha aplikacích mikropočítačů (především v řídicí technice) je třeba řešit úlohy sběru dat analogového charakteru. Přitom je nutný převod analogového signálu na číslicový (převod A/D). Je to úloha mnohem obtížnější než převod D/A. Odráží se to i v té skutečnosti, že se dosud u nás nevyrábějí převodníky A/D v integrované formě. Proto pro měření navrhneme zpětnovazební převodník A/D, který se skládá z převodníku D/A, komparátoru a řídicího obvodu, který je v našem případě realizován mikroprocesorem (blokové schéma je na obrázku 30).

Řídicí obvod pomocí převodníku D/A vygeneruje signál s určitým napětím $u_p$, které se komparátorem porovnává s úrovní měřeného napětí $u$. Podle výsledku komparace, který je dodáván do řídicího obvodu, se zvolí nová hodnota pro převodník, až hodnota $u_p - u$ klesne na úroveň nižší, než odpovídá nejnižšímu bitu převodníku D/A. Poslední hodnota je pak předávána z převodníku ven jako výstupní (DATA OUT).

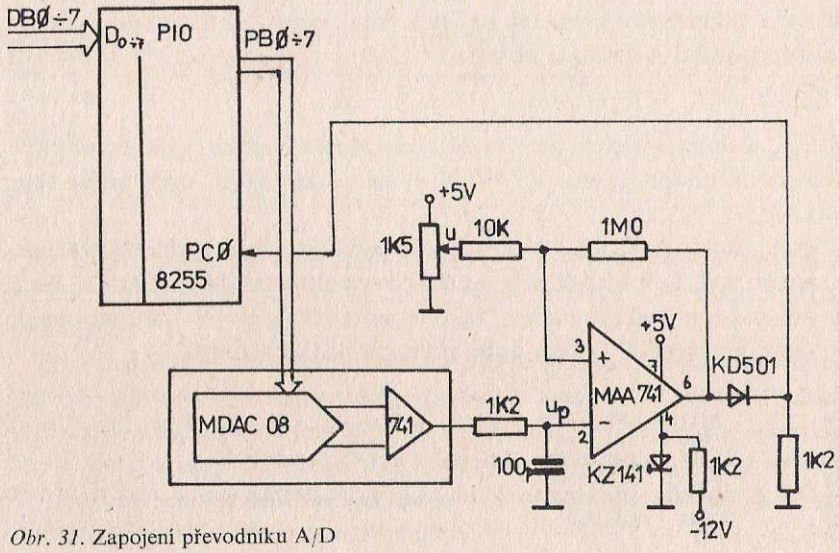
Řídicí obvod může být realizován logickými obvody nebo (jako v našem případě) samotným mikroprocesorem (pokud neanalyzujeme rychlejší průběhy nebo mikroprocesor nezastává důležitější funkce). Vyzkoušíme dvě různé metody řízení převodu, porovnáme jejich rychlost a efektivnost.



Pro zjednodušení uvažujme pouze kladná napětí, neboť i záporná napětí lze na kladná napětí převést (přičtením stejnosměrné složky – za cenu snížení přesnosti) nebo lze zapojení upravit.

### 7.7.2 Zapojení

V zapojení je využito zapojení převodníku D/A z úlohy 7.6 (bez vstupních spínačů), na výstup operačního zesilovače připojíme komparátor, sestavený opět z operačního zesilovače MAA741 napájeného ze zdroje ∓5 V. Hodnota výstupního napětí bude proto ve stejných mezích. Záporné napětí na výstupu z komparátoru potlačíme diodou s odporem, takže výsledný signál, přiváděný na vstup CØ obvodu 8255, má úroveň TTL (obr. 31).




*Obr. 31.* Zapojení převodníku A/D

### 7.7.3 Měření

#### Inkrementační převodník A/D

V první úloze zvolíme nejjednodušší způsob určení velikosti napětí pomocí postupné inkrementace (zvyšování) komparačního napětí $u_p$ až do té doby, než toto napětí přesáhne vstupní napětí $u$. V tuto chvíli přejde

výstup z komparátoru (bit 0 brány C obvodu 8255) na úroveň L, hodnota čítače v programu je úměrná vstupnímu napětí. Tuto hodnotu pomocí zvolené konstanty (dělení čtyřmi) převedeme na hodnotu pro zobrazení, ve smyčce DEL10 programu převedeme na 2 desítkové číslice, které zobrazíme na displeji. Takto získaný číslicový voltmetr je ale třeba „kalibrovat", nejlépe připojením přesného vstupního napětí +5 V a nastavením trimru $R$ tak, aby displej zobrazoval přesně 5,0.

*Poznámka.* Vždy po předání hodnoty čítače do převodníku D/A je třeba do programu vřadit čekací smyčku, která bude kompenzovat zpoždění převodníku D/A a obou operačních zesilovačů. Podle katalogových hodnot nastavte velikost konstanty CON1 v čekací smyčce CEKEJ.

Doba převodu inkrementačního převodníku roste s velikostí vstupního napětí. Označíme-li dobu kroku převodníku A/D (doba trvání čekací smyčky CEKEJ) $T_c$, pak celková doba převodu napětí $u_{vst} = N\,du$, kde $du$ je elementární přírůstek napětí a $N$ je počet kroků, pro dané nastavení převodníku A/D bude přibližně

$$T_{prev} = N(T_c + 26{,}5) + 20 + T_{vyst} \qquad (\mu s)$$

kde $T_{vyst}$ je doba potřebná na převod a zobrazení výsledku (v našem případě doba trvání podprogramu VYSTUP – asi 81 až 144 μs, opět podle velikosti $N$).

Určete závislost $T_{vyst}$ na $N$ a proveďte celkový odhad rychlosti převodu pro napětí 0, 1, 2, 3, 4 a 5 V pro náš číslicový voltmetr (200 dílků na +5 V). Na osciloskopu sledujte napětí. Jaké je teoreticky nejvyšší vstupní napětí pro daný převodník? Určete dobu převodu pro toto napětí.

```
2000                ORG 2000H
0099        MOBIAC  EQU 099H
00F7        PR55    EQU 0F7H
00F6        PC55    EQU 0F6H
00F5        PB55    EQU 0F5H
03AD        DSPTB   EQU 3ADH        ; dekod. tab. na 7segm. displ.
00XX        CON1    EQU XXH

            ; inkrementacni prevodnik
2000 3E99   ADINKR: MVI A,MOBIAC
2002 D3F7           OUT PR55        ; 8255 modus 0, B vyst., C vstup
2004 EF             RST 5           ; smaz displej
2005 0600   DALSI:  MVI B,0         ; zacni od 0
2007 78     INKREM: MOV A,B
2008 D3F5           OUT PB55        ; vysli na prev. D/A
200A 0EXX   CEKEJ0: MVI C,CON1      ; konstanta cek. smycky
200C 0D             DCR C
200D C20A20         JNZ CEKEJ0      ; kompenzuj zpozdeni prevodu
2010 DBF6           IN PC55         ; vstup z komparatoru
2012 1F             RAR
2013 04             INR B           ; zvys citac
```



```
2014 CA1A20         JZ KONEC0       ; uz na maximu
2017 DA0720         JC INKREM       ; jeste nesrovnano
201A 05     KONEC0: DCR B           ; napeti shodna
201B 78             MOV A,B
201C CD2220         CALL VYSTUP     ; zobraz napeti desitkove
201F C30520         JMP DALSI       ; a znova

2022 06FF   VYSTUP: MVI B,0FFH
2024 1F             RAR
2025 1F             RAR             ; vydel hodnotu 4
2026 E63F           ANI 3FH         ; zamaskuj neplatne bity
2028 D60A   DEL10:  SUI 10
202A 04             INR B
202B D22820         JNC DEL10       ; vydel 10
202E C60A           ADI 10          ; dorovnej jednotky
2030 11013C ZOBRAZ: LXI D,3C01H     ; adresa displeje
2033 21AD03         LXI H,DSPTB     ; tabulka prevodu cisel na
2036 48             MOV C,B         ; sedmisegmentovy kod
2037 0600           MVI B,0         ; desitky do reg. BC
2039 09             DAD B
203A 4F             MOV C,A         ; jednotky do BC
203B 7E             MOV A,M         ; kodovane desitky do A
203C 12             STAX D          ; na displej
203D 1B             DCX D
203E 21AD03         LXI H,DSPTB     ; tabulka kodu
2041 09             DAD B
2042 7E             MOV A,M         ; prevedene jednotky
2043 F608           ORI 8           ; rozsvit u nich tecku
2045 12             STAX D          ; a na displej
2046 C9             RET
```

## Převod A/D metodou půlení intervalů (aproximační)

Ze závěrů předchozího měření vychází, že doba převodu je pro inkrementační převodník příliš veliká, navíc je plně závislá na velikosti měřené hodnoty.

Tyto nedostatky odstraňuje *metoda půlení intervalů* (nebo *metoda postupných aproximací*). Její princip spočívá v tom, že se měřená veličina porovnává s hodnotou, která leží v polovině naposledy určeného intervalu. Navíc tento princip je určen (díky binárnímu dělení) pro snadné výpočty ve dvojkové soustavě. V praktických konstrukcích převodníku A/D se používá zásadně tento způsob měření.

Počet kroků převodu je pro všechny hodnoty stejný (je roven počtu bitů převodníku), čímž může být vzorkování vstupního signálu pravidelné a relativně rychlé.

### *Popis programu*

Při výpočtu hodnoty nejprve porovnáme vstupní napětí s výstupním napětím převodníku D/A při zadání čísla s jedničkou v nejvyšším bitu. Pozici jedničky uchováme v registru B. Je-li vstupní hodnota větší než

dané číslo, ponecháme jedničku v dané pozici mezivýsledku, je-li menší, pak tuto jedničku nulujeme (pomocí instrukce XRA B, kde B = 0, popř. je rovno pozici). Posuneme pozici, přidáme ji (logickou funkcí OR) k mezivýsledku a pokračujeme znovu, dokud neproběhne všech 8 kroků (indikace přeplnění jedničkového bitu v pozici do CY).

```
                ; aproximacni prevodnik A/D

2047 3E99     ADAPR:  MVI A,MOBIAC
2049 D3F7             OUT PR55          ; 8255 mod 0, B vyst., C vstup
204B EF               RST 5             ; smaz displej
204C 0680     ADPRV0: MVI B,80H         ; pocatecni hodnota
204E 78               MOV A,B           ; pozice jednicky
204F D3F5     APROX:  OUT PB55          ; na prevodnik D/A
2051 57               MOV D,A           ; aprox. hodnota
2052 0EXX     CEKEJ1: MVI C,CON1
2054 0D               DCR C
2055 C25220           JNZ CEKEJ1        ; vyrovnej zpozdeni prevodu
2058 DBF6             IN PC55
205A E601             ANI 1             ; nastav priznak Z, byl-li
205C 78               MOV A,B
205D 1F               RAR               ; posun pozici
205E C26320           JNZ NECH          ; ups , pak nuluj bit v pozici
2061 0600             MVI B,0           ; ups , pak nebudeme menit bit
2063 4F       NECH:   MOV C,A           ; novou pozici do C
2064 7A               MOV A,D           ; aprox. hodnota
2065 DA6E20           JC KONEC1         ; pozice jednicky mimo byte
2068 A8               XRA B             ; nuluj/nenuluj bit v pozici
2069 41               MOV B,C           ; nova pozice do B
206A B1               ORA C             ; pridej pozici k aprox.hodn.
206B C34F20           JMP APROX         ; jdi na dalsi porovnani

206E CD2220   KONEC1: CALL VYSTUP       ; preved na desit. a zobraz
2071 C34C20           JMP ADPRV0

                      END
```

Sestavte vzorec pro dobu převodu tohoto typu a zjistěte jeho typickou hodnotu (1,3 ms). Z její velikosti je vidět, že tímto způsobem převodu je možné vzorkovat i signály s frekvencí až 50 Hz, pro něž dostáváme asi 15 měřených bodů v periodě.

Na oscilografu sledujte i tentokrát průběh napětí a celkovou dobu převodu pro napětí 0, 1, 2, 3, 4 a 5 V; nakreslete zjištěný průběh.

Zkuste na vstup převodníku napojit zdroj sinusového napětí 50 Hz s amplitudou 5 V a zakreslete průběh během jedné periody s několika postupnými aproximacemi. Rozhodněte, zda určení hodnoty napětí proměnného průběhu je v daném časovém okamžiku zcela správné.

Modifikujte program číslicového voltmetru tak, aby program zaznamenal do tabulky v paměti určený počet bodů daného průběhu tak, aby je bylo možné znovu přenést do analogového tvaru. Rozmyslete možné



úpravy programu generování signálů s obecnějším průběhem z odst. 7.6.3 tak, aby vstupní i výstupní průběhy byly časově téměř shodné. Odhadněte zkreslení nového analogového signálu oproti původnímu.

## 7.8. TESTOVÁNÍ LOGICKÝCH INTEGROVANÝCH OBVODŮ POMOCÍ MIKROPOČÍTAČE

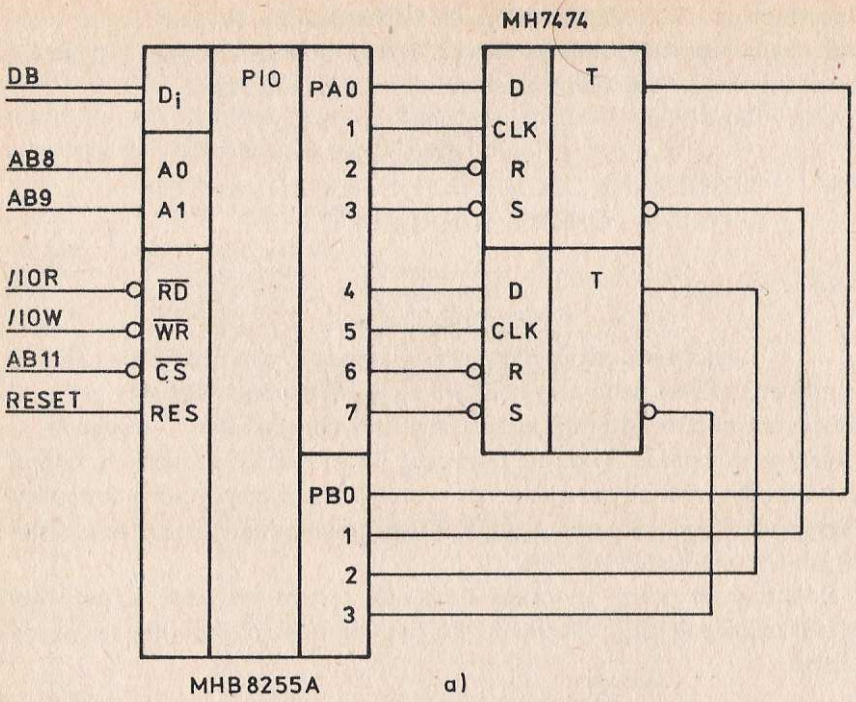
### 7.8.1. Úvod

Logické obvody potřebujeme testovat obvykle před jejich zabudováním do zařízení nebo při podezření na jejich chybnou funkci v zařízení. Testování můžeme provádět na specializovaných zařízeních — testerech —, ručně nebo pomocí různých přístrojů. Použití specializovaných testerů má výhody proti jiným způsobům v rychlosti a komplexnosti testování. Nevýhodou je vysoká cena testerů. Ruční testování je zdlouhavé a u složitých obvodů neproveditelné.

Pokud stačí pouze testování funkčním testem, můžeme k testování využít mikropočítač, ke kterému vhodným způsobem připojíme testovaný obvod.

### 7.8.2. Zapojení

Pro připojení testovaného obvodu k mikropočítači využijeme programovatelný obvod MHB8255A. Brána A může sloužit k připojení vstupu testovaného obvodu, brána B k připojení výstupů testovaného obvodu. Bránu C můžeme případně použít k rozšíření počtu vstupů nebo výstupů nebo ke kontrole hodnot na vstupech testovaného obvodu. Využití brány C předpokládá ovšem úpravu programu testování. Na obr. 32 jsou ukázány příklady připojení testovaných obvodů typu MH7400, MH7474. Obdobným způsobem můžeme připojit i jiné obvody MSI a SSI. Jediná podmínka je, aby počet vstupů a výstupů nepřesáhl možnosti obvodu MHB8255A. Dále je třeba vzít v úvahu zatížitelnost výstupů obvodu MHB8255A. Pokud by vstupní proud testovaného obvodu přesáhl hodnotu výstupního proudu jednotlivých vodičů výstupních datových bran obvodu MHB8255A, je nutné použít oddělovací logické členy.




### 7.8.3. Program pro testování

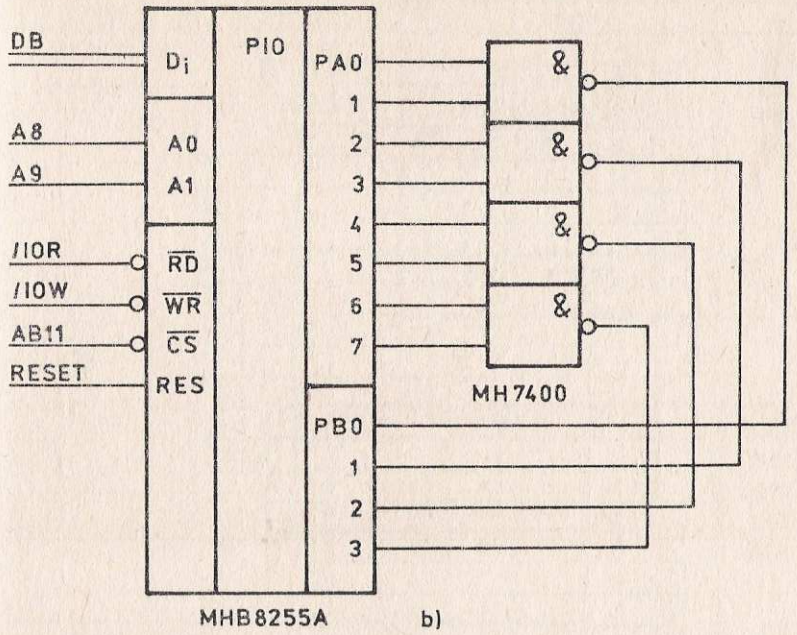
Pro každý logický obvod lze sestavit tabulku funkčních testů. Při sestavování této tabulky vycházíme z funkce daného obvodu. Je třeba brát v úvahu úplné pokrytí poruch typu t0 a t1 na všech vývodech základních logických členů, ze kterých je složen logický obvod integrovaný v jednom pouzdře. Použité symboly mají tento význam:

0 – dolní úroveň signálu (L);
1 – horní úroveň signálu (H);
↑ – přechod z úrovně L na úroveň H;
↓ – přechod z úrovně H na úroveň L.

Na obr. 33 jsou příklady tabulek testů některých vybraných obvodů. Podrobněji je tato problematika popsána v literatuře [9].

Nyní sestavíme program pro vlastní testování jednotlivých obvodů. Můžeme to udělat dvěma způsoby, a to:






*Obr. 32.* Příklady připojení testovaných obvodů

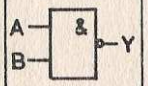
1. pro každý obvod vytvoříme speciální program;
2. vytvoříme univerzální program pro testování větší skupiny obvodů.

Nevýhody prvního způsobu jsou zřejmé. Proto se pokusíme o způsob druhý, tzn. sestavit univerzální program. Takový program potom bude testovat daný obvod podle tabulky testu, kterou pro každý obvod vytvoříme. V této tabulce mohou být např. tyto údaje:

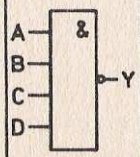
1. položka – počet testovacích kroků;
2. položka – maska pro vstupní bránu B obvodu 8255, slouží k maskování hodnot nepoužitých vstupů brány B obvodu 8255;
3. položka – hodnota zapisovaná do brány A v prvním kroku;
4. položka – správná hodnota z brány B v prvním kroku;
5. položka – hodnota zapisovaná do brány A v druhém kroku;
6. položka – správná hodnota z brány B v druhém kroku,

.
.
.

| typ | test | | | | |
|---|---|---|---|---|---|
| MH7400<br>MH7403<br>MH7437<br>MH7438 | | 1(4, 9,12) | 2(5, 10,13) | 3(6, 8,11) | |
| | | A | B | Y | |
| | 1 | 0 | 1 | 1 | |
| | 2 | 1 | 0 | 1 | |
| | 3 | 1 | 1 | 0 | |
| | | vstup | | výst. | |




| typ | test | | | | | |
|---|---|---|---|---|---|---|
| MH7420<br>MH74S20<br>MH7440<br>MH74S40 | | 1 (9) | 2 (10) | 4 (12) | 5 (13) | 6 (8) |
| | | A | B | C | D | Y |
| | 1 | 0 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 0 | 1 | 1 | 1 |
| | 3 | 1 | 1 | 0 | 1 | 1 |
| | 4 | 1 | 1 | 1 | 0 | 1 |
| | 5 | 1 | 1 | 1 | 1 | 0 |
| | | vstup | | | | výst. |



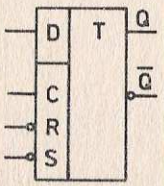

a)

Testovací program bude potom následující:

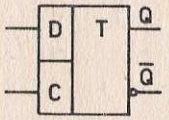
```
                ; testovaci program
2000                    ORG 2000H
2000 3E8A               MVI A,8AH       ;nastaveni rezimu obvodu 8255
2002 D3F7               OUT CNTRL       ;A-vystup,B-vstup
2004 215020             LXI H,TABUL     ;adresa zacatku tabulky testu
2007 46                 MOV B,M         ;pocet kroku testu
2008 23                 INX H           ;adresa masky brany B
2009 4E                 MOV C,M         ;maska B
200A 23         TEST:   INX H           ;adresa vzorku do A
200B 7E                 MOV A,M
200C D3F4               OUT PORTA       ;hodnota vstupu testovaneho obvodu
200E 23                 INX H           ;adresa vystupni hodnoty
200F DBF5               IN PORTB        ;skutecna hodnota vystupu
2011 A1                 ANA C           ;testovaneho obvodu
2012 BE                 CMP M           ;otestuj spravnost
```



| typ | test | | | | | | |
|---|---|---|---|---|---|---|---|
| MH7474<br>MH74S74 | | 1 (13) | 2 (12) | 3 (11) | 4 (10) | 5 (9) | 6 (8) |
| | | R | D | C | S | Q | $\overline{Q}$ |
| | 1 | ↑ | 0 | 1 | 0 | 1 | 0 |
| | 2 | 1 | 0 | 1 | ↑ | 1 | 0 |
| | 3 | ↑ | 0 | 0 | 0 | 1 | 0 |
| | 4 | 1 | 0 | 0 | ↑ | 1 | 0 |
| | 5 | 1 | 0 | ↑ | 1 | 0 | 1 |
| | 6 | 0 | 1 | 0 | ↑ | 0 | 1 |
| | 7 | ↑ | 1 | 0 | 1 | 0 | 1 |
| | 8 | 1 | 1 | ↑ | 1 | 1 | 0 |
| | 9 | 0 | 1 | ↑ | ↑ | 0 | 1 |
| | 10 | ↑ | 1 | 1 | 1 | 0 | 1 |
| | | vstup | | | | výstup | |




| typ | test | | | | |
|---|---|---|---|---|---|
| MH7475 | | 2(3, 6,7) | 13 (4) | 16(15, 10, 9) | 1(14, 11,8) |
| | | D | C | Q | $\overline{Q}$ |
| | 1 | 0 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 1 | 0 |
| | 3 | 1 | 0 | 1 | 0 |
| | 4 | 0 | 0 | 1 | 0 |
| | 5 | 0 | 1 | 0 | 1 |
| | 6 | 0 | 0 | 0 | 1 |
| | 7 | 1 | 0 | 0 | 1 |
| | | vstup | | výstup | |




b)

*Obr. 33.* Tabulky testů vybraných logických obvodů

```
2013 C23202          JNZ ERR          ;v pripade chyby skok do monitoru
2016 05              DCR B            ;zda uz vsechny kroky
2017 C20A20          JNZ TEST         ;pokud ne, pokracuj v testovani
201A 212120          LXI H,DOBR       ;obvod je v poradku
201D 0E04            MVI C,4          ;podej o tom zpravu
201F E7              RST 4
2020 76              HLT
                     ;
2021 0D      DOBR:   DB 0DH,0,0BH,13H
2022 00
2023 0B
2024 13
2050         TABUL   EQU 2050H
0232         ERR     EQU 232H
00F4         PORTA   EQU 0F4H
00F5         PORTB   EQU 0F5H
00F7         CNTRL   EQU 0F7H
                     ;
                     END
```

V případě, že je obvod v pořádku, vypíše se na displeji zpráva *dobr*, jinak chybové hlášení *Err*.

Nyní si ukážeme příklady sestavené tabulky testů pro daný testovací program. Test obvodu MH7400 bude mít tři kroky:

```
2050                 ORG 2050H
2050 03      TABUL:  DB 3,0FH         ;3 kroky testu, maska pro 4 vystupy
2051 0F
2052 AA              DB 0AAH,0FH      ;1.krok = 1.radek tabulky
2053 0F
2054 55              DB 55H,0FH       ;2.krok = 2.radek tabulky
2055 0F
2056 FF              DB 0FFH,0        ;3.krok = 3.radek tabulky
2057 00
```

Test obvodu MH7474 bude mít 14 kroků:

```
2050                 ORG 2050H
 2050 0E     TABUL:  DB 0EH,0FH       ;14 kroku testu, maska pro 4 vystupy
 2051 0F
             ; prvni radek tabulky testu
 2052 22             DB 22H,0FH       ;vstupy R=S=D=L, CLK=H
 2053 0F
 2054 AA             DB 0AAH,5        ;       R=D=L, S=CLK=H
 2055 05
             ; druhy radek tabulky testu
 2056 EE             DB 0EEH,5        ;vstupy D=L, R=CLK=S=H
 2057 05
             ; treti radek tabulky testu
 2058 00             DB 0,0FH         ;vstupy R=D=CLK=S=L
 2059 0F
 205A 88             DB 88H,5         ;       D=CLK=S=L, R=H
 205B 05
             ; ctvrty radek tabulky testu
 205C CC             DB 0CCH,5        ;vstupy D=CLK=L, R=S=H
 205D 05
             ; paty radek tabulky testu
 205E EE             DB 0EEH,0AH      ;vstupy D=L, R=CLK=S=H
 205F 0A
```



```
             ; sesty radek tabulky testu
 2060 11             DB 11H,0FH       ;vstupy R=CLK=S=L, D=H
 2061 0F
 2062 55             DB 55H,0AH       ;       R=CLK=L, D=S=H
 2063 0A
             ; sedmy radek tabulky testu
 2064 DD             DB 0DDH,0AH      ;vstupy CLK=L, R=D=S=H
 2065 0A
             ; osmy radek tabulky testu
 2066 FF             DB 0FFH,5        ;vstupy R=D=CLK=S=H
 2067 05
             ; devaty radek tabulky testu
 2068 11             DB 11H,0FH       ;vstupy R=CLK=S=L, D=H
 2069 0F
 206A 77             DB 77H,0AH       ;       R=L, D=CLK=S=H
 206B 0A
             ; desaty radek tabulky testu
 206C FF             DB 0FFH,0AH      ;vstupy R=D=CLK=S=H
 206D 0A
```

Obdobným způsobem můžeme podle tabulky testu a zapojení testovaného obvodu k obvodu MHB 8255A sestavit tabulky programu pro další testované obvody.

## 7.9. SÉRIOVÝ PŘENOS DAT S VYUŽITÍM OBVODU 8255

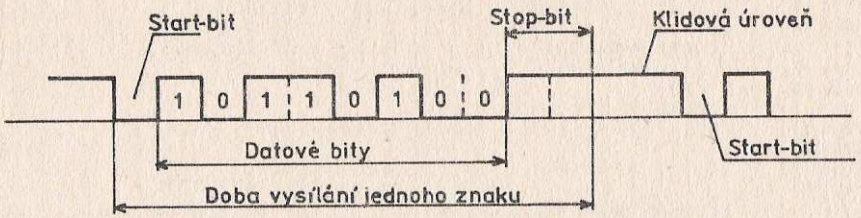
### 7.9.1. Úvod

Přenos dat mezi dvěma systémy, např. periferním zařízením a mikropočítačem, lze realizovat dvěma způsoby, a to:

– paralelně;
– sériově.

Pro paralelní přenos je třeba několik vodičů datové sběrnice (jejich počet je dán velikostí přenášeného slova, obvykle 8 nebo 16 bitů), vodiče signálů pro řízení přenosu a vodiče signálové země. Po sériový přenos dat potřebujeme pouze jeden datový vodič a vodič signálové země. Přenos je potom realizován v pevné časové posloupnosti bit po bitu. Sériový přenos můžeme realizovat jako synchronní nebo asynchronní. Při *synchronním přenosu* dat jsou data vysílána a přijímána synchronně s daným hodinovým signálem. Tento hodinový signál se nepřenáší mezi systémy, ale je generován v přenosovém zařízení (modemu). Na straně vysílače je generátor frekvence závislé na přenosové rychlosti a na straně přijímače je tento generátor dolaďován na základě přijímané zprávy. Z tohoto důvodu je třeba, aby se z vysílače do přijímače neustále přenášela data. Pokud nejsou k dispozici

významová data, je třeba vysílat domluvenou výplň, která slouží pouze k udržení synchronizace generátoru hodinových impulsů na přijímací straně.




*Obr. 34.* Průběh signálu na lince při asynchronním přenosu

Při *asynchronním způsobu přenosu* je zpráva rozložena na jednotlivé znaky dlouhé $n = 5$ až 8 bitů. Každý znak začíná start-bitem a končí stop-bitem (obr. 34). Příjem znaku se provádí tak, že přijímač čeká na začátek start-bitu, na lince je to přechod z úrovně H do L. Potom otestuje, zda úroveň L trvá až do poloviny trvání start-bitu. Tak se najde střed přenášených bitů. Nyní se vždy po době trvání přenosu jednoho bitu otestuje vstupní informace na lince. Tím se získá hodnota jednotlivých bitů vstupní informace. Po daném počtu datových bitů je očekáván stop-bit, tj. nastane přechod na lince do stavu H. Stop-bit trvá nejméně po dobu přenosu jednoho bitu. Obvykle lze na straně vysílače délku stop-bitu nastavit v rozmezí 1, 1,5 a 2 bitů. Po stop-bitu následuje na lince buď klidová úroveň (úroveň H), nebo start-bit dalšího znaku, po kterém dojde k přijetí nového znaku. Přenášené znaky mohou být zabezpečeny pro možnost zjištění chyby přenosu pomocí *parity*, sudé nebo liché. Potom datový bit $n + 1$ je paritní bit.

Při asynchronním přenosu se vykonávají následující činnosti:

– vyslání start-bitu vysílačem, na straně přijímače detekce začátku a potom středu přijatého start-bitu;

– sériové vysílání/příjem bitů znaku;

– generování a vyslání paritního bitu vysílačem, příjem a vyhodnocení paritního bitu přijímačem;

– vyslání stop-bitu a jeho detekce na straně přijímače;

– vyslání klidové úrovně, pokud není další znak.

Jak tento přenos realizovat?

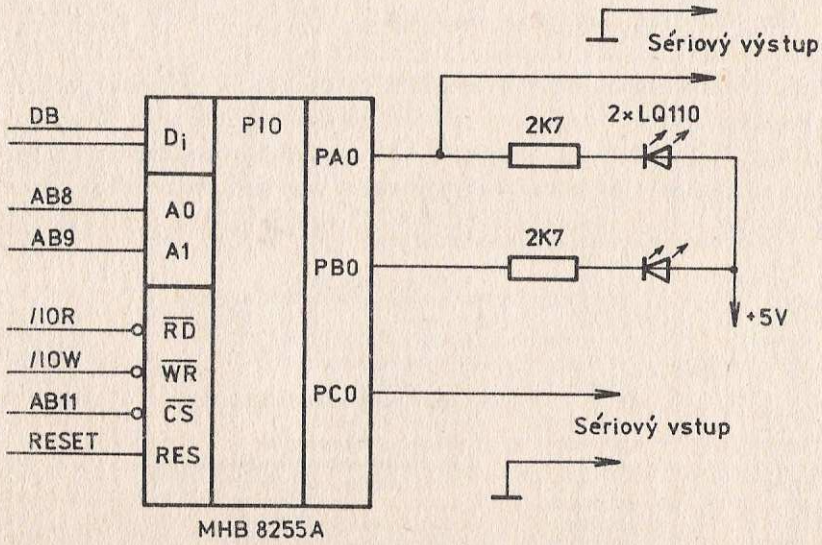
– Pomocí obvodů SSI, MSI sestavit vysílač, přijímač.



– Využít speciálních obvodů LSI pro řízení sériového přenosu, např. MHB8251, U856D, MHB1012 atd.

– Využít program mikropočítače.

Pro vlastní realizaci potřebného proudového nebo napěťového zobrazení logických hodnot na lince platí mezinárodní standardy – proudová smyčka 20 mA, RS-232C, V24/28. Lze využít již vyráběných přenosových zařízení různých výrobců.




*Obr. 35.* Sériový vstup a výstup pomocí obvodu 8255

## 7.9.2. Zapojení

V této úloze si ukážeme sériový přenos dat, který je řízen mikropočítačem. Jako vysílač a přijímač je využit programovatelný obvod MHB8255A. Přenos je realizován mezi dvěma mikropočítači TEMS 80-03. Zapojení obvodu MHB8255A pro sériový přenos dat je na obr. 35. Tento obvod je naprogramován do modu 0, brány A, B jsou výstupní, brána C je vstupní. Pro vlastní vysílání dat slouží výstup PA0. Pro indikaci přenášených úrovní je na výstup připojena světelná emisní dioda. Přes výstup

PB0 budeme určovat okamžik středu vysílaných/přijímaných bitů. Vývod PC0 slouží jako vstup přijímaných dat. Pro zobrazení vysílaných/přijímaných znaků použijeme zapojení displeje z čl. 7.5.

### 7.9.3. Program pro vysílání

Znaky, které chceme vysílat, zadáváme pomocí klávesnice mikropočítače TEMS 80-03. Po stisknutí příslušné klávesy převede program kód stisknuté klávesy na kód ASCII. Tím je umožněno zadávat omezený soubor znaků – šestnáctkové číslice a písmena G, H, I, J, K, L. Zadaný znak zobrazíme pro kontrolu na displeji. Vypočteme paritní bit a spustíme sériové vysílání, tj. start-bit, datové bity, stop-bit. Rychlost přenosu je 1 bit/s. Dioda na výstupu PA0 svítí, pokud vysíláme úroveň L, a dioda na výstupu PB0 blikne vždy uprostřed intervalu přenášeného bitu.

```
                ; seriove vysilani znaku ASCII
2000                    ORG 2000H
2000 3E89               MVI A,89H       ;nastav obvod 8255
2002 D3F7               OUT CNTRL       ;A,B vystup, C VSTUP, rezim 0
2004 3E01               MVI A,1
2006 D3F5               OUT PORTB       ;zhasni diodu na PB0
2008 3E01       START:  MVI A,1         ;klidova uroven H
200A D3F4               OUT PORTA       ;na vystup
200C D7                 RST 2           ;cekej na stisknuti klavesy
200D CD6920             CALL ASCII      ;preved na kod ASCII
2010 D3EF               OUT MH12        ;zobraz vysilany znak
2012 CD1F20             CALL PARBIT     ;vypocitej hodnotu paritniho bitu
2015 CD2620             CALL VYSLI      ;vysli seriove na linku
2018 3E20               MVI A,20H
201A D3EF               OUT MH12        ;zhasni displej
201C C30820             JMP START       ;klidovy stav
                ;
                ;  generovani paritniho bitu
201F E67F       PARBIT: ANI 7FH         ;vynuluj paritni bit
2021 B7                 ORA A           ;vypocitej hodnotu parity
2022 E8                 RPE             ;pri sude parite navrat
2023 F680               ORI 80H         ;pri liche dopln jednicku
2025 C9                 RET             ;navrat
                ;
                ;  vysilani znaku
2026 4F         VYSLI:  MOV C,A         ;uklid znak
2027 0608               MVI B,8         ;pocitadlo bitu
2029 AF                 XRA A           ;vysli start-bit
202A D3F4               OUT PORTA
202C CD4A20             CALL DOBA
202F 79                 MOV A,C         ;vysilany znak
2030 D3F4       CYKL:   OUT PORTA       ;vysila se od 0.bitu
2032 CD4A20             CALL DOBA       ;doba vysilani bitu
2035 79                 MOV A,C         ;priprav dalsi bit
2036 0F                 RRC
2037 4F                 MOV C,A
2038 05                 DCR B           ;uz vyslany vsechny bity?
```



```
2039 C23F20             JNZ VEN         ;kdyz ano, na vysilani stop-bitu
203C C33020             JMP CYKL        ;jinak pokracuj ve vysilani
203F 3E01       VEN:    MVI A,1         ;stop-bit na vystup
2041 D3F4               OUT PORTA
2043 CD4A20             CALL DOBA       ;trvani stop-bitu po dobu
2046 CD4A20             CALL DOBA       ;dvou datovych bitu
2049 C9                 RET             ;navrat
                ;
                ;  generovani casoveho intervalu s bliknutim uprostred
204A CD5420     DOBA:   CALL CAS        ;cekej 0.5 s
204D CD5B20             CALL BLIK       ;blikni
2050 CD5420             CALL CAS        ;cekej
2053 C9                 RET             ;navrat
                ;
                ;  generovani intervalu delky 0.5 s
2054 110080     CAS:    LXI D,8000H
2057 CD1802             CALL DELAY
205A C9                 RET
0218            DELAY   EQU 218H
                ;
                ;  bliknuti diodou PB0
205B AF         BLIK:   XRA A
205C D3F5               OUT PORTB       ;rozsvit diodu
205E 110010             LXI D,1000H
2061 CD1802             CALL DELAY      ;cekej
2064 3E01               MVI A,1         ;a zhasni ji
2066 D3F5               OUT PORTB
2068 C9                 RET

                ;  prevod kodu klavesy na kod ASCII
2069 F630       ASCII:  ORI 30H         ;pridej 30H
206B FE39               CPI 39H         ;vysledek (= 39H?
206D D8                 RC              ;kdyz mensi, vrat se
206E C8                 RZ              ;kdyz roven, vrat se
206F C607               ADI 7           ;jinak pricti 7
2071 C9                 RET
                ;
00F4            PORTA   EQU 0F4H
00F5            PORTB   EQU 0F5H
00F6            PORTC   EQU 0F6H
00F7            CNTRL   EQU 0F7H
00EF            MH12    EQU 0EFH
                ;
                        END
```

### 7.9.4. Program pro příjem

Znak, který jsme vyslali jedním mikropočítačem, můžeme nyní přijmout druhým mikropočítačem pomocí následujícího programu. Tento program využívá některé podprogramy z vysílače. Při příjmu čeká na příchod start-bitu a v polovině intervalu blikne diodou na výstupu PB0. Tato dioda blikne vždy uprostřed datových bitů. Po příjmu znaku se tento znak zobrazí na displeji z čl. 7.5. Program potom přejde opět na čekání příjmu dalšího znaku.

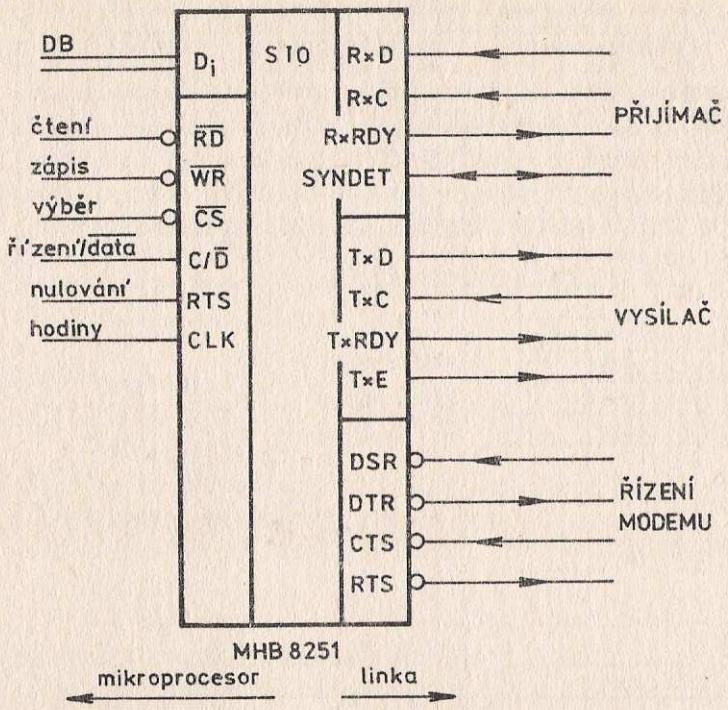
```
                ;  prijem seriove vysilanych znaku a zobrazeni
2080                    ORG 2080H
2080 3E89               MVI A,89H       ;nastaveni obvodu 8255
2082 D3F7               OUT CNTRL       ;A,B vystup, C vstup
2084 0608       ZNAK:   MVI B,8         ;pocet bitu ve znaku
2086 DBF6       CEK:    IN PORTC        ;cti uroven na lince
2088 1F                 RAR             ;je nulova?
2089 DA8620             JC CEK          ;kdyz ne, cekej
208C CD5420             CALL CAS        ;cekej 0.5 s
208F DBF6               IN PORTC        ;zda jeste nulova uroven na lince
2091 1F                 RAR
2092 DA8620             JC CEK          ;kdyz ne, cekej znovu na start bit
2095 CD5B20             CALL BLIK       ;blikni
2098 CD5420             CALL CAS        ;cekej 0.5 s
209B CD5420     DATA:   CALL CAS        ;cekej 0.5 s
209E DBF6               IN PORTC        ;datovy bit
20A0 1F                 RAR
20A1 79                 MOV A,C         ;pridej do reg C
20A2 1F                 RAR
20A3 4F                 MOV C,A
20A4 CD5420             CALL CAS        ;cekej 0.5 s
20A7 05                 DCR B           ;pripocitej dalsi bit
20A8 C29B20             JNZ DATA        ;pokracuj v prijmu datovych bitu
20AB CD5420             CALL CAS        ;prijmi stop-bit
20AE DBF6               IN PORTC        ;hodnota stop-bitu
20B0 1F                 RAR             ;je to H?
20B1 D23202             JNC ERR         ;kdyz ne, chyba
20B4 CD5B20             CALL BLIK       ;blikni
20B7 CD5420             CALL CAS        ;cekej 0.5 s
20BA 79                 MOV A,C         ;precteny znak
20BB CD1F20             CALL PARBIT     ;spocitej hodnotu paritniho bitu
20BE B9                 CMP C           ;zkontroluj s doslou hodnotou
20BF C23202             JNZ ERR         ;odskok v pripade neshody
20C2 D3EF               OUT MH12        ;zobraz na displeji
20C4 C38420             JMP ZNAK        ;cekej na dalsi znak
                ;
00EF            MH12    EQU 0EFH
00F6            PORTC   EQU 0F6H
00F7            CNTRL   EQU 0F7H
2054            CAS     EQU 2054H
205B            BLIK    EQU 205BH
0232            ERR     EQU 232H
201F            PARBIT  EQU 201FH
                ;
                        END
```

## 7.10. SÉRIOVÝ PŘENOS DAT S OBVODEM MHB8251

### 7.10.1. Úvod

Pro sériový přenos dat mezi dvěma systémy můžeme použít specializované obvody LSI. Jedním z nich je obvod MHB8251. Jde o programovatelný obvod, který můžeme použít k řízení synchronního a asynchronního sériového přenosu znaků o pěti až osmi bitech. Maximální přenosová rychlost pro synchronní přenos je 56 kbit/s a pro asynchronní 9,6 kbit/s.





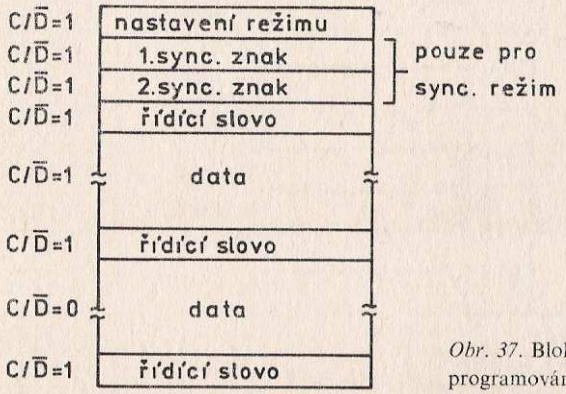

*Obr. 36.* Programovatelný sériový obvod 8251

Schematická značka tohoto obvodu je na obr. 36. Signály připojené na linku mají tento význam:

| | |
|---|---|
| R×D | – přijímaná data |
| $\overline{R \times C}$ | – hodiny přijímače |
| R×RDY | – indikace přijetí znaku přijímačem (úroveň H – znak přijat) |
| SYNDET | – vstup/výstup pro vnější synchronizaci |
| T×D | – vysílaná data |
| $\overline{T \times C}$ | – hodiny vysílače |
| T×RDY | – indikace, že do vysílače lze zapsat další znak (úroveň H – lze zapsat znak) |
| T×E | – indikace, že vyrovnávací paměť vysílače je prázdná (úroveň H – paměť prázdná) |
| DSR | – výstup pro řízení připojeného modemu |
| CTS | – vstup pro kontrolu činnosti vysílače připojeného modemu (úroveň L umožní vysílání dat z 8251, úroveň H zablokuje vysílání dat z 8251) |

RTS — výstup pro řízení zapnutí/vypnutí vysílače připojeného modemu (úroveň L — vysílač zapnut, H — vysílač vypnut).

Před použitím obvodu 8251 k přenosu dat je nutné tento obvod naprogramovat. Pro určení, zda je zapisované slovo z mikroprocesoru do obvodu 8251 řídicí nebo datové, je využit adresový vstup označený C/D. Tento vstup se obvykle připojuje na adresový bit A0. Pokud C/D = L, zadáváme data, jinak řídicí slovo. Blokový diagram postupu programování obvodu 8251 je na obr. 37.




*Obr. 37.* Blokový diagram programování obvodu 8251

Dále budeme popisovat pouze použití obvodu 8251 pro asynchronní režim. Tvar slabiky pro nastavení režimu je na obr. 38.
Význam jednotlivých bitů je tento:
B2, B1 — dělicí poměr rychlosti
00 — nastavení synchronního režimu
01 — 1×
10 — 16×
11 — 64×
L2, L1 — délka přijímaného/vysílaného znaku
00 — 5 bitů
01 — 6 bitů
10 — 7 bitů
11 — 8 bitů
PEN — povolení parity
1 — povoleno
0 — zablokováno



| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| S2 | S1 | EP | PEN | L2 | L1 | B2 | B1 |

*Obr. 38.* Formát slova pro nastavení asynchronního režimu obvodu 8251

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| EH | IR | RTS | ER | SBRK | RxE | DTR | TxEN |

*Obr. 39.* Řídicí slovo obvodu 8251

EP — typ parity
1 — sudá parita
0 — lichá parita
S2, S1 — délka stop-biu
00 — chybné nastavení
01 — 1 bit
10 — 1,5 bitu
11 — 2 bity
Režim lze nastavit vždy jen po vynulování, které lze provést buď technickými prostředky (RES = L), nebo programově v řídicím slově. Tvar *řídicího slova* je na obr. 39. Jednotlivé bity mají tento význam:
T×EN — zapnutí/vypnutí vysílače
1 — zapnutí
0 — vypnutí
DTR — ovládání výstupu DTR
1 — DTR = L
0 — DTR = H
R×E — zapnutí/vypnutí přijímače
1 — zapnutí
0 — vypnutí
SBRK — 1 = vysílání znaku BREAK
ER — 1 = vynulování příznaku chyb PE, OE, FE
RTS — ovládání výstupu RTS
1 — RTS = L
0 — UTS = H
IR — 1 = programové nulování obvodu 8255
EH — význam pouze v synchronním režimu

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| DSR | SYNDET | FE | OE | PE | TxEMPT | RxRDY | TxRDY |

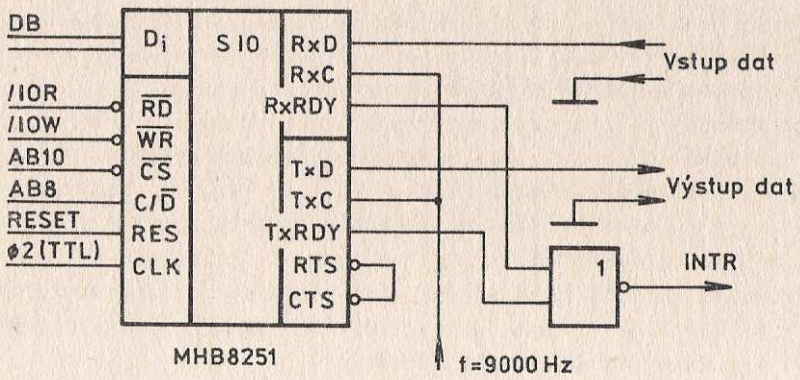
*Obr. 40.* Stavové slovo obvodu 8251

Při čtení z obvodu 8251 používáme pro určení typu čteného slova opět řídicí vstup C/$\overline{D}$. Pokud C/$\overline{D}$ = L, čteme vstupní data, jinak čteme stavové slovo obvodu 8251. Tvar přečteného *stavového slova* je na obr. 40. Jednotlivé bity mají tento význam:

T×RDY – 1 = do vysílače lze zapsat další znak
R×RDY – 1 = znak přijat přijímačem
T×EMPTY – 1 = vyrovnávací paměť vysílače prázdná
PE – 1 = chyba parity, PE se nuluje řídicím slovem, kde ER = 1
OE – 1 = chyba přetečení přijímače. OE se nuluje řídicím slovem, kde ER = 1
FE – 1 = nebyl zjištěn stop-bit v asynchronním režimu, FE se nuluje řídicím slovem, kde ER = 1
DSR – 1 = vstup DSR = L

## 7.10.2. Zapojení

V této úloze využijeme obvod 8251 z mikropočítače TEMS 80-03. Schéma zapojení je na obr. 41. Pro použití přerušovacího systému k ovládání obvodu 8251 použijeme vývody R×RDY a T×RDY, které jsou




*Obr. 41.* Zapojení obvodu 8251 pro přenos dat



přivedeny na vstup logického členu NOR. Výstup tohoto členu je potom žádost o přerušení. Přenos dat můžeme realizovat z výstupu na vstup téhož obvodu 8251 nebo mezi dvěma mikropočítači TEMS 80-03.

V této úloze je použit sériový přenos dat na úrovni TTL. Tento způsob přenosu lze použít do vzdálenosti řádově jednotek metrů. Pro delší vzdálenost je nutné použít buď proudovou smyčku ±20 mA, nebo standard RS-232-C s napěťovými úrovněmi pro hodnotu L: −3 V až −12 V, pro hodnotu H: 3 V až 12 V.

Pro zobrazování přijatých znaků použijeme zapojení displeje alfanumerických znaků z čl. 7.5.

## 7.10.3. Program pro přenos dat

Přenos bude realizován v asynchronním režimu. Rychlost přenosu dat bude 140 bit/s. Délka znaku 7 bitů, sudý paritní bit, 2 stop-bity. Při použití hodinového signálu 9 kHz bude dělicí poměr přenosové rychlosti 64 (9 000 : 140 = 64). Z těchto údajů vyplývá tvar slova pro nastavení režimu a řídicího slova. Iniciace obvodu 8251 bude tedy

```
MVI A,OFBH
OUT CNTR    ;    zápis slova pro nastavení režimu
MVI A,35H
OUT CNTR    ;    zápis řídicího slova
```

Takto můžeme ovšem inicializovat obvod 8251 pouze v případě, když jsme si jisti, že předcházelo nulování buď technickými, nebo programovými prostředky. Abychom se tomuto předpokladu vyhnuli, zapíšeme na adresu CNTR třikrát hodnotu 0 a potom 40H. Tím provedeme programové vynulování obvodu MHB8251 z každého nastaveného stavu.

Program pro vysílání a příjem dat je rozdělen na dvě části. V první části, které budeme říkat *hlavní program*, je nejprve vynulován a nastaven obvod 8251 do asynchronního režimu a zapnut vysílač a přijímač. Následující instrukcí je uvolněn přerušovací systém a hlavní program přejde do programové smyčky, kde se čeká na stisknutí klávesy znaku na klávesnici mikropočítače TEMS 80-03. Kód stisknutého znaku se uloží do proměnné KLAV, odkud si ji převezme *program obsluhy vysílače*. Potom hlavní program opět čeká na stisknutí další klávesy.

Druhá část programu je obsluha vysílače/přijímače dat obvodu 8251.

Přechod do této části programu je dán přerušením, které bude vyvoláno signálem T × RDY nebo R × RDY. Jakmile dojde k přerušení, obslužný program čte a analyzuje stavové slovo. Při bezchybném příjmu znaku je zobrazen na displeji. Pokud je zjištěna chyba při příjmu znaku, je indikována na displeji příznakem chyby (viz výpis programu). Pokud je vysílač schopen vysílat další znak, je mu předán ze symbolické adresy KLAV, kam byl uložen při stisknutí klávesy. Po obsluze přerušení následuje návrat do hlavního programu a čeká se na stisk další klávesy.

```
                ; prenos znaku obvodem 8251
2000                    ORG 2000H
2000 AF                 XRA A           ;nulovani 8251
2001 D3FB               OUT CNTR
2003 D3FB               OUT CNTR
2005 D3FB               OUT CNTR
2007 3E40               MVI A,40H
2009 D3FB               OUT CNTR        ;IR=1
200B 3EFB               MVI A,0FBH      ;asynchronni rezim
200D D3FB               OUT CNTR
200F 3E35               MVI A,35H       ;zapni vysilac,prijimac
2011 D3FB               OUT CNTR
2013 FB                 EI              ;povoleni prijeti zadosti o preruseni
2014 D7         ZNAK:   RST 2           ;cekej na stisk klavesy
2015 CD1E20             CALL ASCII
2018 322020             STA KLAV        ;uloz kod klavesy
201B C31420             JMP ZNAK        ;dalsi znak
                ;
201E F630       ASCII:  ORI 30H
2020 FE39               CPI 39H
2022 D8                 RC
2023 C8                 RZ
2024 C607               ADI 7
2026 C9                 RET
                ;
                ;  obsluha preruseni od 8251
2050                    ORG 2050H
2050 DBFB       INTR:   IN CNTR         ;cti stavove slovo
2052 47                 MOV B,A         ;uloz je
2053 E602               ANI 2           ;prerusil prijimac?
2055 CA6D20             JZ VYSL         ;kdyz ne, zkus vysilac
2058 78                 MOV A,B         ;nebyla chyba?
2059 E634               ANI 34H
205B CA6920             JZ DOBRY        ;jestlize ne, precti znak
205E 3E40               MVI A,'@'       ;zobraz priznak chyby @
2060 D3EF               OUT MH12
2062 3E35               MVI A,35H       ;nuluj priznak chyb v 8251
2064 D3FB               OUT CNTR
2066 C36D20             JMP VYSL        ;skok na obsluhu vysilace
2069 DBFA       DOBRY:  IN DATA         ;precti prijaty znak
206B D3EF               OUT MH12        ;zobraz ho
206D 78         VYSL:   MOV A,B         ;ma vysilac co vysilat?
206E E601               ANI 1

2070 CA7820             JZ KONEC
2073 3A2020             LDA KLAV        ;kdyz ne, zadej dalsi znak
2076 D3FA               OUT DATA
2078 FB         KONEC:  EI              ;povoleni prijeti zadosti o preruseni
2079 C9                 RET
                ;
```



```
23CC                    ORG 23CCH
23CC C35020             JMP INTR        ;obsluha preruseni RST 7
                ;
00FB            CNTR    EQU 0FBH
00FA            DATA    EQU 0FAH
00EF            MH12    EQU 0EFH
2020            KLAV    EQU 2020H
                ;
                        END
```

## 7.11. OBSLUHA PŘERUŠENÍ

### 7.11.1. Teoretický rozbor

Složitější řídicí systémy, které pracují v reálném čase, často potřebují reagovat na vnější události, které mají větší důležitost než právě vykonávaný program. Signály přerušení, generované např. různými periferními zařízeními, časovacími obvody, ale i samotnou základní jednotkou (při zjištěných poruchách) vyvolají obslužné podprogramy, po jejichž skončení se výpočet vrací k původnímu programu, který byl přerušen.

Existuje-li v systému pouze jediná příčina přerušení, je jak struktura přerušovacího systému, tak obsluha žádostí velmi jednoduchá, signál žádosti o přerušení se přímo přivede na vstup INTR mikroprocesoru (viz čl. 7.10.).

Jestliže však může vzniknout několik žádostí o přerušení různé důležitosti, je třeba rozhodovat, v jakém pořadí je obsluhovat. Kromě toho je možné přímo mikroprocesorem přerušení zamaskovat (u 8080 všechna, u jiných – Z80, 8086 aj. – existují i nemaskovatelná přerušení) pomocí instrukce DI a opět povolit pomocí instrukce EI. Tyto instrukce mění stav vnitřního klopného obvodu INTE (po EI je INTE = 1).

Jestliže každému zdroji přerušení přísluší vlastní podprogram pro obsluhu přerušení, hovoříme o tzv. *vektorovém přerušení* (obvodové prostředky generují adresu obslužných podprogramů – *přerušovací vektory*). Naopak, existuje-li pouze jediný obslužný podprogram, který zpětně vyhledává zdroj tohoto přerušení a teprve podle něho volá příslušný obslužný program, mluvíme o *nevektorovém přerušení*.

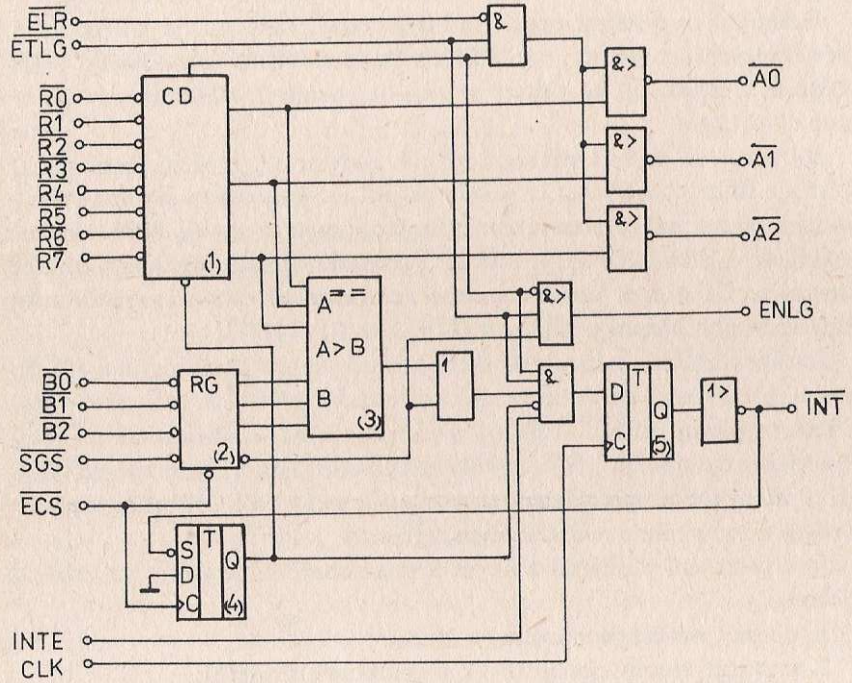
Je-li přerušení povoleno a přijde-li požadavek na obsluhu, pak mikroprocesor:

1. ukončí právě vykonávanou instrukci;
2. vynuluje klopný obvod INTE (zablokuje přerušení);
3. vyšle potvrzovací signál INTA na sběrnici.

Zdroj přerušení jako odpověď na signál INTA vysílá přerušovací vektor (instrukční znak – obvod 3214 vysílá instrukci RST $n$ – volání podprogramu na adrese $8n$, obvod 8259 vysílá přímo instrukci CALL (OCDH) s adresou vybranou z přerušovací tabulky).

Mikroprocesor určenou skokovou instrukci provede okamžitě. Pro jednotlivé obslužné podprogramy jsou v paměti vyhrazena pole 8 slabik (u obvodu 8259 i 4 slabiky) na obslužný program (nebo jeho začátek). U počítače TEMS jsou uživateli dostupná pouze dvě přerušení – RST 6, popř. RST 7, (z nichž je veden odskok na adresy v paměti RAM – 23C9H, popř. 23CCH. Sem musí uživatel vložit skokové instrukce na vlastní obslužné programy, které jsou většinou ukončeny instrukcí RET, zaručující návrat do přerušené části programu.

Umožňuje-li systém přerušit i obslužné podprogramy dalšími, důležitějšími přerušeními, jde o *prioritní (víceúrovňový) systém přerušení.* Může-li se v daný okamžik obsluhovat pouze jediný zdroj přerušení, hovoříme o *jednoúrovňovém (neprioritním) systému.*




*Obr. 42.* Blokové schéma obvodu 3214



Obvod, který umožňuje hlášení signálu s nejvyšší prioritou, se nazývá *prioritní řadič přerušení,* uživatel může k tomuto účelu použít:

1. obvod MH3214, který generuje kód instrukce RST $n$;
2. obvod I8259 (ekvivalent v SSSR – K580IK59), který generuje kód instrukce CALL a adresu obslužného podprogramu, je podstatně složitější a umožňuje mnoho operací nad množinou požadavků o přerušení (např. maskování libovolných přerušení, změny priority aj.).

Vysvětlíme si činnost obvodu 3214 (blokové schéma obvodu je na obr. 42). Základem je prioritní kodér (1) s 8 vstupy $R_i$, které jsou kódovány na tři výstupy. Další částí je stavový registr (2), do něhož se z počítače zapisuje počet povolených přerušení ($p$) počínaje $R_7$ a konče $R_0$. Hodnota L na vstupu $\overline{ECS}$ (u TEMS je ovládán adresou A12) povoluje zápis do tohoto registru. Výstup prioritního kodéru se porovnává v komparátoru (3) s informací ze stavového registru, aby se určilo, které z požadavků jsou povoleny (požadavky $R_i = 8 - p$). Existuje-li takový požadavek, vyvolá se signál INT (přes klopný obvod přerušení (5)) a nastaví se klopný obvod potlačení přerušení (4). Po skončení obsluhy je třeba zapsat znovu masku přerušení, tím se povolí přijetí dalšího požadavku.

Chceme-li využít více požadavků na přerušení než 8, je nutné využít více obvodů 3214 a zapojit vstupy ELR (povolení úrovně), ETLG (povolení této skupiny úrovní) a výstup ENLG (povolení následující skupiny úrovní).

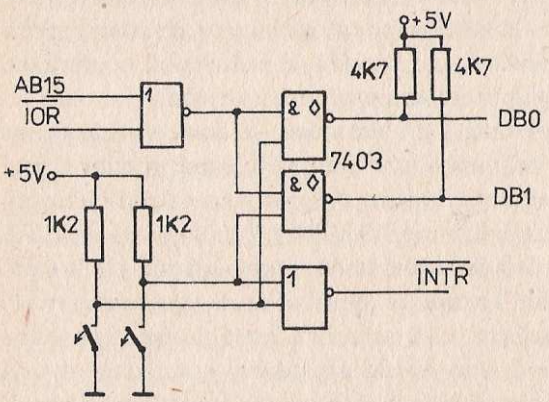
Pro připojení obvodu 3214 na datovou sběrnici se používá obvod 3212, který na signál INT převezme vektor přerušení z obvodu 3214 a vygeneruje signál INTR do mikroprocesoru. V průběhu potvrzovacího signálu INTA z mikropočítače je vektor přiveden na datovou sběrnici a předá adresu obslužného podprogramu.

### 7.11.2. Zapojení

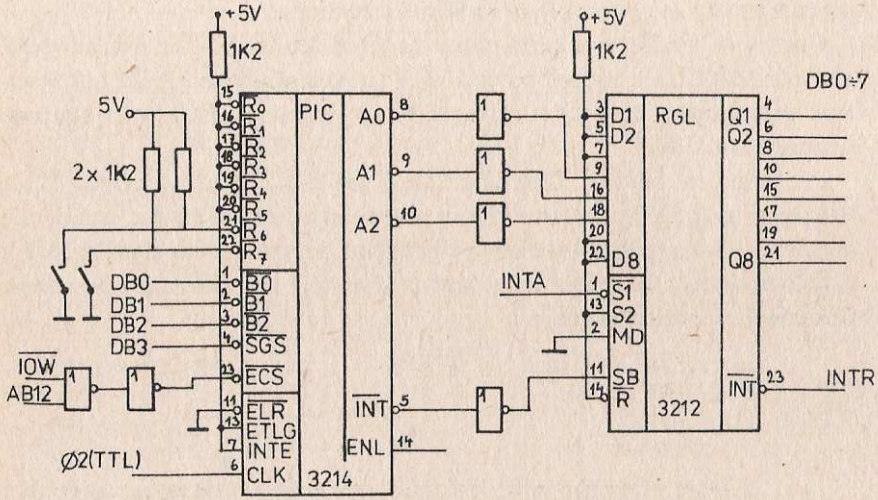
Zdroje přerušení pro obsluhu *nevektorového přerušení* zapojíme podle obr. 43. Spínači 1 a 2 simulujeme dva zdroje přerušení. Tyto požadavky jsou sečteny a přivedeny jako signál INTR přímo do mikropočítače. Druhá část obvodu umožňuje po instrukci IN 7FH dekódovat, který ze spínačů přerušení způsobil.

Zapojení pro obsluhu *vektorového přerušení* je na obr. 44. Mezi obvody 3214 a 3212 jsou vloženy inventory, abychom získali správný kód in-

strukce RST *n* (kód 11×××111B, kde ××× = *n*). Protože na počítači TEMS 80-03 jsou uživateli k dispozici pouze dvě přerušení, spínače, které simulují zdroj přerušení, připojíme na vstupy R6 a R7 řadiče přerušení.




*Obr. 43.* Zapojení pro obsluhu nevektorového přerušení




*Obr. 44.* Zapojení pro obsluhu nevektorového přerušení



## 7.11.3. Měření

### Obsluha nevektorových přerušení

Pro tento případ je charakteristické, že se obsluhuje pouze jediný signál přerušení, a tedy existuje i jediný přerušovací obslužný podprogram. Je třeba rozeznat, který zdroj způsobil dané přerušení a podle toho skočit na odpovídající část přerušovacího podprogramu. Situace se může řešit buď pomocí obvodu, který určí zdroj přerušení (např. multiplexor řízený čítačem, jehož stav se při určení zdroje přerušení zapamatuje), nebo pomocí programu. Programové řešení má několik výhod. Kromě variability obvodu je zde možné splnit i další požadavky obsluhy, např. proměnnou masku žádaných přerušení, možnost měnit prioritu apod. Velkou nevýhodou ale je, zvláště pro větší počet zdrojů, pomalost obsluhy, která často u přerušovacích podprogramů hraje rozhodující roli.

V naší úloze rozlišujeme pouze dva zdroje přerušení, jejich stav se načítá instrukcí IN TLAC (7FH), v načtených datech je každému zdroji vyhrazen jeden bit. Pořadí načítání těchto bitů určuje i prioritu zdrojů.

Využíváme zde obslužné podprogramy INTER6 a INTER7 z následující úlohy. Podle těchto podprogramů se na určitou dobu rozsvěcují číslice 6 nebo 7, podle právě obsluhovaného zdroje přerušení, i když sekvence instrukcí obsluhující stavový registr nebudou využity.

```
2000                    ORG 2000H
007F            TLAC    EQU 7FH         ; brana tlacitek
00EF            P3214   EQU 0EFH        ; stav.reg. 3214
0218            DELAY   EQU 218H
3C00            SEGMEN  EQU 3C00H       ; adresa 7segment. displeje
0232            ERROR   EQU 232H        ; podprogram chyboveho hlaseni
8000            TIME1   EQU 8000H
000A            TIME2   EQU 10

                ; hlavni program pro obe ulohy
2000 3E02       MAIPRG: MVI A,2
2002 D3EF               OUT P3214       ; povol 2 zadosti o preruseni
2004 FB                 EI              ; povol prijeti zadosti o preruseni
2005 C30520     CEKEJ:  JMP CEKEJ

                ; program pro obsluhu RST 7 u nevektoroveho preruseni
2008 F5         NOVECT: PUSH PSW        ; uschovej registry
2009 DB7F               IN TLAC         ; hledej zdroj preruseni
200B 1F                 RAR
200C D22420             JNC INTER7+1    ; tlac.1 , pak prerus. 7
200F 1F                 RAR
2010 D21720             JNC INTER6+1    ; tlac.2 , pak prerus. 6
2013 C33202     NOVEC1: JMP ERROR       ; nestisknuto zadne , pak chyba

23CC                    ORG 23CCH
23CC C30820             JMP NOVEC1      ; prepis adresu odskoku do podprog.
                                          obsluhy preruseni 7
```

Modifikujte program tak, aby bylo možné maskovat přerušení libovolné úrovně (použijte masku, kterou může modifikovat i následující obslužný program).

## Obsluha vektorových přerušení

Opět využijeme dva zdroje přerušení simulované tlačítky, tentokrát je každý zapojen na svůj vlastní vstup prioritního kodéru v obvodu 3214 (obr. 44). Každý zdroj má také svůj obslužný podprogram. Pro odskok do těchto podprogramů je třeba umístit skokové instrukce na tyto podprogramy na adresy 23C9H a 23CCH.

Tyto podprogramy zobrazují asi na dobu 10 s číslice 6, popř. 7, které indikují právě obsluhovaný program. V každém podprogramu jsou na začátku ukládány všechny potřebné registry, aby jejich obsahy zůstaly zachovány pro návrat do přerušeného programu. (Dále je zde čekací smyčka délky asi 0,5 s, která odstraňuje několikanásobnou odezvu na stisk tlačítka, způsobenou zákmity.) Potom nastavíme skupinu přijatých požadavků a povolíme přerušení na mikroprocesoru. U základního programu má zdroj R7 vyšší prioritu než R6. Zdroj R7 může přerušit obsluhu přerušení od zdroje R6. Na konci obslužného podprogramu se nastavuje počet povolených přerušení na dvě, tj. jsou povolena obě přerušení.

Ověřte, jak lze navzájem přerušovat jednotlivé programy.

```
                ; program obsluhy vektoroveho preruseni RST 6, RST 7
23C9                    ORG 23C9H       ; prepis odskokove adresy na
23C9 C31620             JMP INTER6      ; prerus. podprog. - RST 6
23CC C34320             JMP INTER7      ;                  - RST 7

2016                    ORG NOVEC1+3
                ; obsluha preruseni 6
2016 F5         INTER6: PUSH PSW        ; uschovej vsechny potrebne reg.
2017 C5                 PUSH B
2018 D5                 PUSH D
2019 110080             LXI D,TIME1
201C CD1802             CALL DELAY      ; cekej na ustaleni tlacitka
201F 3E01               MVI A,1
2021 D3EF               OUT P3214       ; akceptuj jedinou zadost o preruseni
2023 00                 NOP             ; volno pro zmeny
2024 FB                 EI              ; povol prijeti zadosti o preruseni
2025 3ED7               MVI A,0D7H      ; kod cislice 6
2027 32003C             STA SEGMEN      ; vysli na displej
202A 060A               MVI B,TIME2     ; zobrazuj asi 10 s
202C 110000     CEKEJ6: LXI D,0
202F CD1802             CALL DELAY
2032 05                 DCR B
2033 C22C20             JNZ CEKEJ6
2036 AF                 XRA A
```



```
2037 32003C             STA SEGMEN      ; zhasni segmentovku
203A 3E02               MVI A,2
203C D3EF               OUT P3214       ; prijmi obe zadosti o preruseni
203E 00                 NOP             ; misto na opravy
203F D1                 POP D
2040 C1                 POP B
2041 F1                 POP PSW
2042 C9                 RET

                ; obsluha preruseni 7
2043 F5         INTER7: PUSH PSW        ; uschovej vsechny potrebne reg.
2044 C5                 PUSH B
2045 D5                 PUSH D
2046 110080             LXI D,TIME1
2049 CD1802             CALL DELAY      ; cekej na ustaleni tlacitka
204C 3E00               MVI A,0
204E D3EF               OUT P3214       ; neprijimej zadnou zadost o  preruseni
2050 00                 NOP             ; volno pro zmeny

2051 FB                 EI              ; povol prijeti zadosti o preruseni
2052 3E70               MVI A,070H      ; kod cislice 7
2054 32003C             STA SEGMEN      ; vysli na displej
2057 060A               MVI B,TIME2     ; zobrazuj asi 10 s
2059 110000     CEKEJ7: LXI D,0
205C CD1802             CALL DELAY
205F 05                 DCR B
2060 C25920             JNZ CEKEJ7
2063 AF                 XRA A
2064 32003C             STA SEGMEN      ; zhasni segmentovku
2067 3E02               MVI A,2
2069 D3EF               OUT P3214       ; prijmi obe zadosti o preruseni
206B 00                 NOP             ; misto na opravy
206C D1                 POP D
206D C1                 POP B
206E F1                 POP PSW
206F C9                 RET

                END
```

Pozměňte obslužný podprogram tak, aby programy bylo možné přerušit:

1. požadavkem stejné priority;
2. požadavkem stejné a nižší priority;
3. aby vůbec nešly přerušit.

S využitím znalostí o přerušovacím systému upravte zapojení a program v čl. 7.10. pro sériový přenos dat s obvodem 8251 tak, aby přerušení R6 bylo generováno vysílačem, a přerušení R7 přijímačem.

## Doplňující úloha

Navrhněte zapojení a sestavte program pro soutěž v postřehu: Hlavní program se skládá z čekací smyčky proměnné délky (náhodně určované např. obsahy části paměti), po níž se rozsvítí dioda typu LED. Úkolem soutěžících je stisknout ihned po rozsvícení diody tlačítko. Komu se to

podaří dřív, připíše si bod ke stavu zobrazovanému na displeji. Po změně skóre dioda zhasne a program pokračuje čekací smyčkou. V části dat je možné zobrazovat i dobu reakce vítěze.

Vyřešte použitím blokování přerušení předčasné stisknutí tlačítka, nerozhodné stavy atd. Projeví se nějaká výhoda určitého tlačítka?

## 7.12. AKUSTICKÝ VÝSTUP, GENEROVÁNÍ MELODIE S OBVODEM 8253

### 7.12.1. Teoretický rozbor

V mnoha aplikacích mikropočítačů je nutné použít také akustický výstup, který by upozornil obsluhu na vznik žádané, popř. mimořádné situace a na další požadavky.

Pro generování signálu nejprve použijeme přímo základní obvody mikropočítače: procesor střídavě vysílá úrovně H a L s periodou určenou požadovanou výškou tónu. Je samozřejmé, že obdélníkový signál není zcela „čistý", neboť obsahuje celou řadu vyšších harmonických. Signál se předává z mikroprocesoru přes výstupní obvod (např. 8255) na reproduktor se zesilovačem.

Kromě generování tónu dané frekvence si ukážeme i generování jednoduché posloupnosti tónů (melodie).

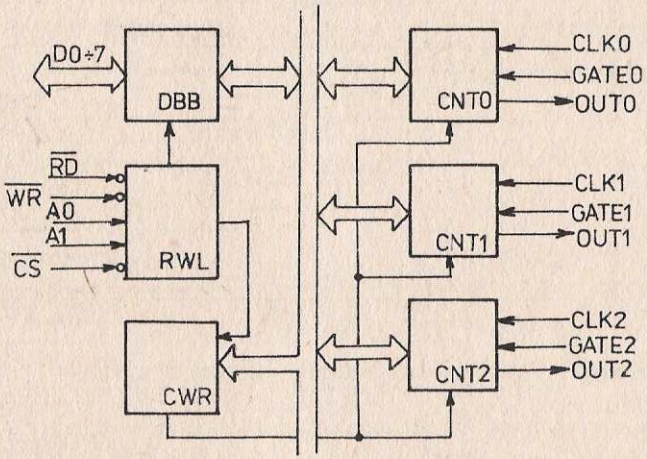
Pro generování průběhů se složitějšími časovými závislostmi (nejen pro zvukový výstup) lze velmi výhodně využít programovatelný časovač 8253, který přebírá od mikroprocesoru řízení většiny časových závislostí, a tím mu umožní věnovat se pouze řídicí činnosti. Obvod 8253 se v praxi velice často používá v řídicích systémech, kde jsou složitější časové vazby.

### 7.12.2. Popis obvodu 8253 – programovatelného časovače

Obvod 8253 (ekvivalent v SSSR – KR580VI53) je programovatelný časovač, který se skládá ze tří nezávislých 16bitových čítačů (blokové schéma obvodu je na obr. 45). Každý z nich může být nezávisle naprogramován v jednom z 6 režimů (modů). Počáteční hodnotu čítačů je možné zadávat buď dvojkově, nebo desítkově. Čítače čítají k nule, činnost každého



z nich může být navíc řízena prostřednictvím příslušného vstupu GATE. Kromě čítačů je součástí obvodu ještě vyrovnávací paměť datové sběrnice (DBB), čtecí/zápisové obvody (RWL) a registr řídicího slova (CWR), který řídí funkce INPUT, OUTPUT a GATE. Při použití obvodu jako čítače lze kdykoliv číst stav čítačů bez zastavení hodinových impulsů. Jejich frekvenční rozsah pro čítače je 0 až 2 MHz.




*Obr. 45.* Blokové schéma vnitřního zapojení obvodu 8253

### Programování obvodu 8253

Všechny režimy jsou plně určeny programem. Nejprve pomocí řídicího slova (viz. obr. 46) vybereme čítač, na který se z CWR vysílá druh režimu a další informace, následují 1 nebo 2 slabiky (byte) počáteční hodnoty do registru daného čítače, který je určován adresami A0, A1. Během čítání je možné číst okamžitý stav čítačů dvěma způsoby; a to:

1. přímo instrukcí IN z daného čítače (určen A0, A1), jsou-li hodinové impulsy zastaveny signálem GATE nebo při pomalém čítání;

2. řídicím slovem RL1, RL0 = 00 přehrajeme okamžitý stav čítače do vnitřního registru, který potom instrukcemi IN načítáme.

| SC1 | SC0 | RL1 | RL0 | M2 | M1 | M0 | BCD |
|---|---|---|---|---|---|---|---|

*Obr. 46.* Řídicí slovo obvodu 8253

*Význam bitů řídicího slova*

SC1,0 – výběr čítače

| | |
|---|---|
| L L | čítač 0 |
| L H | čítač 1 |
| H L | čítač 2 |
| H H | neplatné |

RL1,0 – čtení zápis

| | |
|---|---|
| L L | zápis stavu do střadačů |
| L H | čti/zapiš pouze horní slabiku |
| H L | dolní slabiku |
| H H | čti/zapiš nejprve dolní, pak horní slabiku |

M2.1.0 – volba modu

| | |
|---|---|
| L L L | modus 0 |
| L L H | modus 1 |
| X H L | modus 2 |
| X H H | modus 3 |
| H L L | modus 4 |
| H L H | modus 5 |

BCD – způsob čítání

| | |
|---|---|
| L | binární čítač 16 bitů |
| H | BCD čítač – 4 dekády |

Popis druhů režimu, modů (typické průběhy jsou na obr. 47)

*Modus 0 – přerušení na konci čítání*

Po nahrání hodnoty je výstup ve stavu L, dokud čítač nedokončí čítání, pak výstup přejde do stavu H, dokud se nenastaví nový modus nebo nový stav čítače. Vstup GATE v úrovni L přerušuje čítání.

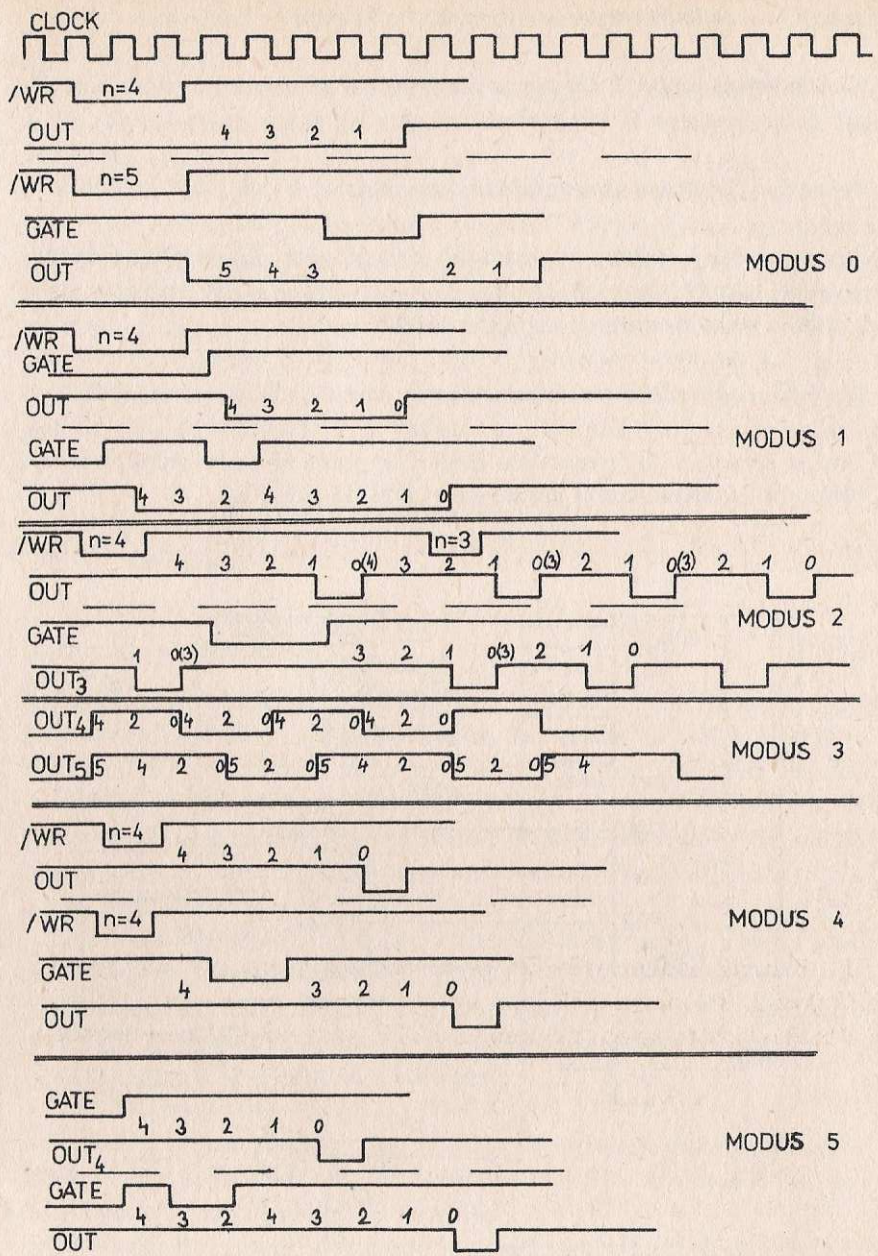
*Modus 1 – opakovaný monostabilní výstup*

Po nahrání hodnoty způsobí první náběžná hrana přechod výstupu do stavu L. Po ukončení čítání přechází výstup do stavu H. Čítání je spouštěno každou náběžnou hranou GATE (vždy od počáteční hodnoty).

*Modus 2 – dělič frekvence (dělič N)*

Po spuštění je výstup ve stavu H, po dočítání přejde na jeden takt do stavu L, perioda výstupních impulsů bude *N*, při změně čítače se ovlivní až následující perioda. Pro GATE = L přejde výstup do stavu H, při přechodu do stavu H se čítání spouští od počáteční hodnoty.






*Obr. 47.* Časové průběhy jednotlivých modů obvodu 8253

*Modus 3 – dělič frekvence se symetrickým výstupem*

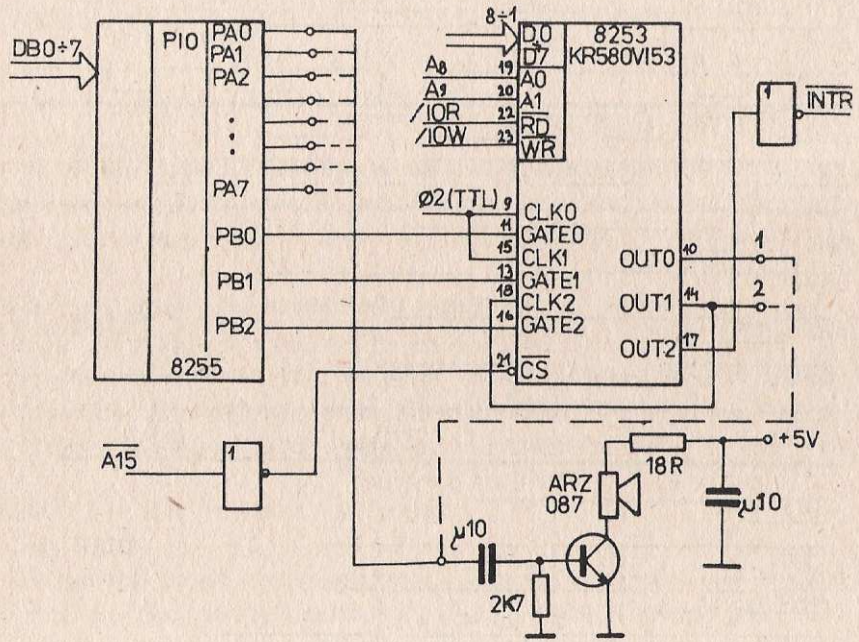
Je obdobou modu 2, výstup je ale ve stavu H pouze polovinu periody (pro lichá *N* je stav H o takt delší než stav L).

*Modus 4 – programově spouštěná synchronizace*

Po spuštění je výstup ve stavu H. Po skončení čítání přejde výstup na jeden takt do stavu L. Změna hodnoty čítače se projeví až v další periodě. Čítání lze zastavit signálem GATE = L.

*Modus 5 – obvodová synchronizace*

Čítač se spouští vždy přechodem GATE do stavu H z počáteční hodnoty. Výstup po skončení čítání jde na jeden takt do stavu L.




*Obr. 48.* Zapojení pro generování zvukových signálů



## 7.12.3. Zapojení

V první úloze budeme generovat signál pouze pomocí obvodu 8255, reproduktor můžeme připojovat střídavě k jednotlivým bitům brány A.

V dalších dvou úlohách využijeme pro generování obdélníkového průběhu čítač 0 obvodu 8253, na jehož výstup OUT0 připojujeme reproduktor.

V rozšiřující úloze, která využívá ještě čítačů 1 a 2 pro generování délky tónu, vyvedeme z výstupu OUT2 ještě přerušovací signál INTR do mikropočítače. Čítače 1 a 2 jsou v tomto případě zřetězeny (mohou tvořit až 32bitový čítač), neboť 16 bitů pro převod frekvence hodinových impulsů na frekvenci asi 16 Hz nestačí. Druhá možnost je dělit frekvenci např. pomocí obvodu MH74193 aj. a výstupní signál použít pro buzení jediného čítače.

## 7.12.4. Měření

### Generování tónů zadaných z klávesnice (s využitím obvodu 8255)

V této úloze budeme generovat tóny, jejichž výška (perioda) bude zadávána vždy dvěma znaky z klávesnice, tón trvá do zadání dalšího tónu. Na vývod A0 obvodu 8255 můžeme generovat tóny od 256 Hz do několika kilohertzů, přesnost tónu se ale ztrácí přibližně u 1 kHz. I tak můžeme využít více než dvě oktávy, což ve většině případů zcela stačí. Na dalších bitech brány A máme současně tóny frekvencí $f/2$, $f/4$, $f/8$ až $f/128$ (tj. tóny o 1 až 7 oktáv nižší).

Po iniciaci obvodu 8255 čekáme na zadání délky periody (výšky tónu) pomocí dvou znaků z klávesnice. Program GTHEX z monitoru je zde rozepsán do dvou částí, abychom mohli po dobu generování tónu kontrolovat, zda již nebyla stisknuta klávesa dalšího tónu. Jsou to:

1. instrukce RST 2 – čtení z klávesnice;
2. zbytek programu – CALL GTHEX + 1.

Perioda je určena čekací smyčkou DRZ0, vlastní tón je generován pomocí čítače v registru B, jehož obsah se vysílá na bránu A obvodu 8255 a který mění hodnoty jednotlivých bitů brány A.

Konstanty pro zadávání jednotlivých tónů jsou v druhé části tabulky 7 (na konci této kapitoly).

```
2000                    ORG 2000H
008B            MOA1BC  EQU 8BH
00F7            PR55    EQU 0F7H
003C            PKLAV   EQU 3CH
00F4            PA55    EQU 0F4H
00F5            PB55    EQU 0F5H
0248            GTHEX   EQU 0248H
003B            NACTI   EQU 003BH       ; pokracovani podprog. RST 2

                ; program pro generovani tonu zadavanych
                ; delkou periody z klavesnice ( s 8255)

2000 3E8B       TON1:   MVI A,MOA1BC    ; nastav 8255 do modu 0
2002 D3F7               OUT PR55        ; brana A vstup, B,C vystup
2004 D7                 RST 2           ; prvni cast instrukce CALL GTHEX
2005 0682       NOVTON: MVI B,82H       ; prebirej 2 znaky v datovem poli
2007 CD4902             CALL GTHEX+1    ; 2. cast instrukce CALL GTHEX
200A 53         CONTON: MOV D,E         ; delku pulperiody tonu do D
200B 15         DRZ0:   DCR D
200C C20B20             JNZ DRZ0        ; drz uroven do poloviny periody
200F 04                 INR B           ; zmen urovne
2010 78                 MOV A,B
2011 D3F4               OUT PA55        ; vysli na reproduktor
2013 DB3C               IN PKLAV        ; stav klavesnice
2015 17                 RAL             ; byla stisknuta ?
2016 D20A20             JNC CONTON      ; ne - pokracuj v tonu
2019 CD3B00             CALL NACTI      ; dokonci podprog. vstupu znaku
201C C30520             JMP NOVTON      ; nacti vysku noveho tonu
```

## Generování melodie uložené v paměti (s využitím obvodu 8253)

K mikroprocesoru připojíme obvod 8253 podle obrázku. Jako zdroj frekvence hodinových impulsů zvolíme signál Φ2 (TTL), reproduktor připojíme na výstup OUTØ. Čítač Ø nastavíme do modu 3 (dělič frekvence se symetrickým výstupem).

Vstupy GATE 0, 1, 2 jsou připojeny na výstup brány B obvodu 8255. Program bude přehrávat melodii uloženou v dané oblasti paměti v následujícím tvaru: 1 slabika délky tónu (v 1/16 taktu) + 2 slabiky délky periody (odpovídá výšce tónu – kódování v první části tabulky 7). Je-li délka = 0, program se ukončí.

Program začíná zadáním čtyř znaků z klávesnice, které určí rychlost přehrávání melodie.

```
                ; program pro prehrani melodie z tabulky (s 8253)
007C            P53CN0  EQU 07CH
007D            P53CN1  EQU P53CN0+1
007E            P53CN2  EQU P53CN0+2
007F            PR53    EQU P53CN0+3
0036            M3CN0   EQU 00110110B
```



```
201F 0684       MEL53:  MVI B,84H
2021 CD4802             CALL GTHEX      ; nacti 4 znaky tempa z klaves.
2024 EB                 XCHG
2025 22FE21             SHLD TEMPO      ; dej do pameti
2028 3E36               MVI A,M3CN0     ; nastav citac 0 na modus 3
202A D37F               OUT PR53

202C 110022             LXI D,HLAS1
202F CD3E20     DALTON: CALL STTON1     ; vygeneruj ton
2032 CD4E20     DRZTON: CALL DELKA      ; cekej 1/16 tonu
2035 05                 DCR B           ; uz patricna delka ?
2036 C23220             JNZ DRZTON      ; ne - cekej stale
2039 C32F20             JMP DALTON      ; ano - vysli dalsi ton

203C F1         KONECO: POP PSW         ; vyber a zrus navrat. adresu
203D CF         KONEC:  RST 1

203E 1A         STTON1: LDAX D
203F B7                 ORA A
2040 CA3C20             JZ KONECO       ; pro delku tonu=0 ukonci melodii
2043 47                 MOV B,A         ; delka v 1/16 tonu
2044 13                 INX D
2045 1A                 LDAX D
2046 D37C               OUT P53CN0      ; dolni slabika delice
2048 13                 INX D
2049 1A                 LDAX D
204A D37C               OUT P53CN0      ; horni slabika na 8253
204C 13                 INX D
204D C9                 RET

204E E5         DELKA:  PUSH H
204F 2AFE21             LHLD TEMPO
2052 2B         DECRHL: DCX H
2053 7C                 MOV A,H
2054 B5                 ORA L

2055 C25220             JNZ DECRHL
2058 E1                 POP H           ; obnov puvodni registry
2059 C9                 RET
```

Program je koncipován tak, aby jej bylo možné rozšířit na generování dvouhlasé melodie (2. hlas může být uložen v jiné oblasti paměti), je jen třeba modifikovat program STTON1 pro druhý hlas a v části DRZTON kontrolovat stav čítačů délky obou tónů – při nulovém obsahu čítače spustit znovu daný tón. Navrhněte tuto modifikaci!

Rozmyslete, jak je možné vygenerovat pauzu (vstup GATE, popř. tóny extrémních frekvencí). Navrhněte tuto modifikaci programu (pauzu zadávejte do paměti jako tón s periodou rovnou nule).

## Generování melodie zadané z klávesnice (s využitím obvodu 8253)

Využijeme zapojení z předcházející úlohy. Po nastavení obvodu 8253 se čeká na zadání délky tónu do datového pole (opět v 1/16 základního tónu) a výšky (periody) do adresového pole. Po uplynutí daného tónu

se čítač zastaví pomocí impulsu GATE0 = 0. Tento impuls je generován pomocí bitu 0 brány B obvodu 8255 a čeká se na zadání dalšího tónu. Zadáním nulové délky program končí.

```
                ; program pro generovani melodie zadane
                ; z klavesnice ( s 8253 )

0000            MOABIC  EQU 00H

205A 210014     MEL153: LXI H,1400H     ; delka celeho tonu asi 0.96 s
205D 22FE21             SHLD TEMPO
2060 3E36               MVI A,M3CN0
2062 D37F               OUT PR53        ; citac 0 do modu 3
2064 3E00               MVI A,MOABIC
2066 D3F7               OUT PR55
2068 3E01               MVI A,1
206A D3F5               OUT PB55        ; GATE 0 do 1

206C 0682       NOV1N1: MVI B,82H
206E CD4802             CALL GTHEX      ; delku tonu z dat. pole
2071 D5                 PUSH D
2072 0684               MVI B,84H
2074 CD4802             CALL GTHEX      ; delku pulperiody do adr. pole
2077 C1                 POP B           ; delka tonu do C
2078 3E01               MVI A,1
207A D3F5               OUT PB55        ; znovu GATE 0 do log. 1
207C 7B                 MOV A,E
207D D37C               OUT P53CN0      ; dolni slabika
207F 7A                 MOV A,D
2080 D37C               OUT P53CN0      ; horni slabika do citace 0
2082 79                 MOV A,C
2083 B7                 ORA A
2084 CA3D20             JZ KONEC
2087 CD4E20     DRZ1:   CALL DELKA
208A 0D                 DCR C
208B C28720             JNZ DRZ1
208E AF                 XRA A
208F D3F5               OUT PB55        ; impuls na GATE 0
2091 C36C20             JMP NOV1N1      ; dalsi ton
```

Upravte program tak, aby hodnoty generovaných tónů byly současně ukládány do paměti tak, aby mohly být přehrávány programem v předchozí úloze.

Napište program, který zapíše (případně i hraje) melodii zadanou takto:

a) 1 znak jako délka tónu v 1/16 (pro celý tón = 16/16 – zadávej 0) – zobrazuj jej v poli dat – zadávání ukonči stiskem jiné klávesy;

b) dvouznakové pole – vstup pomocí GTHEX – výška tónu (c = 00, cis = 01, d = 02 atd.). S pomocí tabulky TABTÓN ukládejte melodii do paměti tak, aby se dala přehrát pomocí programu předchozí úlohy.



## Generování melodie s přerušením

Aby mikroprocesor nemusel stále kontrolovat délku tónu a mohl se věnovat jiné činnosti, lze využít zbývající čítače obvodu 8253 a přerušovacího systému pro generování přerušení po přesně určené době (opět vždy po 1/16 taktu). Toto přerušení má tutéž funkci jako podprogram DELKA v předchozích programech.

```
                ; program pro prehrani melodie ( s 8253 + prerusenim)

0076            M3CN1   EQU 01110110B
00A0            M0CN2   EQU 10100000B

2094 0682       INTMEL: MVI B,82H
2096 CD4802             CALL GTHEX      ; nacti 2 znaky tempa
2099 7B                 MOV A,E
209A 32FE21             STA TEMPO
209D 3E36               MVI A,M3CN0     ; citac 0 do modu 3
209F D37F               OUT PR53
20A1 3E76               MVI A,M3CN1     ; citac 1 do modu 3
20A3 D37F               OUT PR53
20A5 3EA0               MVI A,M0CN2
20A7 D37F               OUT PR53        ; citac 2 do modu 0
20A9 210004             LXI H,400H
20AC 7D                 MOV A,L
20AD D37D               OUT P53CN1
20AF 7C                 MOV A,H
20B0 D37D               OUT P53CN1      ; citac 1 jako delic 400H
20B2 CDD020             CALL RST7B      ; nastav citac 2, povol preruseni
20B5 110022             LXI D,HLAS1
20B8 1A         DALTN1: LDAX D
20B9 B7                 ORA A           ; delka z pameti
20BA CA3D20             JZ KONEC        ; pro nulovou delku ukonci
20BD 47                 MOV B,A
20BE 13                 INX D
20BF 1A                 LDAX D
20C0 D37C               OUT P53CN0      ; dolni slabika pulperiody
20C2 13                 INX D
20C3 1A                 LDAX D
20C4 D37C               OUT P53CN0      ; horni slabika do 8253
20C6 13                 INX D
20C7 78         CEKEJ:  MOV A,B
20C8 B7                 ORA A           ; uz B=0 ?
20C9 C2C720             JNZ CEKEJ       ; nahrazuje hlavni program
20CC C32F20             JMP DALTON      ; jdi na dalsi ton

20CF 05         RST7A:  DCR B           ; dekrementuj citac delky
20D0 F5         RST7B:  PUSH PSW
20D1 3AFE21             LDA TEMPO       ; nastav znovu citac 2
20D4 D37E               OUT P53CN2

20D6 F1                 POP PSW
20D7 FB                 EI              ; povol preruseni
20D8 C9                 RET

21FE                    ORG 21FEH
21FE            TEMPO:  DS 2
2200            HLAS1:  DS 255
```

```
23CC                ORG 23CCH
23CC C3CF20         JMP RST7A       ; odskok z RST 7

                    END
```

Přerušovací impulsy dostaneme z hodinových impulsů Φ2 s využitím čítačů 1 a 2. Čítač 1 (v modu 3) slouží pouze jako dělič frekvence se symetrickým výstupem a čítač 2 (v modu 0) generuje přerušovací impulsy, které dostává mikroprocesor jako přerušení úrovně 7 (přes obvod 3214 nebo přímo připojením výstupu INTA obvodu 8228 přes rezistor na napětí +12 V).

*Tabulka 7.* Konstanty pro generování tónů

| Tón | Frekvence (Hz) | Konstanty 2. úlohy | | Konstanty 1. úlohy | |
|---|---|---|---|---|---|
| | | dekad | hexa | dekad | hexa |
| C | 130.8 | 16 655 | 3D27 | 518 | |
| C+ | 138.6 | 14 777 | 39B9 | 488 | |
| D | 146.8 | 13 947 | 367B | 461 | |
| D+ | 155.6 | 13 165 | 336D | 435 | |
| E | 164.8 | 12 426 | 308A | 410 | |
| F | 174.6 | 11 728 | 2DD0 | 387 | |
| F+ | 185.0 | 11 070 | 2B3E | 365 | |
| G | 196.0 | 10 449 | 28D1 | 344 | |
| G+ | 207.7 | 9 862 | 2686 | 324 | |
| A | 220.0 | 9 309 | 245D | 306 | |
| A+ | 233.1 | 8 786 | 2252 | 289 | |
| H | 246.9 | 8 293 | 2065 | 272 | |
| C | 261.6 | 7 827 | 1E93 | 257 | cca 00 |
| C+ | 277.2 | 7 388 | 1CDC | 242 | F2 |
| D | 293.7 | 6 973 | 1B3D | 228 | E4 |
| D+ | 311.1 | 6 582 | 19B6 | 215 | D7 |
| E | 329.6 | 6 213 | 1845 | 203 | CB |
| F | 349.2 | 5 864 | 16E8 | 191 | BF |
| F+ | 370.0 | 5 535 | 159F | 180 | B4 |
| G | 392.0 | 5 224 | 1468 | 170 | AA |
| G+ | 415.3 | 4 931 | 1343 | 160 | 0A |
| A | 440.0 | 4 654 | 122E | 151 | 97 |
| A+ | 466.2 | 4 393 | 1129 | 142 | 8E |
| H | 493.9 | 4 146 | 1032 | 134 | 86 |
| C | 523.2 | 3 914 | F4A | 126 | 7E |
| C+ | 554.4 | 3 694 | E6E | 119 | 77 |



Tab. 7 – pokračování

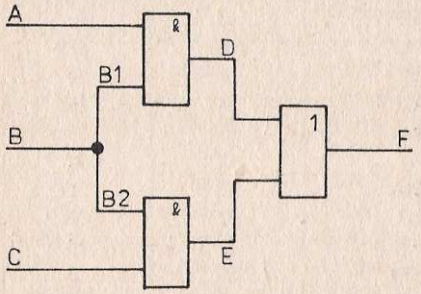
| Tón | Frekvence (Hz) | Konstanty 2. úlohy | | Konstanty 1. úlohy | |
|---|---|---|---|---|---|
| | | dekad | hexa | dekad | hexa |
| D | 587.3 | 3 486 | D9E | 112 | 70 |
| D+ | 622.3 | 3 291 | CDB | 105 | 69 |
| E | 659.3 | 3 106 | C22 | 99 | 63 |
| F | 698.5 | 2 932 | B74 | 93 | 5D |
| F+ | 740.0 | 2 767 | ACF | 88 | 58 |
| G | 784.0 | 2 612 | A34 | 83 | 53 |
| G+ | 830.6 | 2 465 | 9A1 | 78 | 4E |
| A | 880.0 | 2 327 | 917 | 73 | 49 |
| A+ | 932.3 | 2 196 | 894 | 69 | 45 |
| H | 987.8 | 2 073 | 819 | 65 | 41 |
| C | 1 046.5 | 1 957 | 7A5 | 61 | 3D |
| C+ | 1 108.7 | 1 847 | 737 | 57 | 39 |
| D | 1 174.7 | 1 743 | 6CF | 54 | 36 |
| D+ | 1 244.5 | 1 645 | 66D | 51 | 33 |
| E | 1 318.5 | 1 553 | 611 | 48 | 30 |
| F | 1 396.9 | 1 466 | 5BA | 45 | 2D |
| F+ | 1 480.0 | 1 383 | 567 | 42 | 2A |
| G | 1 568.0 | 1 306 | 51A | 39 | 27 |
| G+ | 1 661.2 | 1 232 | 4D0 | 37 | 25 |
| A | 1 760.0 | 1 163 | 48B | 35 | 23 |
| A+ | 1 864.6 | 1 098 | 44A | 32 | 20 |
| H | 1 975.5 | 1 036 | 40C | 30 | 1E |

# 8. Diagnostika a modelování — cvičení

## 8.1. INTUITIVNÍ ZCITLIVĚNÍ CESTY

### Zadání

Intuitivním zcitlivěním cesty sestavte úplný diagnotický test pro obvod zadaný na obr. 49.




*Obr. 49.* Příklad obvodu s větvením

### Úvod

S vytvářením diagnostických testů jsme se seznámili v čl. 4.3, kde byl také uveden příklad sestavení testu pro obvod bez větvení. V obvodech s větvením, což je náš případ, se zcitlivění cesty komplikuje tím, že z některých vodičů vede na primární výstup více cest. Volbu určité citlivé cesty ale nedokážeme posoudit jinak než odvozením celého kroku testu. Vzhledem k tomu, že ve složitějších obvodech končí celá řada takovýchto pokusů nezdarem, musíme právě v této fázi velmi uplatnit intuici, na niž se odvolává název metody. Nezdary jsou způsobeny tím, že při neúmyslném zcitlivění většího počtu cest může dojít k jejich vzájemnému zablokování. To nastává v případě, kdy větve po průchodu určitým počtem logických členů vstupují do společného logického členu, tzn. znovu se sbíhají. Změna hodnoty signálu před rozvětvením se tedy nemůže přenášet na primární výstup. Přesto ale některé změny hodnoty signálu v některých z těchto větví mohou



být v tomto případě detektovány. Pro usnadnění práce se při intuitivním zcitlivování většinou omezujeme jen na jednu cestu. Omezíme-li se ale jen na jednu citlivou cestu, nemusíme najít řešení, protože některé poruchy v obvodech s větvením jsou detektovatelné pouze zcitlivěním většího počtu cest.

### Postup cvičení

Vraťme se ale k našemu příkladu. Kromě primárních vstupů *A, B, C* jsou na obr. 49 označeny také větve *B1* a *B2*, na něž se vstup B větví. Předpokládáme, že se tento vodič větví uvnitř testované jednotky. Podle druhé věty z čl. 4.3 bude úplnost testu zaručena, bude-li test kromě poruch t0 a t1 na vodičích *A, B, C* detektovat i tyto poruchy na vodičích *B1* a *B2*. První krok testu sestavíme takto:

1. na vodiči *A* volíme poruchu t0;
2. na tento vodič přivedeme hodnotu 1;
3. citlivou cestu z vodiče *A* na vodič *D* zajistí hodnota $B = 1$, citlivou cestu z *D* na *F* zajistí $E = 0$;
4. hodnotu $E = 0$ můžeme realizovat třemi způsoby, a to dvojicemi hodnot $BC = 10$, $BC = 01$ nebo $BC = 00$; vzhledem k tomu, že *B* musí být rovno 1, můžeme použít pouze $BC = 10$;
5. kromě poruchy t0 na vodiči *A* jsou tímto krokem testu detektovány i poruchy t0 na vodičích *B* a *B1*.

Další krok testu sestavujeme pro poruchu t1 na vodiči *A*. Citlivou cestu můžeme zajistit hodnotou $B = 1$. Hodnota *C* může být libovolná, ale pokud zvolíme $C = 0$, detektujeme i poruchu t1 na vodiči *C*. Při detekci poruchy t1 na vodiči *B* můžeme k realizaci citlivé cesty z *B* na *F* použít např. kombinaci $AC = 10$. Tímto krokem testu je dále detektována porucha t1 na vodiči *B1*. Čtvrtý krok testu sestavujeme pro detekci poruchy t0 na vodiči *B2*. Citlivou cestu *B2F* zajistí např. kombinace $AC = 01$. Tento krok dále detektuje poruchy t0 na vodičích *B* a *C*. Zbývá pokrýt poruchu t1 na vodiči *B2*. Citlivou cestu zajistíme opět kombinací $AC = 01$. Krok dále detektuje i poruchu t1 na vodiči *B*.

Výsledný test je zapsán v tab. 8. Můžeme se v ní přesvědčit o nutnosti testovat každou větev samostatně. První čtyři kroky totiž detektují poruchy na všech primárních vstupech, ale porucha t1 na vodiči *B2* přitom zůstává nedetektována. Pro úplnost je třeba dodat, že uvedený test není minimální,

protože dosazením $C = 1$ ve třetím kroku by byly poruchy t1 detektovány na vodičích *B1* a *B2* současně. Tím by se pátý krok stal zbytečným. Takto zkrácený test je uveden v tab. 9.

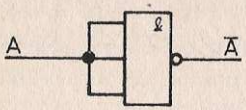
*Tabulka 8.* Detekční test s nadbytečným krokem

| *A* | *B* | *C* | *F* | *A* | *B* | *B1* | *B2* | *C* |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | t0 | t0 | t0 | | |
| 0 | 1 | 0 | 0 | t1 | | | | t1 |
| 1 | 0 | 0 | 0 | | t1 | t1 | | |
| 0 | 1 | 1 | 1 | | t0 | | t0 | t0 |
| 0 | 0 | 1 | 0 | | t1 | | t1 | |

*Tabulka 9.* Zkrácený detekční test

| *A* | *B* | *C* | *F* | *A* | *B* | *B1* | *B2* | *C* |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | t0 | t0 | t0 | | |
| 0 | 1 | 0 | 0 | t1 | | | | t1 |
| 1 | 0 | 1 | 0 | | t1 | t1 | t1 | |
| 0 | 1 | 1 | 1 | | t0 | | t0 | t0 |

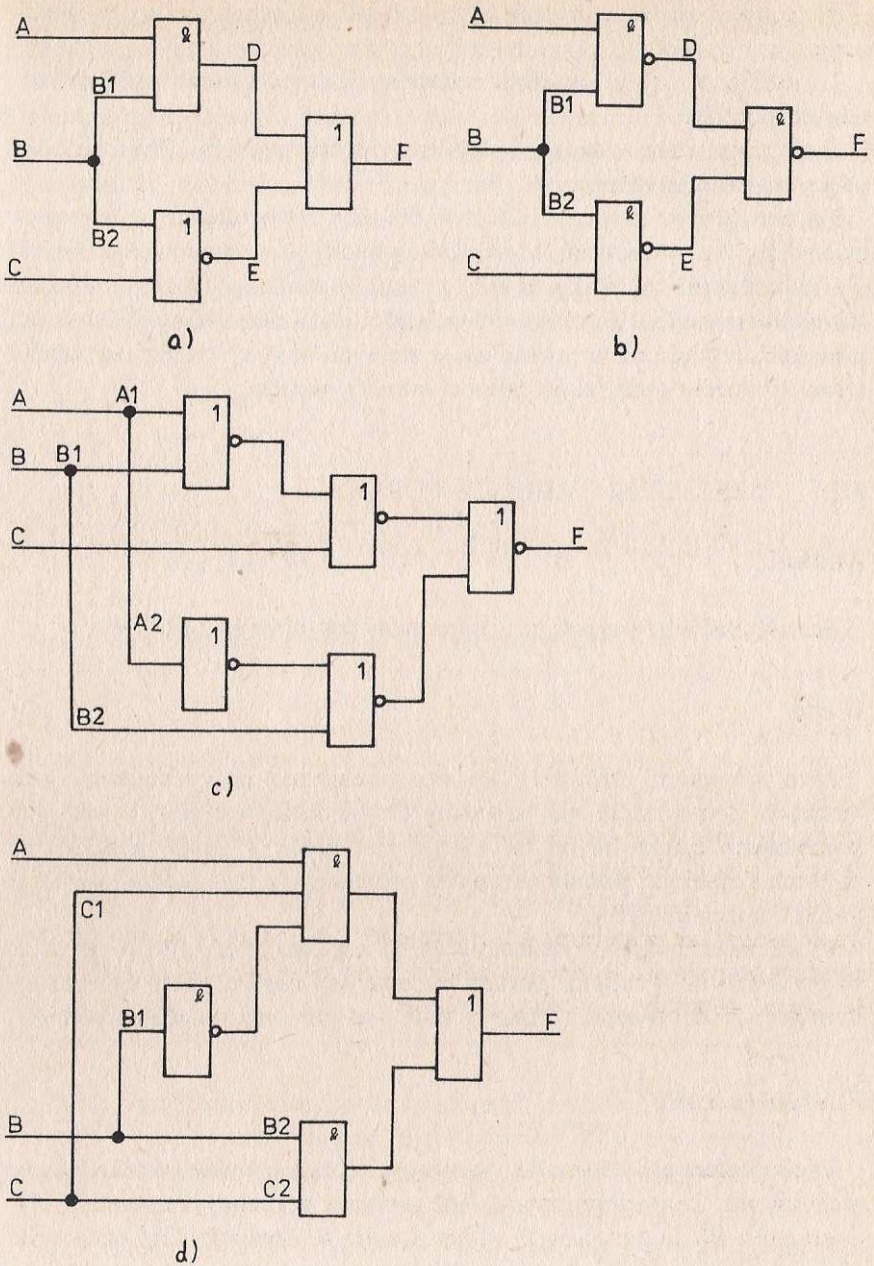
Obvody, které obsahují nadbytečné (redundantní) součástky (vodiče, logické členy), mohou diagnostiku velmi zkomplikovat. Názvem *redundantní* označujeme součástku, kterou lze z logického obvodu vyjmout a nahradit ji zdrojem konstantního signálu 0 nebo 1, aniž by se tím změnila výstupní funkce obvodu. Příklad redundance vidíme na obr. 50, kde je trojvstupový člen NAND použit ve funkci invertoru. Je zřejmé, že kterékoli dva ze vstupů lze připojit na zdroj trvalé jedničky, aniž by se tím funkce invertoru změnila. Za redundantní můžeme prohlásit kterékoli dva ze všech tří vstupů, ale nikdy ne všechny současně.




*Obr. 50.* Příklad obvodu s redundancí

Vzhledem k tomu, že redundantní součástku můžeme vyjmout z obvodu, aniž by se změnila výstupní funkce obvodu, nelze ani poruchy t0 a t1 redundantních součástek detektovat na výstupu. To znamená, že test obvodu s redundantními součástkami nemůže být úplný.






*Obr. 51.* Příklady obvodů pro procvičování sestavování testů

Při generování testu pro obvody s redundancí se za správný výsledek považuje:

1. zjištění, že obvod obsahuje redundanci, a výčet všech nedetektovatelných poruch;

2. sestavení testu, který je úplný pro všechny poruchy. Tento postup odpovídá požadavkům praxe.

Na obr. 51 jsou uvedeny další čtyři obvody, které můžete použít k procvičování sestavování testů. Mezi těmito obvody jsou také obvody obsahující redundantní součástky. Každý z testů je možné vytvořit v několika různých variantách. Z tohoto důvodu zde řešení neuvádíme. Následující praktická cvičení budou prováděna se stejnými obvody. Správnost vašeho řešení si můžete ověřit např. pomocí tabulky poruch.

## 8.2. SESTAVENÍ TABULKY PORUCH

### Zadání

Sestavte tabulku poruch pro jednoduchý test obvodu z obr. 49.

### Úvod

Jako *jednoduchý (triviální) test* označujeme test na vyzkoušení všech možných funkcí, které má testovaný obvod realizovat. Pro $n$-vstupový kombinační logický obvod má takový test délku $2^n$ kroků a skládá se ze všech kombinací hodnot vstupních proměnných. Pro zadaný obvod je to test maximální délky.

Pro označení poruch zavedeme zkrácené označení, např. $A/0$ pro poruchu t0 na vodiči $A$. V tabulce poruch budeme pro úsporu místa vynechávat lomítko a číslici 0 nebo 1 budeme psát pod písmeno označující vodič.

### Postup cvičení

Východiskem při sestavování tabulky poruch je obvykle tabulka funkčních hodnot. Tu vyplníme na základě simulace poruchových stavů. V obvodu (obr. 49) je označeno 8 vodičů (vodič *B1* budeme dále označovat písmenem $G$ a vodič *B2* písmenem $H$). Sestavujeme-li tabulku pro všechny



poruchy typu t, musíme počítat s 16 poruchami. Pro jednotlivé poruchy bude tedy v tabulce 16 sloupců (pravá část tabulky). V levé části tabulky je vedle sloupce s kombinacemi hodnot vstupních proměnných sloupec s hodnotou výstupu $F$ v bezporuchovém stavu (viz tab. 10). Při odvozování hodnot výstupní proměnné při výskytu jednotlivých poruch, tj. při simulaci poruch, je účelné vycházet z algebraického popisu funkce testovaného obvodu, do něhož za příslušné proměnné dosazujeme konstanty. Strukturu našeho obvodu můžeme popsat vztahem $F = AB + BC$. Do tohoto vztahu budeme postupně dosazovat hodnoty $A = 0$, $A = 1$, $B = 0,\ldots,$ $C = 1$. Tak získáme vztahy pro funkci degenerovanou přítomností jednotlivých poruch.

$A = 0$: $F = BC$
$A = 1$: $F = B + BC = B$
$B = 0$: $F = 0$
$B = 1$: $F = A + C$
$C = 0$: $F = AB$
$C = 1$: $F = AB + B = B$

*Tabulka 10.* Tabulka funkčních hodnot pro jednotlivé poruchy

| $A$ | $B$ | $C$ | $F$ | $A$ 0 | $A$ 1 | $B$ 0 | $B$ 1 | $C$ 0 | $C$ 1 | $D$ 0 | $D$ 1 | $E$ 0 | $E$ 1 | $F$ 0 | $F$ 1 | $G$ 0 | $G$ 1 | $H$ 0 | $H$ 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

Na základě těchto vztahů můžeme vyplnit prvních 6 sloupců pravé části tabulky. Např. první sloupec (pro poruchu $A/0$) získáme dosazením hodnot vstupních proměnných do vztahu $F = BC$, druhý sloupec ($A/1$) je určen vztahem $F = B$, tzn. opíšeme do něho hodnoty proměnné $B$, třetí sloupec ($B/0$) bude obsahovat samé nuly atd. Pro algebraický zápis dalších poruchových stavů musíme použít substituci odpovídající sché-

matu zapojení. Výstupní proměnnou musíme vždy vyjádřit jako funkci té proměnné, za kterou máme dosadit konstantu. Např. pro reprezentaci poruch vodiče $D$ potřebujeme vztah $F = D + BC$. Tak dostaneme další vztahy, které využijeme stejným způsobem pro vytváření dalších sloupců tabulky poruch. Můžeme psát

| | | |
|---|---|---|
| $F = D + BC$ | $D = 0$: | $F = BC$ |
| | $D = 1$: | $F = 1$ |
| $F = AB + E$ | $E = 0$: | $F = AB$ |
| | $E = 1$: | $F = 1$ |
| $F = F$ | $F = 0$: | $F = 0$ |
| | $F = 1$: | $F = 1$ |
| $F = AG + BC$ | $G = 0$: | $F = BC$ |
| | $G = 1$: | $F = A + BC$ |
| $F = AB + HC$ | $H = 0$: | $F = AB$ |
| | $H = 1$: | $F = AB + C$ |

V tabulce 10 je několik shodných sloupců. Týká se to především sloupců odpovídajících poruchám t0 na vývodech součinových logických členů ($A/0$, $D/0$, $G/0$ a $C/0$, $E/0$, $H/0$) a sloupců odpovídajících poruchám t1 na vývodech součtového logického členu ($D/1$, $E/1$, $F/1$). Poruchy, jimž odpovídají shodné sloupce v tabulce, nazýváme *nerozlišitelné*. Skupiny nerozlišitelných poruch, které jsme právě popsali, nazýváme *třídy ekvivalence* poruch jednotlivých logických členů. Pro každý $n$-vstupový logický člen má třída ekvivalence $n + 1$ prvků (jedna porucha pro každý vývod). Strukturu těchto tříd pro základní logické členy popisuje tab. 11.

V tabulce 10 lze kromě vyjmenovaných tříd ekvivalence najít ještě další shodné sloupce, a to $A/1$, $C/1$ a $B/0$, $F/0$. V tomto případě jde o nerozlišitelné poruchy na vstupech různých logických členů, s jejichž výskytem je v logických obvodech také třeba počítat.

Při sestavování tabulky poruch je účelné respektovat existenci nerozlišitelných poruch, přičemž třídy ekvivalence jednotlivých členů lze velmi snadno určit předem. Jestliže v tab. 10 nahradíme každou třídu ekvivalence poruchou výstupu příslušného členu, dvojici $A/1$, $C/1$ poruchou $A/1$ a dvojici $B/0$, $F/0$ poruchou $F/0$, získáme zjednodušenou tabulku funkčních hodnot. Příkladem takové tabulky je tab. 12. Kombinace hodnot vstupních proměnných jsou v ní zakódovány jako číslo kroku ($K$). Z této tabulky již snadno vytvoříme tabulku poruch. Provedeme to operací nonekvivalence všech poruchových sloupců se sloupcem $F$. Dostaneme tak tabulku 13.



*Tabulka 11.* Třídy ekvivalence poruch pro základní logické členy

| Typ | Třída ekvivalence | |
|---|---|---|
| | vstupy | výstupy |
| AND | t0 | t0 |
| NAND | t0 | t1 |
| OR | t1 | t1 |
| NOR | t1 | t0 |

*Tabulka 12.* Tabulka funkčních hodnot bez nerozlišitelných poruch

| $K$ | $F$ | $A$ 1 | $B$ 1 | $D$ 0 | $E$ 0 | $F$ 0 | $F$ 1 | $G$ 1 | $H$ 1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

*Tabulka 13.* Tabulka poruch

| $K$ | $A$ 1 | $B$ 1 | $D$ 0 | $E$ 0 | $F$ 0 | $F$ 1 | $G$ 1 | $H$ 1 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | |
| 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ● |
| 3 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | ● |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| 6 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | ● |
| 7 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | × | | × | × | ■ | ■ | | | |

Obdobným způsobem můžeme sestavit tabulku poruch pro jednoduché testy obvodů zadaných na obr. 51. Vytvořené tabulky poruch budeme v následujícím praktickém cvičení používat pro minimalizaci testu.

Důležitým předpokladem je úplnost seznamu poruch, pro něž tabulku poruch sestavujeme. Na základě vět o úplnosti testu kombinačních logických obvodů se můžeme pro diagnostický test omezit na poruchy primárních vstupů a větví za každým bodem větvení. Tabulka poruch sestavená pro takto redukovaný seznam poruch nám umožní odvodit minimální diagnostický test. Pokud však chceme odvodit lokalizační test, musíme sestavit tabulku poruch pro všechny poruchy, včetně poruch vnitřních vodičů obvodu a primárních vstupů, tak jako jsme to provedli v našem případě.

Při sestavování tabulky poruch pro obvod s několika výstupy dostáváme opět dvourozměrné pole jednobitových hodnot. Každý prvek pak vyjadřuje shodu nebo neshodu celého výstupního vektoru, odvozeného pro příslušný krok testu a pro simulovanou poruchu s výstupním vektorem odpovídajícím bezporuchovému stavu.

Pro usnadnění práce při sestavování tabulky poruch můžeme využít následující program v jazyku BASIC. Tento program čte tabulku funkčních hodnot, vypisuje seznam nedetektovatelných poruch a nerozlišitelných poruch a vytváří tabulku poruch. Program také provádí výpočty, které bude možné využít při následujícím praktickém cvičení, tj. při minimalizaci testu. Tento program je možné využít i pro obvody s jedním výstupem. Byl vytvořen pro počítač IQ 151. Všechny výstupy v tomto programu se provádějí na tiskárně. Pokud nemáte počítač vybaven tiskárnou, je potřeba příkazy LPRINT změnit na příkazy PRINT.

```
10 REM VYTVORENI TABULKY PORUCH
15 CLEAR 500
20 READ VP: FOR I=1 TO VP: READ VP$(I): NEXT I
30 READ VY$
40 READ VO: DIM VO$(VO): FOR I=1 TO VO: READ VO$(I): NEXT I
50 READ PK: DIM FH(PK,2*VO),VK$(PK),VR(PK),F1(PK,2*VO)
60 FOR I=1 TO PK: READ VK$(I),FH(I,0)
70 FOR J=1 TO 2*VO: READ FH(I,J): F1(I,J)=FH(I,J): NEXT J: NEXT I
80 REM VYPIS TABULKY FUNKCNICH HODNOT
85 LPRINT "OPIS TABULKY FUNKCNICH HODNOT"
86 LPRINT "*****************************":LPRINT
90 FOR I=1 TO VP: LPRINT VP$(I);: NEXT I: LPRINT " "VY$" ";
100 FOR I=1 TO VO: LPRINT VO$(I);VO$(I);: NEXT I: LPRINT
110 LPRINT TAB(VP+3);: FOR I=1 TO VO: LPRINT "01";: NEXT I: LPRINT
120 FOR I=1 TO VP+3+2*VO: LPRINT "-";: NEXT I: LPRINT
130 FOR I=1 TO PK: LPRINT VK$(I)" "STR$(FH(I,0))" ";
140 FOR J=1 TO 2*VO: LPRINT STR$(FH(I,J));: NEXT J: LPRINT: NEXT I
150 LPRINT: LPRINT "NEDETEKTOVATELNE PORUCHY:": P=0: R=0
```



```
160 FOR I=1 TO 2*VO: FOR J=1 TO PK
170 IF FH(J,0)<>FH(J,I) THEN 200
180 NEXT J: P=P+1: A=INT((I+1)/2): R=R+1: FH(0,I)=1
190 LPRINT VO$(A)"/"STR$(I+1-2*A)" ";
200 NEXT I: IF P=0 THEN LPRINT"ZADNE"
210 LPRINT: LPRINT "SEZNAM NEROZLISITELNYCH PORUCH:"
220 FOR I=1 TO 2*VO-1: IF FH(0,I)=1 THEN 280
230 P=0: FOR J=I+1 TO 2*VO: IF FH(0,J)=1 THEN 270
240 FOR K=1 TO PK: IF FH(K,I)<>FH(K,J) THEN 270
245 NEXT K: FH(0,J)=1: R=R+1
250 IF P=0 THEN A=INT((I+1)/2):LPRINT VO$(A)"/"STR$(I+1-2*A)" ";
260 P=P+1: A=INT((J+1)/2):LPRINT VO$(A)"/"STR$(J+1-2*A)" ";
270 NEXT J: IF P<>0 THEN LPRINT
280 NEXT I: LPRINT
290 REM VYPIS ZKRACENE TABULKY FUNKCNICH HODNOT
300 LPRINT "ZKRACENA TABULKA FUNKCNICH HODNOT"
310 LPRINT "*****************************": LPRINT
320 GOSUB 800
330 FOR I=1 TO 2*VO: IF FH(0,I)=1 THEN 350
340 FOR J=1 TO PK: FH(J,I)=(FH(J,0)+FH(J,I))AND 1: NEXT J
350 NEXT I: LPRINT
360 LPRINT "TABULKA PORUCH"
370 LPRINT "**************": LPRINT
380 GOSUB 800: LPRINT
385 REM NADBYTECNY RADEK
390 P=0: FOR I=1 TO PK: IF VR(I)>0 THEN 450
400 FOR J=1 TO PK: IF VR(J)>0 OR I=J THEN 440
410 FOR K=1 TO 2*VO: IF FH(0,K)=1 THEN 430
420 IF FH(I,K)>FH(J,K) THEN 440
430 NEXT K: VR(I)=1: P=1: GOTO 450
440 NEXTJ
445 REM NADBYTECNY SLOUPEC
450 NEXT I: FOR I=1 TO 2*VO: IF FH(0,I)=1 THEN 510
460 FOR J=1 TO 2*VO: IF FH(0,J)=1 OR I=J THEN 500
470 FOR K=1 TO PK: IF VR(K)>0 THEN 490
480 IF FH(K,I)<FH(K,J) THEN 500
490 NEXT K: FH(0,I)=1: P=1: GOTO 510
500 NEXT J
505 REM NEZBYTNY RADEK
510 NEXT I: FOR I=1 TO 2*VO: IF FH(0,I)=1 THEN 570
520 X=0: FOR J=1 TO PK: IF VR(J)>0 THEN 540
530 IF FH(J,I)=1 THEN X=X+1: K=J
540 NEXT J: IF X<>1 THEN 570
550 VR(K)=2: FOR J=1 TO 2*VO: IF FH(K,J)=1 THEN FH(0,J)=1
560 NEXT J: P=1
570 NEXT I
580 K=0: FOR I=1 TO 2*VO: IF FH(0,I)=0 THEN K=K+1
590 NEXT I: IF K<>0 AND P=1 THEN 390
600 LPRINT "NEZBYTNE RADKY PRI PREDBEZNE REDUKCI:"
610 FOR I=1 TO VP: LPRINT VP$(I);: NEXT I: LPRINT
620 FOR I=1 TO PK: IF VR(I)<>2 THEN 640
630 LPRINT VK$(I)
640 NEXT I: LPRINT
645 IF K=0 THEN LPRINT"MINIMALIZACE PROVEDENA": GOTO 675
650 LPRINT "ZBYVA RESIT TUTO CYKLICKOU MATICI:": GOSUB 800
655 INPUT "DO TESTU PRIDAME KROK";K
660 FOR I=1 TO 2*VO: FH(0,I)=FH(0,I)+FH(K,I): NEXT I
665 VR(K)=2: K=0: FOR I=1 TO 2*VO: IF FH(0,I)=0 THEN K=K+1
670 IF K<>0 THEN 655
675 LPRINT "TEST OBSAHUJE TYTO KROKY:"
680 FOR I=1 TO PK: IF VR(I)<>2 THEN 690
685 LPRINT VK$(I)
690 NEXT I: LPRINT
```

```
692 REM SESTAVENI SLOVNIKU PORUCH
694 DIM SL$(2*VO),SL(2*VO)
696 FOR I=1 TO 2*VO
698 FOR J=1 TO PK: IF VR(J)<>2  THEN 702
700 SL$(I)=SL$(I)+STR$(F1(J,I))
702 NEXT J: SL(I)=I: NEXT I
704 FOR I=2*VO TO 2 STEP -1
706 M$=SL$(I): K=I
708 FOR J=1 TO I-1
710 IF SL$(J)>M$ THEN K=J: M$=SL$(J)
715 NEXT J
720 X$=SL$(I): SL$(I)=SL$(K):SL$(K)=X$
721 X=SL(I): SL(I)=SL(K):SL(K)=X
725 NEXT I
727 LPRINT "SLOVNIK PORUCH"
730 LPRINT "**************":LPRINT
732 LPRINT "KROK      PORUCHY"
734 X=1: FOR I=1 TO PK: IF VR(I)<>2 THEN 738
736 LPRINT STR$(X);: X=X+1
738 NEXT I:LPRINT:LPRINT "------------------"
740 M$="XXX": FOR I=1 TO 2*VO
741 IF M$<>SL$(I) THENLPRINT:LPRINT SL$(I)" ";:M$=SL$(I)
742 A=INT((SL(I)+1)/2):LPRINT VO$(A)"/"STR$(SL(I)+1-2*A)" ";
744 NEXT I:LPRINT
799 END
800 FOR I=1 TO VP: LPRINT VP$(I);: NEXT I: LPRINT " "VY$" ";
810 FOR I=1 TO 2*VO: IF FH(0,I)=1 THEN 830
820 LPRINT VO$(INT((I+1)/2));
830 NEXT I: LPRINT
840 LPRINT TAB(VP+3);: FOR I=1 TO 2*VO: IF FH(0,I)=1 THEN 860
850 LPRINT STR$((I+1)AND 1);
860 NEXT I: LPRINT
870 FOR I=1 TO VP+3+2*VO-R: LPRINT "-";: NEXT I: LPRINT
880 FOR I=1 TO PK: IF VR(I)<>0 THEN 915
885 LPRINT VK$(I)" "STR$(FH(I,0))" ";
890 FOR J=1 TO 2*VO: IF FH(0,J)=1 THEN 910
900 LPRINT STR$(FH(I,J));:
910 NEXT J: LPRINT
915 NEXT I: RETURN
1000 DATA 3,A,B,C,F,8,A,B,C,D,E,F,G,H,8
1010 DATA 000,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0
1020 DATA 001,0,0,0,0,1,0,0,0,1,0,1,0,1,0,0,0,1
1030 DATA 010,0,0,1,0,0,0,1,0,1,0,1,0,1,0,0,0,0
1040 DATA 011,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,0,1
1050 DATA 100,0,0,0,0,1,0,0,0,1,0,1,0,1,0,1,0,0
1060 DATA 101,0,0,0,0,1,0,0,0,1,0,1,0,1,0,1,0,1
1070 DATA 110,1,0,1,0,1,1,1,0,1,1,1,0,1,0,1,1,1
1080 DATA 111,1,1,1,0,1,1,1,1,1,1,1,0,1,1,1,1,1
```

V uvedeném programu nejsou prováděny žádné kontroly správnosti vstupních dat. Z tohoto důvodu je nutné věnovat zadávání vstupních dat zvýšenou pozornost. Data jsou zadávána deklarací DATA na konci programu (od příkazu s číslem 1000). Data jsou zde uvedena v tomto pořadí:

- počet vstupních proměnných;
- označení jednotlivých vstupních proměnných;
- označení výstupní proměnné;
- počet vodičů;



- označení jednotlivých vodičů;
- počet kroků testu;
- jednotlivé řádky tabulky funkčních hodnot, tj. hodnoty vstupních proměnných, hodnota na výstupu v bezporuchovém stavu a funkční hodnoty na výstupu při jednotlivých poruchách v pořadí t0 na prvním vodiči, t1 na prvním vodiči, t0 na druhém vodiči atd. Označení proměnných a vodičů musí být jednoznakové.

Následuje ukázka výstupu tohoto programu. Jsou zde uvedeny stejné informace jako v tab. 10, tab. 12 a tab. 13. Nerozlišitelné poruchy jsou zde označeny poruchou, která je v tabulce uvedena nejdříve. Výstup programu je ukončen výpočty, které se týkají minimalizace testu. Toto využijeme v následujícím cvičení.

```
OPIS TABULKY FUNKCNICH HODNOT
*****************************

ABC F AABBCCDDEEFFGGHH
      0101010101010101
-------------------------
000 0 0000000101010000
001 0 0001000101010001
010 0 0100010101010000
011 1 1101011101011101
100 0 0001000101010100
101 0 0001000101010101
110 1 0101110111010111
111 1 1101111111011111

NEDETEKTOVATELNE PORUCHY:
ZADNE

SEZNAM NEROZLISITELNYCH PORUCH:
A/0 D/0 G/0
A/1 C/1
B/0 F/0
C/0 E/0 H/0
D/1 E/1 F/1

ZKRACENA TABULKA FUNKCNICH HODNOT
*********************************

ABC F AABBCDGH
      01010111
----------------
000 0 00000100
001 0 00010101
010 0 01000100
011 1 11010111
100 0 00010110
101 0 00010111
110 1 01011111
111 1 11011111
```

```
TABULKA PORUCH
**************

ABC F AABBCDGH
      01010111
----------------
000 0 00000100
001 0 00010101
010 0 01000100
011 1 00101000
100 0 00010110
101 0 00010111
110 1 10100000
111 1 00100000

NEZBYTNE RADKY PRI PREDBEZNE REDUKCI:
ABC
010
011
101
110

MINIMALIZACE PROVEDENA
TEST OBSAHUJE TYTO KROKY:
010
011
101
110

SLOVNIK PORUCH
**************

KROK      PORUCHY
1234
--------------------

0000 F/0 B/0
0001 H/0 E/0 C/0
0100 G/0 D/0 A/0
0111 H/1 G/1 B/1
1101 C/1 A/1
1111 F/1 E/1 D/1
```

## 8.3. MINIMALIZACE TESTŮ

### Zadání

Minimalizujte test, pro který je tabulkou poruch tab. 13.

### Úvod

Minimalizace spočívá v tom, že v této tabulce hledáme nejmenší podmnožinu řádků, která má v každém sloupci alespoň jednu jedničku. Tak minimalizaci testu převedeme na řešení problému, zda daný test pokrývá všechny poruchy. Tento problém je jednou ze základních úloh, s níž se setkáme nejen v diagnostice, ale také při návrhu logických obvodů.

Máme-li najít libovolné pokrytí tabulky poruch, můžeme použít algoritmus založený na postupném prohledávání tabulky. Podstata tohoto algoritmu spočívá v tom, že libovolně vybereme jeden sloupec tabulky a v něm prvek $t_{ij} = 1$. Tento prvek určuje řádek tabulky, tj. krok testu. Můžeme tedy z tabulky vyřadit všechny sloupce pokryté tímto řádkem, tj. sloupce, kde prvek $f_{ik} = 1$ (pro $k = 1$ atd.). Ze zbylých sloupců opět volíme jeden, který pokryjeme stejným způsobem. Takto pokračujeme, dokud není pokryta celá množina sloupců. Jestliže tento postup opakujeme, až vyčerpáme všechny varianty, získáme množinu řešení, ze které můžeme vybrat řešení minimální. Popsaný postup je ale nesmírně zdlouhavý.

### Postup cvičení

Významného zjednodušení algoritmu řešení dosáhneme, rozdělíme-li postup na předběžnou redukci a řešení cyklické tabulky (matice).

*Předběžná redukce tabulky* se provádí podle těchto pravidel, která můžeme používat v libovolném pořadí:

1. vybrat řádek, který má v některém sloupci pouze jednu jedničku (tzv. *nezbytný řádek*), a vyškrtnout všechny sloupce, v nichž má jedničku;
2. vyškrtnout sloupec, který pokrývá jiný sloupec;
3. vyškrtnout řádek, který je pokryt jiným řádkem (toto zjednodušení je přípustné, nevyžadujeme-li nalezení všech minimálních pokrytí tabulky).

O řádku (sloupci) $A$ říkáme, že *pokrývá* řádek (sloupec) $B$, jestliže $A$ obsahuje jedničky ve všech sloupcích (řádcích), v nichž obsahuje jedničky



i $B$. $A$ tedy může obsahovat některé jedničky navíc nebo může být s $B$ shodný. Shodu dvou řádků (sloupců) je možné považovat za zvláštní případ pokrytí, při němž si můžeme vybrat, který ze shodných řádků (sloupců) vyškrtneme. Při vyškrtávání sloupců lze postupovat zcela libovolně, zatímco vyškrtáváním řádků se připravujeme o některá minimální řešení.

Předběžnou redukci provádíme tak dlouho, dokud alespoň jedno z uvedených pravidel lze použít. Jako *cyklickou* označujeme takovou tabulku poruch, pro kterou není možné použít žádné ze zjednodušujících pravidel. Minimální pokrytí lze v takovéto tabulce najít pouze vyzkoušením všech možností. Při přibližné minimalizaci ale stačí najít jedno vyhovující řešení, což je podstatně jednodušší. V některých případech vystačíme s použitím zjednodušujících pravidel a není nutné řešit pokrytí cyklické tabulky.

Vraťme se ale k našemu příkladu, tj. k tabulce 13. Nejdříve vybereme nezbytné řádky. Zde tedy musíme najít sloupce, ve kterých je pouze jedna jednička. V našem případě to jsou sloupce pro poruchy $A/1$, $D/0$ a $E/0$ (tyto sloupce jsou v tab. 13 označeny křížky). Nezbytné řádky jsou ty řádky, ve kterých jsou v označených sloupcích jedničky. Jde o řádky 2, 3 a 6 (v tabulce jsou označeny tečkou). Tyto řádky budou obsaženy v minimálním řešení. Řádek 3 má jedničky také ve sloupci poruchy $F/0$. Tento sloupec můžeme vyškrtnout. V řádku 2 toto platí pro poruchu $F/1$, vyškrtáváme tedy sloupec poruchy $F/1$. V tabulce 13 jsou tyto sloupce označeny čtverečkem. Po těchto úpravách získáme tabulku 14, kde jsou vynechány i sloupce označené křížkem a nezbytné řádky. V této tabulce řádek 5 obsahuje samé jedničky, a pokrývá tedy všechny ostatní řádky (můžeme je proto vyškrtnout). Zůstal nám tak pouze řádek 5, který přidáme k našemu řešení. Řešení je tedy tvořeno řádky 2, 3, 5 a 6. Pokud se chceme přesvědčit, zda jsou všechny poruchy tímto minimálním testem detektovány, je možné napsat tabulku poruch pro kroky testu 2, 3, 5 a 6 a zkontrolovat, zda v této tabulce je v každém sloupci alespoň jedna jednička. Je to provedeno v tabulce 15. V tomto případě jsme vystačili s použitím zjednodušujících pravidel, nebylo třeba řešit pokrytí cyklické tabulky.

Můžeme také provést kontrolu, zda jsme dostali stejný test jako při cvičení, kde jsme test vytvářeli použitím metody intuitivního zcitlivění cesty (viz. čl. 8.1). V tabulce 9 vidíme, že jsme vytvořili stejný test, pouze jednotlivé kroky testu jsou zde v jiném pořadí.

Pro kontrolu je také možné použít další část výstupu z programu, se kterým jsme se seznámili v předchozím cvičení. Program provede předběžnou

redukci a vypíše cyklickou tabulku (matici). Z cyklické tabulky musíte sami vybrat kroky testu, které doplňujete do minimálního testu. To provedete zadáním čísla kroku (kroky jsou číslovány od jedničky). Pokud přidáváním kroků získáte úplný test, jsou vypsány všechny kroky tohoto testu. V našem případě získáme úplný test již při redukci a cyklická tabulka vypsána není. Poslední část výstupu (slovník poruch) použijeme v dalším cvičení. Pro procvičení můžete minimalizovat další testy, pro které byly v předchozím cvičení vytvořeny tabulky poruch.

*Tabulka 14.* Tabulka poruch s vynechanými nezbytnými řádky

| K | B | G | H | |
|---|---|---|---|---|
| | 1 | 1 | 1 | |
| 0 | 0 | 0 | 0 | ■ |
| 1 | 1 | 0 | 1 | ■ |
| 4 | 1 | 1 | 0 | ■ |
| 5 | 1 | 1 | 1 | |
| 7 | 0 | 0 | 0 | ■ |

*Tabulka 15.* Tabulka poruch pro vybrané kroky testu

| K | A | B | D | E | F | F | G | H |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |



## 8.4. SLOVNÍK PORUCH

### Zadání

Vytvořte koincidenční slovník poruch pro obvod z obr. 49.

### Úvod

Koincidenční slovník je nejjednodušší formou slovníku poruch. Tento slovník je pouze utříděným seznamem odezev na celý test pro všechny poruchové stavy.

### Postup cvičení

Všechny potřebné informace pro vytvoření slovníku poruch získáme např. v tabulce funkčních hodnot, sestavované při vytváření tabulky poruch. Z této tabulky vybereme pouze řádky, které odpovídají krokům minimálního testu. Tím vznikne redukovaná tabulka funkčních hodnot, v našem případě je takovou tabulkou tab. 16. Slovník poruch uvedený v tabulce 17 odvodíme z této tabulky pouze záměnou sloupců za řádky a utříděním řádků vzestupně (v zájmu snadné orientace). V každém řádku slovníku jsou uvedeny ekvivalentní poruchy. V tabulce 17 vidíme, že v důsledku použití minimálního diagnostického testu splynuly kromě tříd ekvivalence i poruchy $B/1$, $G/1$, $H/1$, které jsou jinak rozlišitelné. Je možné je rozlišit prodloužením testu o kroky 001 a 100. Slovník poruch se pro vyhodnocení výsledku testu používá takto: Podle hodnot výstupu obvodu v jednotlivých krocích testu najdeme řádek ve slovníku. V tomto řádku jsou uvedeny poruchy, které v testovaném obvodu mohly nastat.

*Tabulka 16.* Redukovaná tabulka funkčních hodnot

| A | B | C | A | A | B | B | C | C | D | D | E | E | F | F | G | G | H | H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

Pro kontrolu správnosti vaší práce je možné použít poslední část výstupu z programu, se kterým jsme se seznámili při vytváření tabulky poruch. Dostali jsme stejný výsledek, pouze poruchy v jednotlivých řádcích jsou uvedeny v jiném pořadí.

*Tabulka 17.* Slovník poruch

| Krok | | | | Poruchy |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| 0 | 0 | 0 | 0 | *F*/0, *B*/0 |
| 0 | 0 | 0 | 1 | *C*/0, *E*/0, *H*/0 |
| 0 | 1 | 0 | 0 | *A*/0, *D*/0, *G*/0 |
| 0 | 1 | 1 | 1 | *B*/1, *G*/1, *H*/1 |
| 1 | 1 | 0 | 1 | *A*/1, *C*/1 |
| 1 | 1 | 1 | 1 | *D*/1, *E*/1, *F*/1 |

Nevýhodou koincidenčního slovníku poruch je to, že velikost paměti potřebná pro jeho uložení roste s velikostí testovaného obvodu tak rychle, že tento typ slovníku v praxi nelze téměř použít. Může ale sloužit jako východisko pro vytváření slovníků poruch jiných typů. Dalším problémem je vyhodnocení nezařaditelných odezev, které mohou vzniknout např. výskytem většího množství poruch nebo nestálými poruchami. Z těchto důvodů se většinou používají slovníky s nižším rozlišením, např. slovníky se záznamem odezvy v prvním chybném kroku. Tímto typem slovníku se zde již ale nebudeme zabývat.

Pro procvičení vytváření slovníku poruch vytvořte slovníky pro další obvody, pro které jste minimalizovali testy v předchozím cvičení.

## 8.5. TEXTOVÝ EDITOR

### Zadání

Seznamte se s činností a s obsluhou textového editoru.

### Úvod

Textový editor je jedním z programů operačního systému a používá se k vytváření a opravování textových souborů. Textový editor je tedy možné použít i pro zápis a opravy zdrojových programů. Při vytváření nového



programu (po vyvolání textového editoru) se zapisuje nový text z klávesnice. Tento text (program) je uložen v paměti počítače a můžeme jej libovolně měnit. Je také čten překladačem při překladu.

Textový editor může být orientován na opravy celých řádků textu nebo na opravu jednotlivých znaků. Novější verze operačního systému jsou doplněny *obrazovkově orientovaným textovým editorem.* Tento editor využívá možnosti měnit, vkládat a vynechávat znaky v zobrazovaném textu. Úpravy textu pak probíhají podstatně rychleji než ve starších, *řádkově orientovaných verzích textových editorů.*

### Postup cvičení

Dále je vypsán program, který můžete použít pro simulaci mikroprocesoru 8080. Tento program je zapsán v jazyku BASIC pro počítač IQ 151 vybavený modulem VIDEO 64 (na řádku obrazovky je zobrazováno 64 znaků). Program je možné používat i s modulem VIDEO 32, ale texty zobrazované na obrazovce jsou nepřehledné. Tento program obsahuje také jednoduchou verzi textového editoru (příkazy 100 až 360 a 8000 až 8150). Editor zde vyvoláme příkazem EDIT a dále můžeme volit následující funkce:

NEW – zápis nového zdrojového programu;
LIST – výpis programu na obrazovce;
LLIST – výpis programu na tiskárně;
REPL – nahrazení řádku programu jiným řádkem;
INS – vložení nového řádku do textu programu;
DEL – zrušení řádku v textu programu;
END – ukončení činnosti editoru.

Při zápisu řádku programu (funkce: NEW, REPL a INS) je nutné na začátku a konci řádku zapisovat uvozovky. Toto vyžaduje interpretační překladač jazyka BASIC při vstupu textové konstanty příkazem INPUT, pokud tato konstanta obsahuje čárku nebo dvojtečku a pokud začíná mezerou.

Seznamte se s ovládáním tohoto textového editoru. Pomocí příkazu NEW zapište do paměti libovolný program v jazyce symbolických adres mikroprocesoru 8080 a dále v tomto programu provádějte změny. Příkazem LIST nebo LLIST si vždy ověřte, zda každá změna byla provedena tak, jak jste požadovali.

Pokud máte k dispozici dokonalejší textový editor, naučte se s ním pracovat.

```
10 REM PROGRAM PRO SIMULACI MHB 8080
15 CLEAR 2000:J=-1:W4=0
25 DIM S$(100),W(1000):N=100:M=1000
50 PRINT: INPUT "CINNOST - EDIT/TRANS/SIM";G$
60 I=0: IF G$="EDIT" THEN 100
70 IF G$="TRANS" THEN 400
80 IF G$="SIM" THEN 800
90 PRINT "NEROZUMIM": GOTO 50
100 PRINT: PRINT "FUNKCE EDITORU - NEW/LIST/LLIST/REPL"
105 INPUT "                    INS/DEL/END";G$: I=0
110 IF G$="NEW" THEN 250
115 IF G$="LIST" THEN 170
118 IF G$="LLIST" THEN 350
120 IF G$="REPL" THEN 315
125 IF G$="INS" THEN 275
130 IF G$="DEL" THEN 200
135 IF G$="END" THEN PRINT "KONEC EDITORU": GOTO 50
140 PRINT "NEROZUMIM": GOTO 100
160 PRINT "NENI ZDROJOVY PROGRAM": GOTO 100
170 IF J<0 THEN 160
175 G$="VYPIS OD RADKU ": GOSUB 8000
178 Z0=0
180 GOSUB 8085: U=U+1
182 IF U>J THEN PRINT "KONEC PROGRAMU": GOTO 100
185 Z0=Z0+1: IF Z0<10 THEN 180
190 INPUT "POKRACOVAT A/N";G$: IF G$="Y" THEN 178
195 PRINT "VYPIS UKONCEN": GOTO 100
200 IF J<0 THEN 160
205 G$="VYNECHAVAT OD RADKU ": GOSUB 8000
210 INPUT "VYNECHAVAT DO RADKU CISLA";K
215 IF K<U THEN PRINT "CHYBNE CISLO": GOTO210
220 Z1=U: PRINT "VYNECHANY RADKY :"
225 PRINT Z1;TAB(5);:GOSUB 8086: IF U=J THEN 235
230 FOR Z0=U TO J-1: S$(Z0)=S$(Z0+1): NEXT Z0
235 J=J-1: Z1=Z1+1: IF Z1>K THEN 100
240 IF J>=U THEN 225
245 GOTO 100
250 IF J>=0 THEN J=-1: PRINT "ZDROJOVY PROGRAM ZRUSEN"
255 PRINT "VSTUP NOVEHO ZDROJOVEHO PROGRAMU (UKONCIT ZNAKEM @):"
260 IF J+1>100 THEN PRINT "DLOUHY PROGRAM": GOTO 100
265 U=J+1: GOSUB 8120: IF Q<>0 THEN 100
270 J=J+1: S$(J)=G$: GOTO 260
275 IF J<0 THEN 160
280 IF J+1>100 THEN 260
285 G$="VKLADAT ZA RADEK ": GOSUB 8000
290 PRINT "VSTUP PRIKAZU (UKONCIT ZNAKEM @):"
295 U=U+1:IF J+1>100 THEN 260
300 GOSUB 8120: IF Q<>0 THEN 100
305 IF U<=J THEN FOR Z0=J TO U STEP -1: S$(Z0+1)=S$(Z0): NEXT Z0
310 S$(U)=G$: J=J+1: GOTO 295
315 IF J<0 THEN 160
320 G$="NAHRAZOVAT PRIKAZY OD RADKU ": GOSUB 8000
325 PRINT "VSTUP PRIKAZU (UKONCIT ZNAKEM @, PRI * SE PRIKAZ NEMENI)"
330 I=-1: GOSUB 8085: GOSUB 8120: IF Q<>0 THEN 100
335 IF G$<>"*" THEN PRINT "PRIKAZ"U"NAHRAZEN":S$(U)=G$
340 U=U+1: IF U>J THEN 182
345 GOTO 330
350 U=0:IF J<0 THEN 160
356 GOSUB 8130: U=U+1: IF U>J THEN PRINT "KONEC VYPISU": GOTO 100
360 GOTO 356
400 IF J<0 THEN PRINT "NENI ZDROJOVY PROGRAM":GOTO 50
405 W4=0:F=4:INPUT"ZDROJOVY PROGRAM PREKLADAT OD ADRESY";H$
410 GOSUB 8180: IF Q<>0 THEN 405
```



```
415 W7=D
420 INPUT "PROMENNE UMISTIT OD ADRESY";H$
425 GOSUB 8180: IF Q<>0 THEN 420
430 IF D=W7 THEN PRINT "NENI MISTO PRO PROGRAM": GOTO 420
435 IF D<=255 THEN PRINT "DOLNI MEZ ADRESY JE 0100":GOTO 420
440 W8=D:W5=0:W6=0:U=0:W9=0:WD=0
445 IFW4+W5+W6+WD+6>M1THEN PRINT "V PAMETI NENI MISTO":GOTO 50
450 IF W7+W4+2<W8 THEN 460
455 IF W8+WD+1=W7 THEN PRINT "V PAMETI NENI MISTO": GOTO 50
460 IF W7+W4+2>65535 THEN PRINT "VELKA ADRESA": GOTO 50
465 IF W8+WD+1>65535 THEN PRINT "VELKA ADRESA": GOTO 50
470 GOSUB 8085: G$=S$(U): L=LEN(G$): GOSUB 8530: IF Q=0 THEN 485
475 PRINT "OPRAV PRIKAZ (@ KONCI PREKLAD)":GOSUB 8120:IF Q<>0 THEN 50
480 S$(U)=G$:GOTO 470
485 S$(U)=G$:WV=-1:WT=-1:IF L$=" " THEN 545
490 G$=":"+L$:GOSUB 8620:IF Q=0 THEN 510
500 IF W5+W6+1>N-J THEN PRINT "V PAMETI NENI MISTO":GOTO 50
505 S$(N-W5-W6)=G$:W(M-W5-W6)=W7+W4:W5=W5+1:GOTO 545
510 IF W(M-Z0)>=0 THEN PRINT "OPAKOVANA DEKLARACE": GOTO 475
515 Z1=ABS(W(M-Z0)):W(M-Z0)=W7+W4:W9=W9+1:GOTO 532
520 Z1=ABS(W(M-Z0)):IF Z0=W5+W6-1 THEN 530
525 FORQ=0TOW5+W6-2-Z0:X=Z0+Q:S$(N-X)=S$(N-X-1):W(M-X)=W(M-X-1):NEXTQ
530 W5=W5-1:W9=W9-1:A=A-1
532 D=W7+W4-256*INT((W7-W4)/256):W(Z1-W7)=D
535 D=INT((W7+W4)/256):W(Z1+1-W7)=D:Z0=A:GOSUB8630:IF Q=0 THEN 520
545 IF M$=" " THEN 670
550 IF P1=777 THEN 640
555 IF C=7 THEN 610
560 G$=M$: GOSUB 8620: IF Q=0 THEN 585
565 IF W5+W6+1>N-J THEN 500
570 S$(N-W5-W6)=G$: W(M-W5-W6)=W8+WD: WT=-3: WV=W8+WD
575 W6=W6+1:WD=WD+1:GOSUB 8640: IF Q=0 THEN 670
580 W(M-W5-W6+1)=-WV: WV=-WV: WD=WD+1: GOTO 670
585 WT=-3: GOSUB 8640: IF Q=0 THEN 600
590 IF W(M-Z0)<0 THEN 605
595 PRINT "PROMENNA "M$" NENI 16-TI BITOVA":GOTO 475
600 IFW(M-Z0)<0THENPRINT"PROMENNA "M$" NENI 8-BITOVA":GOTO475
605 WV=W(M-Z0):GOTO 670
610 G$=":"+M$:GOSUB 8620: IF Q=0 THEN 630
615 IF W5+W6+1>N-J THEN 500
620 S$(N-W5-W6)=G$:W(M-W5-W6)=-(W7+W4+1):WT=-4
625 WV=-(W7+W4+1): W5=W5+1: W9=W9+1: GOTO 670
630 IF W(M-Z0)<0 THEN 615
635 WT=INT(W(M-Z0)/256):WV=W(M-Z0)-256*WT:GOTO 670
640 F$=MID$(M$,F+1,1):H$=MID$(M$,1,F):IF F$="D" THEN 650
645 GOSUB 8180: GOTO 655
650 GOSUB 8660
655 WV=D-256*INT(D/256): IF F=2 THEN 670
660 WT=INT(D/256)
670 W(W4)=P:W4=W4+1: IF WT=-4 THEN 710
675 IF WT=-3 THEN 700
680 IF WT<>-1 THEN 695
685 IF WV=-1 THEN 720
690 W(W4)=WV:W4=W4+1:GOTO 720
695 W(W4)=WV: W(W4+1)=WT:W4=W4+2: GOTO720
700 IF WV<0 THEN W(W4)=WV: W4=W4+1
705 W(W4)=WV: W4=W4+1: GOTO 720
710 W(W4)=WV: W(W4+1)=WV: W4=W4+2
720 U=U+1:IF U<=J THEN 445
730 F=4:D=W7:GOSUB8160:PRINT "STARTOVNI ADRESA "H$
735 PRINT"PROGRAM JE DLOUHY"W4"BYTU":IF W9=0 THEN 745
740 PRINT W9"NEDEKLAROVANYCH PROMENNYCH":GOTO50
745 D=W7:U=D-W7
```

```
750 F=4:D=U+W7:GOSUB 8160: PRINT H$;TAB(8);
755 F=2:D=W(U):GOSUB 8160:PRINT H$:U=U+1:IF U<W4 THEN 750
795 PRINT"KONEC PREKLADU":GOTO50
800 IFW4=0THEN PRINT"NENI PRELOZENY PROGRAM":GOTO50
810 IF W9<>0 THEN PRINT"NELZE POUZIT":GOTO740
815 W2=0: PRINT"INICIALISACE PRACOVNICH OBLASTI:":GOSUB 8700
820 GOSUB 8720: GOSUB 8750: GOSUB 8770
825 W2=0
826 PRINT:PRINT"FUNKCE SIMULATORU - CLEAR/START/CONT/BREAK"
830 INPUT "                    SET/DISPLAY/TRACE/END";G$
835 IF G$="END" THEN PRINT "KONEC SIMULACE": GOTO 50
840 IF G$="CLEAR" THEN 900
845 IF G$="BREAK" THEN 1010
850 IF G$="TRACE" THEN 1040
855 IF G$="DISPLAY" THEN 1060
860 IF G$="SET" THEN 1200
865 IF G$="START" THEN 1500
870 IF G$="CONT" THEN 1450
875 PRINT"NEROZUMIM":GOTO825
900 GOSUB 8700: GOSUB 8720: GOSUB 8750: GOSUB 8770
905 INPUT "DEFINOVANI 1. PRACOVNI OBLASTI: Y/N";G$
910 IF G$="N" THEN 955
915 F=4:INPUT "OD ADRESY";H$: GOSUB 8180:IF Q<>0 THEN 915
920 GOSUB 8800: IF Q=0 THEN PRINT "CHYBA": GOSUB 8770: GOTO 915
925 W0=D
926 INPUT "DO";H$:GOSUB 8180: IF Q<>0 THEN 926
930 IF D<W0 THEN PRINT "CHYBA":GOTO 926
935 GOSUB 8840: IF Q=0 THEN 950
940 IF Q<>2 THEN PRINT"CHYBA"
945 GOSUB 8770:GOTO905
950 WV=W0:WC=W1-W0+1:GOSUB8770
955 INPUT "DEFINOVANI 2. PRACOVNI OBLASTI: Y/N";G$
960 IF G$="N" THEN 825
965 F=4: INPUT "OD ADRESY";H$:GOSUB 8180: IF Q<>0 THEN 965
970 GOSUB 8800: IF Q=0 THEN PRINT "CHYBA": GOSUB 8770: GOTO 965
975 W0=D
976 INPUT "DO";H$:GOSUB 8180: IF Q<>0 THEN 976
980 IF D<W0 THEN PRINT "CHYBA":GOTO 976
985 GOSUB 8840: IF Q=0 THEN 1000
990 IF Q<>2 THEN PRINT"CHYBA"
995 GOSUB 9770:GOTO955
1000 WT=W0:WP=W1-W0+1:GOSUB 8770:GOTO825
1010 WJ=0:PRINT"VSTUP ADRES ZASTAVENI - UKONCIT @"
1015 IF WJ>31 THEN PRINT "MNOHO ADRES":GOTO 825
1020 F=4:INPUT"ZADEJ ADRESU";H$:IF H$="@" THEN 825
1025 GOSUB 8180: IF Q<>0 THEN 1020
1030 W(M-W5-W6-WD-WJ)=D:WJ=WJ+1:GOTO 1015
1040 W2=1: GOTO 826
1060 PRINT "A F  B C  D E  H L  SP   PC":GOSUB1065:GOTO1090
1065 F=4:D=A8*256+F8:GOSUB 8160:PRINT H$" ";:D=B8*256+C8
1070 GOSUB 8160:PRINT H$" ";:D=D8*256+E8:GOSUB 8160
1080 PRINT H$" ";:D=H8*256+L8:GOSUB 8160:PRINT H$" ";:D=SA+256*SB
1085 GOSUB8160:PRINTH$" ";:D=PA+256*PB:GOSUB8160:PRINTH$:RETURN
1090 INPUT "VYPISOVAT OBSAH PAMETI - Y/N";G$:IF G$="N" THEN 1135
1095 F=4:INPUT"OD ADRESY";H$:GOSUB 8180:IFQ<>0THEN 1095
1100 GOSUB 8800:IF Q=0 THEN 1110
1105 PRINT"ADRESA NELEZI V PRACOVNI OBLASTI":GOSUB8770:GOTO 1095
1110 U=D
1111 Z1=0:F=4:D=U:GOSUB8160:PRINT H$" ";:F=2
1115 D=W(H):GOSUB 8160:PRINT H$" ";:U=U+1:D=U
1120 GOSUB 8800:IFQ<>0 THEN 1135
1125 Z1=Z1+1:IF Z1<8 THEN 1115
1130 PRINT:GOTO 1111
```



```
1135 IF W6=0 THEN 825
1140 INPUT"VYPISOVAT OBSAH PROMENNYCH - Y/N";G$:IFG$="N"THEN 825
1145 U=0
1150 F=2:G$=S$(N-U):F$=MID$(G$,1,1):IF F$=":" THEN 1165
1155 D=M-W5-W6-ABS(W(M-U))+W8:H=W(D):IFW(M-U)<0THEN F=4:H=H+256*W(D-1)
1160 D=H:GOSUB 8160:PRINT G$;TAB(12);H$
1165 U=U+1:IF U>=W5+W6 THEN 825
1170 GOTO 1150
1200 INPUT "ZMENA OBSAHU REGISTRU - Y/N";G$:IF G$="N"THEN 1305
1205 PRINT"VSTUP NOVEHO OBSAHU REGISTRU,@-KONCI,*-NEMENI SE"
1210 F=2:G$="A ":Z1=A8:GOSUB 8870:IF H$="@"THEN 1305
1220 IF H$<>"*"THEN A8=D
1225 G$="B ":Z1=B8:GOSUB 8870:IF H$="@"THEN 1305
1230 IF H$<>"*"THEN B8=D
1235 G$="C ":Z1=C8:GOSUB 8870:IF H$="@"THEN 1305
1240 IF H$<>"*"THEN C8=D
1245 G$="D ":Z1=D8:GOSUB 8870:IF H$="@"THEN 1305
1250 IF H$<>"*"THEN D8=D
1255 G$="E ":Z1=E8:GOSUB 8870:IF H$="@"THEN 1305
1260 IF H$<>"*"THEN E8=D
1265 G$="H ":Z1=H8:GOSUB 8870:IF H$="@"THEN 1305
1270 IF H$<>"*"THEN H8=D
1275 G$="L ":Z1=L8:GOSUB 8870:IF H$="@"THEN 1305
1280 IF H$<>"*"THEN L8=D
1285 F=4:G$="SP":Z1=SA+SB*256:GOSUB 8870:IF H$="@"THEN 1305
1290 IF H$<>"*" THEN SB=INT(D/256):SA=D-SB*256
1295 G$="PC":Z1=PA+PB*256:GOSUB 8870:IF H$="@"THEN 1305
1300 IF H$<>"*" THEN PB=INT(D/256):PA=D-PB*256
1305 INPUT"ZMENA OBSAHU PAMETI - Y/N";G$:IF G$="N" THEN 1360
1310 F=4:INPUT "OD ADRESY";H$:GOSUB 8180:IF Q<>0 THEN 1310
1315 GOSUB 8800: IF Q=0 THEN 1325
1320 PRINT"ADRESA NELEZI V PRACOVNI OBLASTI":GOSUB8770:GOTO 1310
1325 PRINT"VSTUP NOVE HODNOTY,@-KONCI,*-NEMENI SE":U=D
1330 F=4:D=U:GOSUB8160:PRINT H$" ";:G$=H$
1335 F=2:D=W(H):GOSUB 8160:PRINT H$;:INPUT H$
1340 IF H$="@" THEN 1305
1345 IF H$="*" THEN 1356
1350 GOSUB 8180:IF Q<>0 THEN 1330
1355 W(H)=D
1356 U=U+1:D=U:GOSUB8800:IF Q<>0 THEN 1330
1360 IF W6=0 THEN 825
1365 INPUT"MENIT OBSAH PROMENNYCH - Y/N";G$:IF G$="N" THEN 825
1370 PRINT"VSTUP NOVE HODNOTY,@-KONCI,*-NEMENI SE":U=0
1375 F=2:G$=S$(N-U):F$=MID$(G$,1,1):IFF$=":"THEN 1420
1380 D=M-W5-W6-ABS(W(M-U))+W8:H=W(D)
1385 IF W(M-U)<0 THEN F=4:H=H+256*W(D-1)
1390 D=H:GOSUB 8160:PRINT G$;TAB(12);H$;:INPUT H$
1395 IF H$="*" THEN 1420
1400 IF H$="@" THEN 825
1405 GOSUB 8180:IF Q<>0 THEN 1390
1410 H=M-W5-W6-ABS(W(M-U))+W8:IF F=2 THEN W(H)=D:GOTO 1420
1415 W(H-1)=INT(D/256):W(H)=D-256*W(H-1)
1420 U=U+1:IF U>=W5+W6 THEN 825
1425 GOTO 1375
1450 IF WS=0 THEN 1500
1455 F=4:D=PA+PB*256:GOSUB 8160
1460 PRINT"SIMULACE POKRACUJE OD ADRESY ";H$:GOTO 1600
1470 PRINT"CHYBNY OPERACNI ZNAK":GOTO 825
1480 PRINT"ADRESA ZASTAVENI":GOTO825
1490 PRINT"SIMULACE ZASTAVENA":GOTO 825
1500 F=4:INPUT"POCATECNI ADRESA SIMULACE";H$
1505 GOSUB 8180:IF Q<>0THEN 1500
1510 GOSUB 8800:IF Q=0 THEN 1520
```

```
1515 PRINT"ADRESA NELEZI V PRACOVNI OBLASTI"GOSUB 8770:GOTO 825
1520 PB=INT(D/256):PA=D-PB*256:WS=1
1600 F=4:D=PA+PB*256:X3=D:GOSUB 8160:D=X3
1605 IF W2()0 THEN PRINT "PC : "H$;
1610 GOSUB 8800:IF Q=0 THEN 1620
1615 PRINT"ADRESA MIMO PRACOVNI OBLAST"::GOTO 825
1620 D=W(H):IF D(0 OR D)255 THEN 1470
1625 Z=D:IF W2()0 THEN F=2:GOSUB 8160: PRINT"  "H$
1630 Q=0:POKE48,177:POKE49,195:POKE50,129:POKE51,209
1640 CALL 48,4000+Z*10:REM GOTO 4000+Z*10
3498 IF Q()0 THEN 1615
3500 GOSUB 9000
3501 IFW2()0THENGOSUB 1065
3505 IF WJ=0THEN 3520
3510 FOR I=0 TO WJ-1:IF W(M-W5-W6-WH-I)=X3 THEN 1480
3515 NEXT I
3520 X$=INKEY$:IFX$="@" THEN 1490
3525 IFX$=""THEN 1600
3530 PRINT"PRERUSENI"
3535 INPUT"ZADEJ ADRESU PODPROGRAMU PRO ZPRACOVANI PRERUSENI";H$
3540 GOSUB 8180:IF Q()0 THEN 3535
3545 GOSUB 8800:IFQ=0 THEN 1520
3550 PRINT"ADRESA NELEZI V PRACOVNI OBLASTI":GOSUB 8770:GOTO3535
4000 GOTO3500:REM NOP
4010 GOSUB9020:C8=D:GOSUB9020:B8=D:GOTO3498:REM LXI B
4020 D=C8+B8*256:REM STAX B
4021 GOSUB8800:IF Q=1 THEN 1615
4022 W(H)=A8:GOTO3498
4030 Z=B8:Z1=C8:GOSUB 9030:B8=Z:C8=Z1:GOTO3498:REM INX B
4040 D=B8:GOSUB9050:B8=D:GOTO3498:REM INR B
4050 D=B8:GOSUB 9140:B8=D:GOTO3498:REM DCR B
4060 GOSUB9020:B8=D:GOTO3498:REM MVI B
4070 D=A8:GOTO9150:REM RLC
4080 F=2:D=Z:GOSUB8160:PRINTH$"NEDOVOLENY OPERACNI KOD":GOTO825
4090 Z=B8:Z1=C8:GOTO9180:REM DAD B
4100 D=C8+B8*256:GOTO9210:REM LDAX B
4110 Z=B8:Z1=C8:GOSUB9220:B8=Z:C8=Z1:GOTO3498:REM DCX B
4120 D=C8:GOSUB9050:C8=D:GOTO3498:REM INR C
4130 D=C8:GOSUB9140:C8=D:GOTO3498:REM DCR C
4140 GOSUB9020:C8=D:GOTO3498:REM MVI C
4150 D=A8:GOTO9250:REM RRC
4160 GOTO4080
4170 GOSUB9020:E8=D:GOSUB9020:D8=D:GOTO3498:REM LXI D
4180 D=E8+D8*256:GOTO4021:REM STAX D
4190 Z=D8:Z1=E8:GOSUB9030:D8=Z:E8=Z1:GOTO3498:REM INX D
4200 D=D8:GOSUB9050:D8=D:GOTO3498:REM INR D
4210 D=D8:GOSUB9140:D8=D:GOTO3498:REM DCR D
4220 GOSUB9020:D8=D:GOTO3498:REM MVI D
4230 D=A8:GOTO9330:REM RAL
4240 GOTO4080
4250 Z=D8:Z1=E8:GOTO9180:REM DAD D
4260 D=E8+D8*256:GOTO9210:REM LDAX D
4270 Z=D8:Z1=E8:GOSUB9220:D8=Z:E8=Z1:GOTO3498:REM DCX D
4280 D=E8:GOSUB9050:E8=D:GOTO3498:REM INR E
4290 D=E8:GOSUB9140:E8=D:GOTO3498:REM DCR E
4300 GOSUB9020:E8=D:GOTO3498:REM MVI E
4310 D=A8:GOTO9360:REM RAR
4320 GOTO4080
4330 GOSUB9020:L8=D:GOSUB9020:H8=D:GOTO3498:REM LXI H
4340 GOSUB9380:IFQ=1THEN1615:REM SHLD
4342 W(H)=L8:GOSUB9030:D=Z1+Z*256:GOSUB8800:IFQ=1THEN1615
4344 W(H)=H8:GOTO3498
4350 Z=H8:Z1=L8:GOSUB9030:H8=Z:L8=Z1:GOTO3498:REM INX H
```



```
4360 D=H8:GOSUB9050:H8=D:GOTO3498:REM INR H
4370 D=H8:GOSUB9140:H8=D:GOTO3498:REM DCR H
4380 GOSUB9020:H8=D:GOTO3498:REM MVI H
4390 D=A8:GOTO9420:REM DAA
4400 GOTO4080
4410 Z=H8:Z1=L8:GOTO9180:REM DAD H
4420 GOSUB9380:IF Q=1THEN1615:REM LHLD
4422 L8=W(H):GOSUB9030:D=Z1+Z*256:GOSUB8800:IFQ=1THEN1615
4424 H8=W(H):GOTO3498
4430 Z=H8:Z1=L8:GOSUB9220:H8=Z:L8=Z1:GOTO3498:REMDCX H
4440 D=L8:GOSUB9050:L8=D:GOTO3498:REM INR L
4450 D=L8:GOSUB9140:L8=D:GOTO3498:REM DCR L
4460 GOSUB9020:L8=D:GOTO3498:REM MVI L
4470 D=A8:GOTO9500:REM CMA
4480 GOTO4080
4490 GOSUB9020:SA=D:GOSUB9020:SB=D:GOTO3498:REM LXI SP
4500 GOSUB9380:IFQ=1THEN1615:REM STA
4505 W(H)=A8:GOTO3498
4510 Z=SB:Z1=SA:GOSUB9030:SB=Z:SA=Z1:GOTO3498:REM INX SP
4520 D=L8+H8*256:GOTO9520:REM INR M
4530 D=L8+H8*256:GOTO9530:REM DCR M
4540 GOSUB9020:F=D:D=L8+H8*256:GOTO9540:REM MVI M
4550 Z=0:Z1=1:GOSUB9120:GOTO3498:REM STC
4560 GOTO4080
4570 Z=SA:Z1=SB:GOTO9180:REM DAD SP
4580 GOSUB9380:IFQ=1THEN1615:REM LDA
4585 A8=W(H):GOTO3498
4590 Z=SA:Z1=SB:GOSUB9220:SB=Z:SA=Z1:GOTO3498:REM DCX SP
4600 D=A8:GOSUB 9050:A8=D:GOTO3498:REM INR A
4610 D=A8:GOSUB9140:A8=D:GOTO3498
4620 GOSUB9020:A8=D:GOTO3498:REM MVI A
4630 Z=0:GOSUB9110:IFQ()1THENQ=2
4632 Z1=Q-1:GOSUB9120:GOTO3498
4640 GOTO3498:REM MOV B,B
4650 B8=C8:GOTO3498
4660 B8=D8:GOTO3498
4670 B8=E8:GOTO3498
4680 B8=H8:GOTO3498
4690 B8=L8:GOTO3498
4700 GOSUB9550:B8=D:GOTO3498
4710 B8=A8:GOTO3498
4720 C8=B8:GOTO3498
4730 GOTO3498
4740 C8=D8:GOTO3498
4750 C8=E8:GOTO3498
4760 C8=H8:GOTO3498
4770 C8=L8:GOTO3498
4780 GOSUB9550:C8=D:GOTO3498
4790 C8=A8:GOTO3498
4800 D8=B8:GOTO3498
4810 D8=C8:GOTO3498
4820 GOTO3498
4830 D8=E8:GOTO3498
4840 D8=H8:GOTO3498
4850 D8=L8:GOTO3498
4860 GOSUB9550:D8=D:GOTO3498
4870 D8=A8:GOTO3498
4880 E8=B8:GOTO3498
4890 E8=C8:GOTO3498
4900 E8=D8:GOTO3498
4910 GOTO3498
4920 E8=H8:GOTO3498
4930 E8=L8:GOTO3498
```

```
4940 GOSUB9550:E8=D:GOTO3498
4950 E8=A8:GOTO3498
4960 H8=B8:GOTO3498
4970 H8=C8:GOTO3498
4980 H8=D8:GOTO3498
4990 H8=E8:GOTO3498
5000 GOTO3498
5010 H8=L8:GOTO3498
5020 GOSUB9550:H8=D:GOTO3498
5030 H8=A8:GOTO3498
5040 L8=B8:GOTO3498
5050 L8=C8:GOTO3498
5060 L8=D8:GOTO3498
5070 L8=E8:GOTO3498
5080 L8=H8:GOTO3498
5090 GOTO3498
5100 GOSUB9550:L8=D:GOTO3498
5110 L8=A8:GOTO3498
5120 Z=B8:GOTO9560
5130 Z=C8:GOTO9560
5140 Z=D8:GOTO9560
5150 Z=E8:GOTO9560
5160 Z=H8:GOTO9560
5170 Z=L8:GOTO9560
5180 F=4:D=PA+PB*256:GOSUB8160:PRINT"HLT NA ADRESE"H*:GOTO825
5190 Z=A8:GOTO9560
5200 A8=B8:GOTO3498
5210 A8=C8:GOTO3498
5220 A8=D8:GOTO3498
5230 A8=E8:GOTO3498
5240 A8=H8:GOTO3498
5250 A8=L8:GOTO3498
5260 GOSUB9550:A8=D:GOTO3498
5270 GOTO3498
5280 D=B8:GOTO9570:REM ADD
5290 D=C8:GOTO9570
5300 D=D8:GOTO9570
5310 D=E8:GOTO9570
5320 D=H8:GOTO9570
5330 D=L8:GOTO9570
5340 GOSUB9550:GOTO9570
5350 D=A8:GOTO9570
5360 D=B8:GOTO9580:REM ADC
5370 D=C8:GOTO9580
5380 D=D8:GOTO9580
5390 D=E8:GOTO9580
5400 D=H8:GOTO9580
5410 D=L8:GOTO9580
5420 GOSUB9550:GOTO9580
5430 D=A8:GOTO9580
5440 D=B8:GOTO9590:REM SUB
5450 D=C8:GOTO9590
5460 D=D8:GOTO9590
5470 D=E8:GOTO9590
5480 D=H8:GOTO9590
5490 D=L8:GOTO9590
5500 GOSUB9550:GOTO9590
5510 D=A8:GOTO9590
5520 D=B8:GOTO9600:REM SBB
5530 D=C8:GOTO9600
5540 D=D8:GOTO9600
5550 D=E8:GOTO9600
5560 D=H8:GOTO9600
```



```
5570 D=L8:GOTO9600
5580 GOSUB9550:GOTO9600
5590 D=A8:GOTO9600
5600 Z1=B8:GOTO9610:REM ANA
5610 Z1=C8:GOTO9610
5620 Z1=D8:GOTO9610
5630 Z1=E8:GOTO9610
5640 Z1=H8:GOTO9610
5650 Z1=L8:GOTO9610
5660 GOSUB9550:Z1=D:GOTO9610
5670 Z1=A8:GOTO9610
5680 Z1=B8:GOTO9640:REM XRA
5690 Z1=C8:GOTO9640
5700 Z1=D8:GOTO9640
5710 Z1=E8:GOTO9640
5720 Z1=H8:GOTO9640
5730 Z1=L8:GOTO9640
5740 GOSUB9550:Z1=D:GOTO9640
5750 Z1=A8:GOTO9640
5760 Z1=B8:GOTO9630:REM ORA
5770 Z1=C8:GOTO9630
5780 Z1=D8:GOTO9630
5790 Z1=E8:GOTO9630
5800 Z1=H8:GOTO9630
5810 Z1=L8:GOTO9630
5820 GOSUB9550:Z1=D:GOTO9630
5830 Z1=A8:GOTO9630
5840 D=B8:GOTO9670:REM CMP
5850 D=C8:GOTO9670
5860 D=D8:GOTO9670
5870 D=E8:GOTO9670
5880 D=H8:GOTO9670
5890 D=L8:GOTO9670
5900 GOSUB9550:GOTO9670
5910 D=A8:GOTO9670
5920 Z=6:Z1=0:GOTO9680
5930 GOSUB9700:IFQ=1THEN1615
5935 C8=I:B8=J:GOTO3498
5940 Z=6:Z1=0
5945 GOSUB9110:IFQ=Z1THEN5950
5947 GOSUB9000:GOSUB9000:GOTO3498
5950 GOSUB9020:PA=D:GOSUB9020:PB=D:GOTO3498
5960 Z=6:Z1=0
5965 GOSUB9110:IFQ<>Z1THEN5947
5967 I=PA:J=PB:GOSUB9720:IFQ=1THEN1615
5968 GOTO5950
5970 I=C8:J=B8
5971 GOSUB9720:IFQ=1THEN1615
5975 GOTO3498
5980 GOSUB9020:IFQ=1THEN1615
5985 GOTO9570
5990 F=0
5991 I=PA:J=PB:GOSUB9720:IFQ=0THEN1615
5993 PB=0:PB=F*8:GOTO3498
6000 Z=6:Z1=1:GOTO9680
6010 GOTO9685
6020 Z=6:Z1=1:GOTO5945
6030 GOTO4080
6040 Z=6:Z1=1:GOTO5965
6050 GOTO5967
6060 GOSUB9020:IFQ=1THEN1615
6065 GOTO9580
6070 F=1:GOTO5991
```

```
6080 Z=0:Z1=0:GOTO9680
6090 GOSUB9700:IFQ=1THEN1615
6093 E8=I:D8=J:GOTO3498
6100 Z=0:Z1=0:GOTO5945
6110 GOSUB9020:F=2:GOSUB8160:PRINT"PORT: "H$";:D=A8
6115 GOSUB8160:PRINT" "H$" (DATA OUT)":GOTO3498
6120 Z=0:Z1=0:GOTO5965
6130 I=E8:J=D8:GOTO5971
6140 GOSUB9020:IFQ=1THEN1615
6145 GOTO9590
6150 F2:GOTO5991
6160 Z=0:Z1=1:GOTO9680
6170 GOTO4080
6180 Z=0:Z1=1:GOTO5945
6190 GOSUB9020:F=2:GOSUB8160:PRINT"PORT: "H$" DATA IN ";
6192 INPUTH$:GOSUB8180:IFQ<>0THENPRINT"CHYBA":GOTO6192
6195 A8=D:GOTO3498
6200 Z=0:Z1=1:GOTO5965
6210 GOTO4080
6220 GOSUB9020:IFQ=1THEN1615
6225 GOTO9600
6230 F=3:GOTO5991
6240 Z=2:Z1=0:GOTO9680
6250 GOSUB9700:IFQ=1THEN1615
6255 L8=I:H8=J:GOTO3498
6260 Z=2:Z1=0:GOTO5945
6270 GOSUB9700:IFQ=1THEN1615
6272 F=L8:L8=I:I=F:F=H8:H8=J:J=F:GOSUB9720:GOTO3498
6280 Z=2:Z1=0:GOTO5965
6290 I=L8:J=H8:GOTO5971
6300 GOSUB9020:IFQ=1THEN1615
6305 Z1=D:GOTO9610
6310 F=4:GOTO5991
6320 Z=2:Z1=1:GOTO9680
6330 PA=L8:PB=H8:GOTO3498
6340 Z=2:Z1=1:GOTO5945
6350 F=H8:H8=D8:D8=F:F=L8:L8=E8:E8=F:GOTO3498
6360 Z=2:Z1=1:GOTO5965
6370 GOTO4080
6380 GOSUB9020:IFQ=1THEN1615
6385 Z1=D:GOTO9640
6390 F=5:GOTO5991
6400 Z=7:Z1=0:GOTO9680
6410 GOSUB9700:IFQ=1THEN1615
6415 F8=I:A8=J:GOTO3496
6420 Z=7:Z1=0:GOTO5945
6430 GOTO3498
6440 Z=7:Z1=0:GOTO5965
6450 I=F8:J=A8:GOTO5971
6460 GOSUB9020:IFQ=1THEN1615
6465 Z1=D:GOTO9630
6470 F=6:GOTO5991
6480 Z=7:Z1=1:GOTO9680
6490 SA=L8:SB=H8:GOTO3498
6500 Z=7:Z1=1:GOTO5945
6510 GOTO3498
6520 Z=7:Z1=1:GOTO5965
6530 GOTO4080
6540 GOSUB9020:IFQ=1THEN1615
6545 GOTO9670
6550 F=7:GOTO5991
8000 PRINT G$;: INPUT "CISLA";U: IF U<=J AND U>=0 THEN RETURN
8005 PRINT "CHYBNE CISLO": GOTO 8000
```



```
8009 V=V+1
8010 IF V>L THEN Q=2: RETURN
8015 IF MID$(G$,V,1)=" " THEN 8009
8020 Q=0: RETURN
8030 Q=1: Z=0: H=0: W=V: F$=MID$(G$,V,1)
8035 IF F$>="G" AND F$<="Z" THEN 8070
8040 IF Z=0 THEN H=1
8045 IF F$>="A" AND F$<="F" THEN 8070
8050 IF F$=":" THEN Q=-2: RETURN
8055 IF F$="," THEN Q=-1: RETURN
8060 IF Z=0 THEN H=11
8065 IF F$<"0" OR F$>"9" THEN RETURN
8070 Z=Z+1: IF V+1>L THEN 8080
8075 V=V+1: F$=MID$(G$,V,1): IF F$<>" " THEN 8035
8080 V=V+1: Q=0: RETURN
8085 PRINT U;TAB(5);
8086 G$=S$(U): L=LEN(G$): V=1: GOSUB 8010
8090 IF Q<>0 THEN PRINT: RETURN
8095 GOSUB 8030:IF Q<>-2 THENPRINTTAB(17);:G$=MID$(G$,W):GOTO8115
8100 L$=MID$(G$,W,Z):PRINT L$":"TAB(17);:GOSUB 8009:G$=MID$(G$,V)
8115 PRINT G$: RETURN
8120 PRINT U;TAB(4);:INPUT G$: L=LEN(G$): Q=1:IF G$="@" THEN RETURN
8125 Q=0: RETURN
8130 LPRINT U;TAB(5);:G$=S$(U):L=LEN(G$):V=1:GOSUB 8010
8135 IF Q<>0 THEN LPRINT: RETURN
8140 GOSUB 8030:IF Q<>-2 THENLPRINTTAB(17);:G$=MID$(G$,W):GOTO 8150
8145 L$=MID$(G$,W,Z):LPRINT L$":"TAB(17);:GOSUB 8009:G$=MID$(G$,V)
8150 LPRINT G$: RETURN
8160 H$="":FORZ0=1TOF:H=INT(D/16^(F-Z0)):IFH<9THENF$=STR$(H):GOTO8170
8165 F$=CHR$(ASC("A")-10+H)
8170 H$=H$+F$:D=D-H*16^(F-Z0):NEXT Z0: RETURN
8180 Q=1: D=0
8185 IF LEN(H$)<>F THEN PRINT "NEDOVOLENY TVAR":RETURN
8190 Z0=1:
8192 F$=MID$(H$,Z0,1)
8195 IF F$>="0" AND F$<="9" THEN H=VAL(F$):GOTO 8210
8200 IF F$<"A" OR F$>"F" THEN PRINT "NEDOVOLENA CISLICE":RETURN
8205 H=ASC(F$)-ASC("A")+10
8210 D=D+H*16^(F-Z0): Z0=Z0+1: IF Z0<=F THEN 8192
8215 Q=0: RETURN
8230 RESTORE: P=-1: E=0: T=0: M$=" "
8235 P=P+1: IF P<256 THEN 8250
8240 Q=1:IF E<>0 THEN V=E:G$="NEDOVOLENY OPERAND":RETURN
8245 G$="NEDOVOLENY OPERACNI ZNAK":V=A+T:RETURN
8250 GOSUB 8500:IF LEFT$(D$,4)<>Y$ THEN 8235
8255 P1=P1+1: F=4
8260 IF C=1 THEN 8395
8265 IF C=3 THEN 8410
8270 IF C=7 THEN 8420
8275 IF P1>0 THEN 8342
8280 GOSUB 8010: IF Q<>0 THEN G$="CHYBI OPERAND": RETURN
8285 GOSUB 8030:Z0=Z:B=V:E=W:IF Q<>0 THEN 8315
8290 IF C=2 THEN 8335
8300 GOSUB 8010: IF Q<>0 THEN G$="CHYBI ','": RETURN
8305 GOSUB 8030:B=V: IF Q=-1 THEN 8335
8310 V=W:Q=1:G$="',' SYNTAXE": RETURN
8315 IF Q<>-1 THEN 8330
8320 IF C>3 THEN 8335
8325 GOTO 8405
8330 Q=1: G$="SYNTAXE OPERANDU": RETURN
8335 IF Z0=0 THEN V=E:GOTO 8330
8340 E$=MID$(G$,E,Z0)
8342 IF LEN(Z$)<>LEN(E$) OR Z$<>E$ THEN 8385
```

```
8345 V=B: IF C=2 THEN 8395
8350 V=V+1: IF C=5 THEN 8410
8355 IF C=6 THEN 8420
8360 GOSUB 8010: IF Q<>0 THEN G$="CHYBI OPERAND": RETURN
8365 GOSUB 8030:E=W:IF Q<>0 THEN 8330
8370 IF Z=1 THEN 8380
8375 V=W+1: Q=1: G$="DELKA OPERANDU": RETURN
8380 IF X$=MID$(G$,W,1) THEN 8395
8385 W=A: GOTO 8235
8390 V=V+1
8395 IF V>L THEN Q=0:RETURN
8400 F$=MID$(G$,V,1): IF F$=" " THEN 8390
8405 Q=1: G$="SYNTAXE KONCE PRIKAZU": RETURN
8410 F=2
8415 IF D$="SHLD" OR D$="LHLD" OR D$="STA " OR D$="LDA " THEN F=4
8420 GOSUB 8010: IF Q=0 THEN 8430
8425 G$="CHYBI NAVESTI, PROMENNA NEBO DATA": RETURN
8430 GOSUB 8030:IF Q=0 THEN 8434
8432 G$="SYNTAXE NAVESTI, PROMENNE NEBO DAT":RETURN
8434 IF Z=0 THEN 8447
8435 M$=MID$(G$,W,Z): IF H=0 THEN 8440
8437 IF Z=F+1 THEN 8450
8440 IF H>1 THEN 8446
8442 IF Z<=10 THEN 8395
8444 V=W+10:Q=1: G$="DELKA IDENTIFIKATORU": RETURN
8446 IF H<=1 THEN 8448
8447 V=W
8448 Q=1:GOTO8432
8450 R=0:F$=MID$(M$,F+1,1): IF F$<>"D"ANDF$<>"H" THEN 8440
8452 Z$=MID$(M$,R+1,1):IF Z$>="0" AND Z$<="9" THEN 8458
8454 IFF$<>"H" THEN V=W+R: Q=1: G$="CHYBA V CISLE": RETURN
8456 IF Z$<"A" OR Z$>"F" THEN V=W+R:Q=1:G$="CHYBA V HEX.C.":RETURN
8458 R=R+1:IF R<F THEN 8452
8460 P1=777: GOTO 8395
8500 Q=1:READ Y$:IF Y$=" " THEN RETURN
8505 Q=0:Z$=" ":X$=" ": READ C
8510 IF C=1 OR C=3 OR C=7 THEN RETURN
8515 READ Z$:IF C<>4 THEN RETURN
8520 READ X$:RETURN
8530 V=1:GOSUB 8010:IF Q<>0 THEN G$="CHYBI INSTRUKCE":GOTO 8550
8535 L$=" ":GOSUB 8030:IF Q<>-2 THEN 8575
8540 IF H<>11 THEN 8555
8543 V=W:G$="SYNTAXE NAVESTI":GOTO 8550
8545 G$="SYNTAXE INSTRUKCE"
8550 PRINT TAB(V+S+17)"^":PRINT G$:Q=1:RETURN
8555 IF Z>10 THEN V=W+10:G$="DELKA NAVESTI":GOTO8550
8560 IF Z=0 THEN 8543
8565 L$=MID$(G$,W,Z):GOSUB8009:IFQ<>0THENG$="CHYBI INSTRUKCE":GOTO8550
8570 GOSUB 8030
8575 IF Q<>0 THEN 8545
8580 IF Z>4 THEN G$="DELKA INSTRUKCE":V=W+4:GOTO8550
8585 IF Z<2 THEN V=W:GOTO 8545
8590 D$=MID$(G$,W,Z):IF Z=4 THEN 8605
8595 IF Z<=2 THEN D$=D$+" "
8600 D$=D$+" "
8605 A=W:P1=-1:GOSUB 8230:IF Q<>0 THEN 8550
8610 RETURN
8620 Q=1:Z0=0:IF W5+W6=0 THEN RETURN
8625 IF S$(N-Z0)=G$ THEN A=Z0:Q=0:RETURN
8630 Q=1:Z0=Z0+1:IF Z0<W5+W6 THEN 8625
8635 RETURN
8640 Q=1:IF C=6 THEN RETURN
8645 IF D$="STA "OR D$="LDA "OR D$="LHLD"OR D$="SHLD" THEN RETURN
```



```
8650 Q=0: RETURN
8660 D=0:Z0=1
8665 F$=MID$(H$,Z0,1):H=VAL(F$):D=D+H*10^(F-Z0):Z0=Z0+1
8670 IF Z0<>F THEN 8665
8675 RETURN
8700 A8=0:B8=0:C8=0:D8=0:E8=0:H8=0:L8=0:F8=2
8705 SA=0:SB=0:PA=0:PB=0
8710 PRINT"VSECHNY REGISTRY VYNULOVANY":RETURN
8720 IFW6=0THEN PRINT"NEJSOU SYMBOLICKE PROMENNE":RETURN
8725 Z0=0
8727 G$=S$(N-Z0):F$=MID$(G$,1,1)
8730 IFF$=":"THEN8740
8735 D=N-W5-W6-ABS(W(M-Z0))+W8:W(D)=0:IF W(M-Z0)<0THENW(D-1)=0
8740 Z0=Z0+1:IF Z0<W5+W6 THEN 8727
8745 PRINTW6"SYMBOLICKYCH PROMENNYCH NULOVANO":RETURN
8750 WJ=0:WV=0:WT=0:WC=7:WP=0:WS=0:Q=N-W5-W6-WD-W4-32
8755 FOR Z0=W4 TO Q+W4:W(Z0)=0:NEXT Z0
8760 PRINT Q+1"BYTU PRACOVNI OBLASTI NULOVANO":RETURN
8770 F=4:Z1=W7:Q=W7+W4-1:PRINT"PROGRAM":GOSUB8790
8775 Z1=W8:Q=W8+WD-1:PRINT"SYMBOLICKE PROMENNE":GOSUB8790
8780 Z1=WV:Q=WV+WC-1:PRINT"1. PRACOVNI OBLAST":GOSUB8790
8785 Z1=WT:Q=WT+WP-1:PRINT"2. PRACOVNI OBLAST":GOSUB8790:RETURN
8790 IF Q<Z1 THEN PRINT" NEDEFINOVANO":RETURN
8795 D=Z1:GOSUB 8160: PRINT TAB(20)"OD "H$;
8798 D=Q:GOSUB 8160: PRINT " DO ";H$:RETURN
8800 Q=0:IF D>=W7 AND D<=W7+W4-1 THEN H=D-W7:RETURN
8805 IF WD=0 OR D<W8 THEN 8815
8810 IF D<=W8+WD-1 THEN H=M-W5-W6+W8-D:RETURN
8815 IF WC=0 OR D<WV THEN 8825
8820 IF D<=WV+WC-1 THEN H=W4+D-WV:RETURN
8825 IF WP=0 OR D<WT THEN 8835
8830 IF D<=WT+WP-1 THEN H=W4+WC+D-WT:RETURN
8835 Q=1:RETURN
8840 W1=D:D=W0
8843 GOSUB 8800: IF Q=0 THEN Q=1:RETURN
8845 D=D+1:IF D<=W1 THEN 8843
8850 Z0=M-W5-W6-WD-32-W4-WC-WP-W1+W0
8855 IF Z0>=0 THEN PRINT Z0"BYTU PRACOVNI OBLASTI ZBYVA":Q=0:RETURN
8860 PRINT"OBLAST JE PREPLNENA":F=4:D=W0+M-W5-W6-WD-32-W4-WC-WP
8865 GOSUB 8160:PRINT "POSLEDNI POUZITELNA ADRESA JE "H$:Q=2:RETURN
8870 D=Z1:GOSUB 8160:PRINT G$" : "H$;:INPUT H$
8875 IF H$="@" OR H$="*" THEN RETURN
8880 GOSUB 8180:IF Q<>0 THEN 8870
8885 RETURN
9000 PA=PA+1:IF PA<256 THEN 9010
9005 PA=0:PB=PB+1:IF PB=256 THEN PB=0
9010 D=PA+PB*256:RETURN
9020 GOSUB 9000:GOSUB 8800:IF Q=0 THEN D=W(H):RETURN
9025 D=255:RETURN
9030 Z1=Z1+1:IF Z1<256 THEN RETURN
9035 Z1=0:Z=Z+1:IF Z<256 THEN RETURN
9040 Z=0:RETURN
9050 D=D+1:IF D>=256 THEN D=0
9055 Z=7:GOSUB 9100:Z1=0:IF Q<>0 THEN Z1=1
9060 GOSUB 9120:Z=6:Z1=0:IF D<>0 THEN Z1=1
9065 GOSUB 9120:Z=3:GOSUB 9100:Z=4
9070 Z1=0:IF Q<>0 THEN Z1=1
9075 GOSUB 9120:H=0:FOR Z=0 TO 7:GOSUB 9100
9080 IF Q<>0 THEN H=H+1
9085 NEXT Z:Z1=0:IF INT(H/2)=H/2 THEN Z1=1
9090 Z=2:GOSUB9120:RETURN
9100 Q=INT(D/A^Z)-2*INT(D/2^(Z+1)):RETURN
9110 Q=INT(F8/2^Z)-2*INT(F8/2^(Z+1)):RETURN
```

```
9120 GOSUB 9110:IF Q<>0 THEN F8=F8-2^Z
9130 F8=F8+Z1*2^Z:RETURN
9140 D=D-1:IF D)=0 THEN 9055
9145 D=255:GOTO 9055
9150 Z=7:GOSUB9100:Z=0:Z1=Q
9152 GOSUB9120:H=0
9155 H=H+Q*2^Z:GOSUB9100:Z=Z+1:IF Z<8THEN9155
9160 A8=H:GOTO3498
9180 L8=L8+Z1:IFL8<256THEN9190
9185 L8=L8-256:H8=H8+1
9190 H8=H8+Z:Z1=0:IFH8<256THEN9200
9195 H8=H8-256:Z1=1
9200 Z=0:GOSUB9120:GOTO3498
9210 GOSUB8800:IFQ=1THEN1615
9215 A8=W(H):GOTO3498
9220 Z1=Z1-1:IFZ1)=0THENRETURN
9225 Z1=255:Z=Z-1:IFZ)=0THENRETURN
9230 Z=255:RETURN
9250 Z=0:GOSUB9100:Z1=Q
9252 GOSUB9120:Z=7:H=0
9255 H=H+Q*2^Z:GOSUB9100:Z=Z-1:IFZ)=0THEN9255
9260 A8=H:GOTO3498
9330 Z=7:GOSUB9100:Z1=Q:Z=0:GOSUB9110:GOTO9152
9360 Z=0:GOSUB9100:Z1=0:Z=0:GOSUB9110:GOTO9252
9380 GOSUB9020:Z1=D:GOSUB9020:Z=D:D=Z1+Z*256:GOSUB8800:RETURN
9420 H=0:FORZ=0TO3:GOSUB9100:H=H+Q*2^Z:NEXTZ
9425 F=0:IFH)91THEN9435
9430 Z=0:GOSUB9110:IFQ=0THEN9445
9435 D=D+6:IFD<256THEN9445
9440 D=D-256:F=1
9445 H=0:FORZ=4TO7:GOSUB9100:H=H+Q*2^(Z-4):NEXTZ
9450 IFH<=9THENIFF=0THEN9465
9455 F=0:D=D+96:IFD<256THEN9465
9460 D=D-256:F=1
9465 Z=0:Z1=F:GOSUB9120:GOSUB9055:GOTO3498
9500 H=0:FORZ=0TO7:GOSUB9100:IFQ<>1THENH=H+2^Z
9505 NEXTZ:A8=H:GOTO3498
9520 GOSUB8800:IFQ=1THEN1615
9525 D=W(H):GOSUB9050:W(H)=D:GOTO3498
9530 GOSUB8800:IFQ=1THEN1615
9535 D=W(H):GOSUB9140:W(H)=D:GOTO3498
9540 GOSUB8800:IFQ=1THEN1615
9545 W(H)=F:D=F:GOTO3498
9550 D=L8+H8*256:GOSUB8800:IFQ=1THEN1615
9555 D=W(H):RETURN
9560 D=L8+H8*256:GOSUB8800:IFQ=1THEN1615
9565 W(H)=Z:GOTO3498
9570 Z1=0:A8=A8+D:IFA8)=256THENA8=A8-256:Z1=1
9575 Z=0:GOSUB9120:D=A8:GOSUB9055:GOTO3498
9580 Z=0:GOSUB9110:D=D+Q:GOTO9570
9590 Z1=0:A8=A8-D:IFA8<0THENA8=A8+256:Z1=1
9595 GOTO9575
9600 Z=0:GOSUB9110:D=D+Q:GOTO9590
9610 H=0:FORZ=0TO7:D=A8:GOSUB9100:IFQ=0THEN9615
9612 D=Z1:GOSUB9100:IFQ<>0THENH=H+2^Z
9615 NEXTZ:A8=H:Z=3:D=Z1:GOSUB9100:F=0:Z=0:Z1=0
9618 GOSUB9120:D=A8:GOSUB9055:Z1=0:IFF=1THEN9622
9620 Z=3:D=A8:GOSUB9100:IFQ=0THEN9625
9622 Z1=1
9625 GOSUB9120:GOTO3498
9630 F=1:GOTO9642
9640 F=0
9642 H=0:FORZ=0TO7:D=A8:GOSUB9100:D=Z1:IFQ=0THEN9652
```



```
9645 IFF=1THEN9655
9647 GOSUB9100:IFQ=0THEN9655
9650 GOTO9657
9652 GOSUB9100:IFQ=0THEN9657
9655 H=H+2^Z
9657 NEXTZ:A8=H:D=H:GOSUB9055:Z=0:Z1=0
9660 GOSUB9120:Z=4:GOSUB9120:GOTO3498
9670 Z1=0:D=A8-D:IFD<0THEND=D+256:Z1=1
9675 Z=0:GOSUB9120:GOSUB9055:GOTO3498
9680 GOSUB9110:IFQ<>Z1THEN3498
9685 GOSUB9700:IFQ=1THEN1615
9690 PA=I:PB=J:GOTO3498
9700 D=SA+SB*256:GOSUB8800:IFQ=1THENRETURN
9705 I=W(H):Z=SB:Z1=SA:GOSUB9030:D=Z1+Z*256
9710 GOSUB8800:IFQ=1THENRETURN
9715 J=W(H):GOSUB9030:SB=Z:SA=Z1:RETURN
9720 Z=SB:Z1=SA:GOSUB9220:D=Z1+Z*256:GOSUB8800:IFQ=1THENRETURN
9725 W(H)=J:GOSUB9030:D=Z1+Z*256:GOSUB8800:IFQ=1THENRETURN
9730 W(H)=I:SB=Z:SA=Z1:RETURN
10000 DATA NOP ,1,LXI ,6,B,STAX,2,B,INX ,2,B,INR ,2,B
10010 DATA DCR ,2,B,MVI ,5,B,RLC ,1," ",DAD ,2,B,LDAX,2,B
10020 DATA DCX ,2,B,INR ,2,C,DCR ,2,C,MVI ,5,C,RRC ,1
10030 DATA " ",LXI ,6,D,STAX,2,D",INX ,2,D,INR ,2,D,DCR ,2,D,MVI ,5,D
10040 DATA RAL ,1," ",DAD ,2,D,LDAX,2,D,DCX ,2,D,INR ,2,E,DCR ,2,E
10050 DATA MVI ,5,E,RAR ,1," ",LXI ,6,H,SHLD,3,INX ,2,H,INR,2,H
10060 DATA DCR ,2,H,MVI ,5,H,DAA ,1," ",DAD ,2,H,LHLD,3,DCX ,2,H
10070 DATA INR ,2,L,DCR ,2,L,MVI ,5,L,CMA ,1," ",LXI ,6,SP,STA ,3
10080 DATA INX ,2,SP,INR ,2,M,DCR ,2,M,MVI ,5,M,STC ,1," "
10090 DATA DAD ,2,SP,LDA ,3,DCX ,2,SP,INR ,2,A,DCR ,2,A,MVI ,5,A
10100 DATA CMC ,1,MOV ,4,B,B,MOV ,4,B,C,MOV ,4,B,D,MOV ,4,B,E
10110 DATA MOV ,4,B,H,MOV ,4,B,L,MOV ,4,B,M,MOV ,4,B,A,MOV ,4,C,B
10120 DATA MOV ,4,C,C,MOV ,4,C,D,MOV ,4,C,E,MOV ,4,C,H,MOV ,4,C,L
10130 DATA MOV ,4,C,M,MOV ,4,C,A,MOV ,4,D,B,MOV ,4,D,C,MOV ,4,D,D
10140 DATA MOV ,4,D,E,MOV ,4,D,H,MOV ,4,D,L,MOV ,4,D,M,MOV ,4,D,A
10150 DATA MOV ,4,E,B,MOV ,4,E,C,MOV ,4,E,D,MOV ,4,E,E,MOV ,4,E,H
10160 DATA MOV ,4,E,L,MOV ,4,E,M,MOV ,4,E,A,MOV ,4,H,B,MOV ,4,H,C
10170 DATA MOV ,4,H,D,MOV ,4,H,E,MOV ,4,H,H,MOV ,4,H,L,MOV ,4,H,M
10180 DATA MOV ,4,H,A,MOV ,4,L,B,MOV ,4,L,C,MOV ,4,L,D,MOV ,4,L,E
10190 DATA MOV ,4,L,H,MOV ,4,L,L,MOV ,4,L,M,MOV ,4,L,A,MOV ,4,M,B
10200 DATA MOV ,4,M,C,MOV ,4,M,D,MOV ,4,M,E,MOV ,4,M,H,MOV ,4,M,L
10210 DATA HLT ,1,MOV ,4,M,A,MOV ,4,A,B,MOV ,4,A,C,MOV ,4,A,D
10220 DATA MOV ,4,A,E,MOV ,4,A,H,MOV ,4,A,L,MOV ,4,A,M,MOV ,4,A,A
10230 DATA ADD ,2,B,ADD ,2,C,ADD ,2,D,ADD ,2,E,ADD ,2,H,ADD ,2,L
10240 DATA ADD ,2,M,ADD ,2,A,ADC ,2,B,ADC ,2,C,ADC ,2,D,ADC ,2,E
10250 DATA ADC ,2,H,ADC ,2,L,ADC ,2,M,ADC ,2,A,SUB ,2,B,SUB ,2,C
10260 DATA SUB ,2,D,SUB ,2,E,SUB ,2,H,SUB ,2,L,SUB ,2,M,SUB ,2,A
10270 DATA SBB ,2,B,SBB ,2,C,SBB ,2,D,SBB ,2,E,SBB ,2,H,SBB ,2,L
10275 DATA SBB ,2,M,SBB ,2,A,ANA ,2,B,ANA ,2,C,ANA ,2,D,ANA ,2,E
10280 DATA ANA ,2,H,ANA ,2,L,ANA ,2,M,ANA ,2,A,XRA ,2,B,XRA ,2,C
10290 DATA XRA ,2,D,XRA ,2,E,XRA ,2,H,XRA ,2,L,XRA ,2,M,XRA ,2,A
10300 DATA ORA ,2,B,ORA ,2,C,ORA ,2,D,ORA ,2,E,ORA ,2,H,ORA ,2,L
10310 DATA ORA ,2,M,ORA ,2,A,CMP ,2,B,CMP ,2,C,CMP ,2,D,CMP ,2,E
10320 DATA CMP ,2,H,CMP ,2,L,CMP ,2,M,CMP ,2,A,RNZ ,1,POP ,2,B
10330 DATA JNZ ,7,JMP ,7,CNZ ,7,PUSH,2,B,ADI ,3,RST ,2,0,RZ  ,1
10340 DATA RET ,1,JZ  ,7," ",CZ  ,7,CALL,7,ACI ,3,RST ,2,1
10350 DATA RNC ,1,POP ,2,D,JNC ,7,OUT ,3,CNC ,7,PUSH,2,D,SUI ,3
10360 DATA RST ,2,2,RC  ,1," ",JC  ,7,IN  ,3,CC  ,7," ",SBI ,3
10370 DATA RST ,2,3,RPO ,1,POP ,2,H,JPO ,7,XTHL,1,CPO ,7,PUSH,2,H
10380 DATA ANI ,3,RST ,2,4,RPE ,1,PCHL,1,JPE ,7,XCHG,1,CPE ,7," "
10390 DATA XRI ,3,RST ,2,5,RP  ,1,POP ,2,PSW,JP  ,7,DI  ,1
10400 DATA CP  ,7,PUSH,2,PSW,ORI ,3,RST ,2,6,RM  ,1,SPHL,1
10410 DATA JM  ,7,EI  ,1,CM  ,7," ",CPI ,3,RST ,2,7
```

## 8.6. PŘEKLAD PROGRAMU V JAZYKU SYMBOLICKÝCH ADRES

### Zadání

Seznamte se s činností a ovládáním překladače jazyka symbolických adres mikroprocesoru 8080.

### Úvod

Jazyk symbolických adres je programovací jazyk nejnižší úrovně, který se v praxi používá při programování počítačů. Instrukce jsou zde na rozdíl od strojového kódu zapisovány symbolickou formou. Zápis programu je tak čitelnější a srozumitelnější než zápis ve strojovém kódu. To umožňuje snadnější provádění změn v programech oproti programům ve strojovém kódu. Překlad programu z jazyka symbolických adres do strojového jazyka provádí překladač (asembler).

Příkazy jazyka symbolických adres se dělí do těchto tří skupin:

a) instrukce – přímo odpovídající strojovým instrukcím;

b) pseudoinstrukce – příkazy pro překladač;

c) makroinstrukce – zkrácený zápis předem definované posloupnosti instrukcí.

Každá *instrukce* obsahuje operační znak a může obsahovat také návěští a operandy. Návěští je symbolické jméno adresy, na níž je instrukce umístěna; je ukončeno znakem dvojtečky. Návěští se používá tehdy, jestliže se na instrukci odvolává jiná instrukce, např. skoková instrukce. Operační znak tvoří symbolické jméno instrukce. Operandy specifikují (popisují) hodnoty, s nimiž se vykonávají operace předepsané instrukcemi. U bezadresových instrukcí se operandy neuvádějí. Jako operandy mohou vystupovat symbolická jména registrů, návěští atd. Pokud instrukce obsahují více operandů, jsou od sebe odděleny čárkou.

*Pseudoinstrukce* jsou příkazy pro překladač. Patří k nim příkazy určující umístění programu a dat v paměti, příkazy pro rezervování paměťových míst a příkazy pro řízení překladu a pro určení výpisu informací o překladu. Pseudoinstrukce se obvykle zapisují stejným způsobem jako instrukce.

Některé překladače umožňují používat také *makroinstrukce*. Každý zápis makroinstrukce ve zdrojovém programu způsobí vložení do zdrojového programu předem určené posloupnosti instrukcí.



Program zapsaný v jazyku symbolických adres se překládá do cílového jazyka překladačem nazývaným *asembler*. Překladač čte postupně příkazy zdrojového textu programu, přiřazuje symbolickým jménům operačních znaků strojový kód a přiřazuje instrukcím adresy paměťových míst. Překlad samotné instrukce je poměrně jednoduchý, neboť jedné zdrojové instrukci odpovídá jedna cílová instrukce. Potíže vznikají jen tehdy, obsahuje-li instrukce odkaz na objekt, který nebyl zatím ještě definován (např. návěští použité až v další části programu). Z tohoto důvodu většina překladačů pracuje tak, že prochází daný program dvakrát a používá dvě tabulky. Jednou z nich je tabulka definovaných symbolických jmen adres a operandů. Tato tabulka se vytváří při prvním průchodu zdrojového programu. Druhou je tabulka obsahující symbolická jména instrukcí a pseudoinstrukcí. Teprve při druhém průchodu dochází k vlastnímu překladu s využitím této tabulky.

### Postup cvičení

V programu, který je uveden v čl. 8.5, je obsažen také jednoduchý překladač (příkazy 400 až 795, podprogramy 8160 až 8675 a data 10000 až 10410). Tento překladač se spouští příkazem TRANS. Vzhledem k tomu, že prohlížení tabulky symbolických jmen instrukcí je zde značně pomalé, byl tento překladač vytvořen jako jednoprůchodový. Tím byl sice zkrácen čas překladu asi na polovinu, ale stále je ještě překlad pomalý. Tento překladač překládá pouze instrukce a nedokáže zpracovat pseudoinstrukce nebo makroinstrukce. Při zahájení překladu je zde vypsán dotaz na počáteční adresu umístění programu a počáteční adresu umístění tabulky, ve které se v počítači uschová informace o symbolických proměnných. Na tyto dotazy je nutné odpovědět čtyřmístným číslem v šestnáctkové soustavě. Po zadání těchto údajů je zahájen překlad zdrojového programu, který byl zadán pomocí textového editoru (viz předchozí cvičení). Při nalezení chyby v tomto programu překladač ihned vyžaduje její opravu. Je tedy nutné opravenou instrukci zadat z klávesnice. Pokud v programu není uvedeno některé návěští, je překlad ukončen chybovou signalizací. Na závěr překladu je vypsán přeložený program ve strojovém kódu.

Na tomto programu se můžete seznámit, jakým způsobem pracuje překladač jazyka symbolických adres. Pro vlastní překlad jej ale používejte tehdy, pokud nemáte k dispozici žádný jiný dokonalejší překladač.

Pokuste se přeložit následující podprogram, který slouží pro násobení celých čísel.

```
NAS:  MVI  L,Ø
      MOV  D,L
      MOV  H,B
      MVI  C,8
NAS1: DAD  H
      JNC  NAS2
      DAD  D
NAS2: DCR  C
      JNZ  NAS1
      RET
```

Tento program násobí dvě dvojková čísla bez znaménka (s délkou jedné slabiky) uložená v registrech B a E. Výsledek je uložen ve dvojici registrů HL.

## 8.7. SLEDOVÁNÍ PRŮBĚHU VÝPOČTU

### Zadání

Sledujte průběh výpočtu jednoduchého programu.

### Úvod

Pro sledování průběhu výpočtu můžete použít zbývající část programu uvedeného v čl. 8.5. Je to část, která provádí vlastní simulaci činnosti mikroprocesoru 8080. Simulaci spouštíme příkazem SIM a můžeme si vybrat z těchto funkcí:

CLEAR – nulování registrů, nulování obsahu symbolických proměnných a iniciace pracovní oblasti simulátoru;

START – spuštění vlastní simulace (je nutné zadat počáteční adresu programu);

CONT – pokračování v simulaci;

BREAK – zadání až 32 adres, na kterých bude provedeno přerušení průběhu simulace;



SET – dosazení potřebných hodnot do registrů, symbolických proměnných a libovolného paměťového místa;

DISPLAY – výpis obsahů registrů, symbolických proměnných a libovolného paměťového místa;

TRACE – pokud se tento příkaz vyskytne před příkazem START nebo CONT, jsou vypisovány informace o průběhu simulace, v opačném případě simulace probíhá, ale informace vypisovány nejsou;

END – ukončení simulace.

### Postup cvičení

Pomocí tohoto programu můžeme sledovat průběh výpočtu programu, který jsme přeložili v předchozím cvičení (podprogram pro násobení čísel). Program byl přeložen od adresy 3000. Nejdříve pomocí příkazu SET dosadíme do registrů B a E čísla, která chceme násobit, a příkazy TRACE a START spustíme vlastní simulaci s výpisem informací o průběhu simulace. Z vypisovaných informací se pokuste zjistit, jakým způsobem se provádí násobení čísel.

Následuje ukázka vypisovaných informací o průběhu výpočtu. Tato ukázka je výstupem z dokonalejšího simulačního programu. Informace o provedení každé instrukce jsou zde uvedeny na dvou řádcích. V prvním řádku je uvedena adresa instrukce a vlastní instrukce v symbolickém tvaru (pouze adresy jsou uváděny číselně). Druhý řádek zahrnuje obsahy jednotlivých registrů po provedení instrukce. Výpis z našeho programu by se lišil pouze v tom, že by zde nebyly vypisovány instrukce v symbolickém tvaru, ale byly by vypisovány pouze číselné operační znaky bez případných adres.

```
 AF  BC  DE  HL    S    P
3000 MVI  L,00
0056 3230 002E 3000 7F1C 3002
3002 MOV  D,L
0056 3230 002E 3000 7F1C 3003
3003 MOV  H,B
0056 3230 002E 3200 7F1C 3004
3004 MVI  C,08
0056 3208 002E 3200 7F1C 3006
3006 DAD  H
0056 3208 002E 6400 7F1C 3007
3007 CNC  300B
0056 3208 002E 6400 7F1A 300B

 AF  BC  DE  HL    S    P
300B DCR  C
0012 3207 002E 6400 7F1A 300C
300C JNC  3006
```

```
0012 3207 002E 6400 7F1A 3006
3006 DAD  H
0012 3207 002E C800 7F1A 3007
3007 CNC  300B
0012 3207 002E C800 7F18 300B
300B DCR  C
0016 3206 002E C800 7F18 300C
300C JNC  3006
0016 3206 002E C800 7F18 3006

 AF  BC  DE  HL   S   P
```

Z vypisovaných informací můžeme sledovat, jak jednotlivé instrukce mění obsahy registrů, a sledovat, jak probíhá výpočet. To nám může usnadnit hledání chyb při ladění programů zapsaných v jazyku symbolických adres.

V tomto cvičení se pokuste zjistit, jakým způsobem se provádí násobení celých čísel (podprogram pro násobení záměrně není opatřen komentáři). Dále se pokuste odladit pomocí simulace podprogram pro sčítání celých čísel umístěných ve třech slabikách.

## 8.8. VYUŽITÍ BODŮ ZASTAVENÍ

### Zadání

Sledujte průběh výpočtu složitějšího programu.

### Úvod

Při ladění programů se může vyskytnout situace, kdy máme jistotu, že hledaná chyba je v určité části programu. Je tedy zbytečné vypisovat informace o průběhu celého výpočtu.

### Postup cvičení

Můžeme tedy zahájit simulaci bez výpisu informací o průběhu výpočtu. Tuto simulaci přerušíme pomocí bodů zastavení v okamžiku, kdy se výpočet dostane do kritického místa programu. Body zastavení je nutné před spuštěním simulace zadat. Dále budeme pokračovat v simulaci příkazy TRACE a CONT již s výpisem informací. Pokud se dostaneme z kritického místa, můžeme pomocí dalšího bodu zastavení opět přerušit simulaci.



Tímto způsobem je možné část simulace provádět s výpisem informací a část bez výpisu. Je tedy možné získat výpisy o průběhu výpočtu pouze z těch částí programu, které nás zajímají. Toto si můžeme ukázat na následujícím podprogramu pro dělení celých čísel.

```
DEL:  MOV  A,D
      CMA
      MOV  D,A
      MOV  A,E
      CMA
      MOV  E,A
      INX  D
      LXI  H,0
      MVI  A,17D
DEL1: PUSH H
      DAD  D
      JNC  DEL2
      XTHL
DEL2: POP  H
      PUSH PSW
      MOV  A,C
      RAL
      MOV  C,A
      MOV  A,B
      RAL
      MOV  B,A
      MOV  A,L
      RAL
      MOV  L,A
      MOV  A,H
      RAL
      MOV  H,A
      POP  PSW
      DCR  A
      JNZ  DEL1
      ORA  A
      MOV  A,H
      RAR
```

```
MOV  D,A
MOV  A,L
RAR
MOV  E,A
RET
```

Podprogram dělí celá čísla (o délce dvou slabik). Před vyvoláním podprogramu je nutné dělence umístit do dvojice registrů BC, dělitele do dvojice registrů DE. Po provedení výpočtu je podíl uložen v registrech BC a zbytek v registrech DE.

Tento podprogram je třeba pomocí textového editoru zapsat a přeložit překladačem. Při simulaci nejprve sledujte průběh celého výpočtu a pokuste se zjistit, jakým způsobem probíhá dělení. Dále zahajte simulaci znova a pokuste se vypisovat informace o průběhu výpočtů pouze u instrukcí od návěští DEL1 do instrukce s návěštím DEL2.

Při cvičení se pokuste zjistit princip dělení celých čísel, který je použit v uvedeném podprogramu, a pokuste se navrhnout jiný způsob dělení čísel.



# Dodatek 1.

TABULKA KÓDU ISO 7

| b7 | Ø | Ø | Ø | Ø | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| b6 | Ø | Ø | 1 | 1 | Ø | 1 | 1 | 1 |
| b5 | Ø | 1 | Ø | 1 | Ø | 1 | Ø | 1 |
| b4 b3 b2 b1 | | | | | | | | |
| Ø Ø Ø Ø | NULL | DLE | | Ø | @ | P | | p |
| Ø Ø Ø 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| Ø Ø 1 Ø | STX | DC2 | " | 2 | B | R | b | r |
| Ø Ø 1 1 | ETX | DC3 | # | 3 | C | S | c | s |
| Ø 1 Ø Ø | EOT | DC4 | $ | 4 | D | T | d | t |
| Ø 1 Ø 1 | ENT | NAK | % | 5 | E | U | e | u |
| Ø 1 1 Ø | ACK | SYN | & | 6 | F | V | f | v |
| Ø 1 1 1 | BELL | ETB | ' | 7 | G | W | g | w |
| 1 Ø Ø Ø | BS | CAN | ( | 8 | H | X | h | x |
| 1 Ø Ø 1 | HT | EM | ) | 9 | I | Y | i | y |
| 1 Ø 1 Ø | LF | SUB | * | : | J | Z | j | z |
| 1 Ø 1 1 | VT | ESC | + | ; | K | [ | k | { |
| 1 1 Ø Ø | FF | FS | , | < | L | \ | l | |
| 1 1 Ø 1 | CR | GS | − | = | M | ] | m | } |
| 1 1 1 Ø | SO | RS | . | > | N | ∩ | n | |
| 1 1 1 1 | SI | US | / | ? | O | − | o | DEL |

# Dodatek 2.

## 2.1. TABULKA MOCNIN 2, 8 A 16

| $2^n$ | $n$ |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 4 | 2 |
| 8 | 3 |
| 16 | 4 |
| 32 | 5 |
| 64 | 6 |
| 128 | 7 |
| 256 | 8 |
| 512 | 9 |
| 1 024 | 10 |
| 2 048 | 11 |
| 4 096 | 12 |
| 8 192 | 13 |
| 16 384 | 14 |
| 32 768 | 15 |
| 65 536 | 16 |

| $8^n$ | $n$ |
|---|---|
| 1 | 0 |
| 8 | 1 |
| 64 | 2 |
| 512 | 3 |
| 4 096 | 4 |
| 32 768 | 5 |
| 262 144 | 6 |

| $16^n$ | $n$ |
|---|---|
| 1 | 0 |
| 16 | 1 |
| 256 | 2 |
| 4 096 | 3 |
| 65 536 | 4 |

## 2.2. PŘEVOD MEZI ČÍSLY V OSMIČKOVÉ A DESÍTKOVÉ SOUSTAVĚ

a) Z osmičkové do desítkové soustavy

Každé číslici z osmičkové soustavy z odpovídajícího sloupce tabulky přiřaďte desítkovou hodnotu. Sečtením těchto hodnot získáte desítkové vyjádření čísla v osmičkové soustavě.

b) Z desítkové do osmičkové soustavy

Najděte, ve kterém sloupci tabulky je největší část z převáděného čísla. Zapište si číslo řádku jako číslici čísla v osmičkové soustavě. S dalšími čísly proveďte totéž.



| Osmičkové sloupce | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | | 5 | | 4 | | 3 | | 2 | | 1 | |
| OKT = DEK | | OKT = DEK | | OKT = DEK | | OKT = DEK | | OKT = DEK | | OKT = DEK | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 32 768 | 1 | 4 096 | 1 | 512 | 1 | 64 | 1 | 8 | 1 | 1 |
| 2 | 65 536 | 2 | 8 192 | 2 | 1 024 | 2 | 128 | 2 | 16 | 2 | 2 |
| 3 | 98 304 | 3 | 12 288 | 3 | 1 536 | 3 | 192 | 3 | 24 | 3 | 3 |
| 4 | 131 072 | 4 | 16 384 | 4 | 2 048 | 4 | 256 | 4 | 32 | 4 | 4 |
| 5 | 163 840 | 5 | 20 480 | 5 | 2 560 | 5 | 320 | 5 | 40 | 5 | 5 |
| 6 | 196 608 | 6 | 24 576 | 6 | 3 072 | 6 | 384 | 6 | 48 | 6 | 6 |
| 7 | 229 376 | 7 | 28 672 | 7 | 3 584 | 7 | 458 | 7 | 56 | 7 | 7 |

## 2.3. PŘEVOD MEZI ČÍSLY V ŠESTNÁCTKOVÉ A DESÍTKOVÉ SOUSTAVĚ

Postup převodu je stejný jako u převodu mezi čísly v osmičkové a desítkové soustavě.

| Šestnáctkové sloupce | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | | 3 | | 2 | | 1 | |
| HEX = DEK | | HEX = DEK | | HEX = DEK | | HEX = DEK | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4 096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8 192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12 288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16 384 | 4 | 1 024 | 4 | 64 | 4 | 4 |
| 5 | 20 480 | 5 | 1 280 | 5 | 80 | 5 | 5 |
| 6 | 24 576 | 6 | 1 536 | 6 | 96 | 6 | 6 |
| 7 | 28 672 | 7 | 1 792 | 7 | 112 | 7 | 7 |
| 8 | 32 768 | 8 | 2 048 | 8 | 128 | 8 | 8 |
| 9 | 36 864 | 9 | 2 304 | 9 | 144 | 9 | 9 |
| A | 40 960 | A | 2 560 | A | 160 | A | 10 |
| B | 45 056 | B | 2 816 | B | 176 | B | 11 |
| C | 49 152 | C | 3 072 | C | 192 | C | 12 |
| D | 53 284 | D | 3 328 | D | 208 | D | 13 |
| E | 57 344 | E | 3 584 | E | 224 | E | 14 |
| F | 61 440 | F | 3 840 | F | 240 | F | 15 |

# Dodatek 3.

## OPERAČNÍ KÓD POČÍTAČE ADT 4000

*Zkratka* *Význam (anglicky a česky)*

Operace s registry

NOP No operation

A/BLS Prázdná instrukce
AB Left shift

SLA/B Skip if least bit A/B = 0
Přeskok při nenulovém nejnižším bitu registru A nebo B
Aritmetický posun A nebo B o jeden bit vlevo

A/BRS A/B right shift
Aritmetický posun A nebo B o jeden bit vpravo

RA/BL Rotate A/B left
Rotace registru A nebo B o jeden bit vlevo

RA/BR Rotate A/B right
Rotace registru A nebo B o jeden bit vpravo

A/BLR A/B left shift reset sign
Posun A nebo B vlevo o jeden bit, znaménkový bit se vynuluje

ERA/B Rotate E right with A/B
Rotace E vpravo o jeden bit s A nebo B

ELA/B Rotate E left with A/B
Rotace E vlevo o jeden bit s A nebo B

A/BLF Rotace A/B left four bits
Rotace A nebo B vlevo o čtyři bity

Skupina: změny a přeskoky (alter/skip group – ASG)

CLA/B Clear A/B
Nulování registru A nebo B

CMA/B Complement A/B
Inverze registru A nebo B

CCA/B Clear complement A/B
Vynulování a inverze, tedy nastaví se 177777B

CLE Clear E
Vynulování registru E

CME Complement E
Inverze registru E

CCE Clear complement E
Vynulování a inverze, tedy E se nastaví do 1



SEZ Skip if E is zero
Přeskok při nulovém obsahu E

SSA/B Skip if sign of A/B = 0
Přeskok při kladném obsahu registru A/B

SLA/B Skip if least bit A/B = 0
Přeskok při nulovém nejnižším bitu A/B

INA/B Increment A/B by one
Přírůstek registru A/B o jedničku

SZA/B Skip if A/B is zero
Přeskok při nulovém registru A/B

RSS Reverse skip sense
Negace (obrácení) podmínky pro přeskok
(samostatně uvedeno – přeskok vždy)

Skupina: registr OV (overflow)

STO Set overflow bit
Nastavení bitu přeplnění

CLO Clear overflow bit
Vynulování bitu přeplnění

SOC Skip if overflow bit is clear
Přeskok při nulovém bitu přeplnění

SOS Skip if overflow bit is set
Přeskok při nastaveném bitu přeplnění

Vstupní a výstupní operace (Input/output group)

HLT Halt program
Zastavení provádění programu – stop

STF Set flag bit
Nastavení příznaku ukončení operace V/V na adresované jednotce

CLF Clear flag bit
Vynulování příznaku ukončení operace V/V na adresované jednotce

SFC Skip if flag is clear
Přeskok při nedokončené operaci V/V na adresované jednotce

SFS Skip if flag is set
Přeskok při dokončené operaci V/V na adresované jednotce

MIA/B Merge input into A/B
Logický součet vstupní/výstupní vyrovnávací paměti s A/B, výsledek do registru A/B

LIA/B Load input into A/B
Vložení obsahu vstupní/výstupní vyrovnávací paměti do registru A nebo B

OTA/B Output A/B
Vložení obsahu registru A nebo B do vstupní/výstupní vyrovnávací paměti

STC Set control bit
Start operace V/V na adresované jednotce

CLC Clear control bit
Zastavení operace V/V na adresované jednotce

Instrukce rozšířené aritmetiky (EAG)

BA je dvojice 16bitových registrů B a A
Y je 16bitová adresa, která následuje za operačním znakem instrukce
X je počet posuvů (rotací), vyjádřený 4 bity operačního znaku

MPY Y Multiply
Násobení
v registru A je násobenec, Y udává adresu násobitele, součin je v BA, celočíselný s dvojnásobnou délkou

DIV Y Divide
Dělení
v BA je dělenec, Y udává adresu dělitele, v A je pak podíl, v B je zbytek, příznak OV indikuje dělení nulou nebo že podíl přesáhl bit 15

DLD Y Double load
Dvojí vložení
1. slovo se vloží do A, 2. slovo se vloží do B, Y udává adresu prvního čteného slova

DST Y Double store
Dvojí uložení
1. slovo se bere z A, 2. slovo se bere z B, Y udává adresu, kam se uloží AB

ASR X Arithmetic shift right
Posun BA vpravo
Bit (B15) zůstává beze změny

ASL X Arithmetic shift left
Posun BA vlevo
je-li (B15) = (B14), nastaví se příznak OV, potom se provede posun, (B14) se ztrácí

LSR X Logical shift right
Logický posun BA vpravo
Zanedbá se znaménko (bity zleva se plní nulami), bity vystupující zprava ... (A1) (A0), se ztrácejí

LSL X Logical shift left
Logický posun BA vlevo
Zanedbá se znaménko, bit (B15) se ztrácí

RRR X Rotate right
Rotace BA vpravo

RRL X Rotace left
Rotace BA vlevo

Bity 0 až 3 operačního znaku určují počet posunů (X). Hodnota X leží v rozsahu 0 až 15. Nula znamená 16 posunů (rotací)



Kódování instrukcí počítače ADT 4000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D/I | AND | 001 | | 0 | Z/C | D/E | *LS | 000 | CLE | D/E | SL* | *LS | | | 000 |
| D/I | XOR | 010 | | 0 | Z/C | D/E | *RS | 001 | | D/E | | *RS | | | 001 |
| D/I | IOR | 011 | | 0 | Z/C | D/E | R*L | 010 | | D/E | | R*L | | | 010 |
| D/I | JSB | 001 | | 1 | Z/C | D/E | R*R | 011 | | D/E | | R*R | | | 011 |
| D/I | JMP | 010 | | 1 | Z/C | D/E | *LR | 100 | | D/E | | *LF | | | 100 |
| D/I | ISZ | 011 | | 1 | Z/C | D/E | ER* | 101 | | D/E | | ER* | | | 101 |
| D/I | AD* | 100 | | A/B | Z/C | D/E | AL* | 110 | | D/E | | EL* | | | 110 |
| D/I | CP* | 101 | | A/B | Z/C | D/E | *LF | 111 | | D/E | | *LF | | | 111 |
| D/I | LD* | 110 | | A/B | Z/C | | | | | | | | | | |
| D/I | ST* | 111 | | A/B | Z/C | | | | | | | | | | |
| 0 | SQR | 000 | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | | | | A/B | 0 | D/E | | | | | | | | | |
| | NOP | | | 0 | 0 | 0 | | 000 | 0 | 0 | 0 | | | | 000 |
| 0 | ASG | 000 | | A/B | 1 | CL*01 | CLE 01 | | SEZ | SS* | | IN* | | SZ* | RSS |
| | | | | A/B | 1 | CM*10 | CME 10 | | | | *SL | | | | |
| | | | | A/B | 1 | CC*11 | CCE 11 | | | | | | | | |
| 1 | IOG | 000 | | | 1 | H/C | HLT 001 | | vstupní/výstupní adresa | | | | | | |
| | | | | | 1 | 0 | STF 001 | | | | | | | | |
| | | | | | 1 | 1 | CLF 001 | | | | | | | | |
| | | | | | 1 | 0 | SFC 010 | | | | | | | | |
| | | | | | 1 | 0 | SFS 011 | | | | | | | | |
| | | | | A/D | 1 | H/C | MI*100 | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | A/B | 1 | H/C | LI* 101 | | | | | | | | |
| | | | | 0 | 1 | H/C | OT* 110 | | | | | | | | |
| | | | | A/B | 1 | H/C | STC 111 | | | | | | | | |
| | | | | 1 | 1 | H/C | CLC 111 | | | | | | | | |
| | | | | | 1 | 0 | STO 001 | | 000 | 001 | | | | | |
| | | | | | 1 | I | CLO 001 | | 000 | 001 | | | | | |
| | | | | | 1 | H/C | SOC 010 | | 000 | 001 | | | | | |
| 1 | EAG | 000 | | MPY | 000 | ** | 010 | | 000 | 000 | | | | | |
| | | | | DIV | 000 | ** | 100 | | 000 | 000 | | | | | |
| | | | | DLD | 100 | ** | 010 | | 000 | 000 | | | | | |
| | | | | DST | 100 | ** | 100 | | 000 | 000 | | | | | |
| | | | | ASR | 001 | | 000 | | 0 1 | | | | počet bitů | | |
| | | | | ASL | 000 | | 000 | | 0 1 | | | | | | |
| | | | | LSR | 001 | | 000 | | 1 0 | | | | | | |
| | | | | LSL | 000 | | 000 | | 1 0 | | | | | | |
| | | | | RRR | 001 | | 001 | | 0 0 | | | | | | |
| | | | | RRL | 00C | | 001 | | 0 0 | | | | | | |
| 1 | FLP | 000 | | | 101 | 0 | C FAD** | | 000 | 0 | 000 | | | | |
| | | | | | | | FSB** | 001 | | | | | | | |
| | | | | | | | FMP** | 010 | | | | | | | |
| | | | | | | | FDY** | 011 | | | | | | | |
| | | | | | | | FIX | 100 | | | | | | | |
| | | | | | | | FLT | 101 | | | | | | | |

*Poznámka:*

* – znamená registr A
D/I – DIRECT/INDIRECT – přímá/nepřímá adresa
Z/C – ZERO/CURRENT – stránka nula/běžná stránka
D/E – DISABLE/ENABLE – určuje platnost následující trojice bitů
H/C – HOLD/CLEAR – ponechává/nuluje příznak dokončení/zahájení vstupní/výstupní operace



# Dodatek 4.

## OPERAČNÍ KÓD MIKROPROCESORU 8080

a) Číselné řazení

| Kód | Symbol | | Délka |
|---|---|---|---|
| 00 | NOP | | 1 |
| 01 | LXI | B, w | 3 |
| 02 | STAX | B | 1 |
| 03 | INX | B | 1 |
| 04 | INR | B | 1 |
| 05 | DCR | B | 1 |
| 06 | MVI | B, D | 2 |
| 07 | RLC | | 1 |
| 08 | | | |
| 09 | DAD | B | 1 |
| 0A | LDAX | B | 1 |
| 0B | DCX | B | 1 |
| 0C | INR | C | 1 |
| 0D | DCR | C | 1 |
| 0E | MVI | C, d | 2 |
| 0F | RRC | | 1 |
| 10 | | | |
| 11 | LXI | D, w | 3 |
| 12 | STAX | D | 1 |
| 13 | INX | D | 1 |
| 14 | INR | D | 1 |
| 15 | DCR | D | 1 |
| 16 | MVI | D, d | 2 |
| 17 | RAL | | 1 |
| 18 | | | |
| 19 | DAD | D | 1 |
| 1A | LDAX | D | 1 |
| 1B | DCX | D | 1 |
| 1C | INR | E | 1 |
| 1D | DCR | E | 1 |
| 1E | MVI | E, d | 2 |

| Kód | Symbol | | Délka |
|---|---|---|---|
| 1F | RAR | | 1 |
| 20 | | | 1 |
| 21 | LXI | H, w | 3 |
| 22 | SHLD | a | 3 |
| 23 | INX | H | 1 |
| 24 | INR | H | 1 |
| 25 | DCR | H | 1 |
| 26 | MVI | H, d | 2 |
| 27 | DAA | | 1 |
| 28 | | | |
| 29 | DAD | H | 1 |
| 2A | LHLD | a | 3 |
| 2B | DCX | H | 1 |
| 2C | INR | L | 1 |
| 2D | DCR | L | 1 |
| 2E | MVI | L, d | 2 |
| 2F | CMA | | 1 |
| 30 | | | 1 |
| 31 | LXI | SP, w | 3 |
| 32 | STA | a | 3 |
| 33 | INX | SP | 1 |
| 34 | INR | M | 1 |
| 35 | DCR | M | 1 |
| 36 | MVI | M, d | 2 |
| 37 | STC | | 1 |
| 38 | | | |
| 39 | DAD | SP | 1 |
| 3A | LDA | a | 3 |
| 3B | DCX | SP | 1 |
| 3C | INR | A | 1 |
| 3D | DCR | A | 1 |

| Kód | Symbol | | Délka |
|---|---|---|---|
| 3E | MVI | A, d | 2 |
| 3F | CMC | | 1 |
| 40 | MOV | B, B | 1 |
| 41 | MOV | B, C | 1 |
| 42 | MOV | B, D | 1 |
| 43 | MOV | B, E | 1 |
| 44 | MOV | B, H | 1 |
| 45 | MOV | B, L | 1 |
| 46 | MOV | B, M | 1 |
| 47 | MOV | B, A | 1 |
| 48 | MOV | C, B | 1 |
| 49 | MOV | C, C | 1 |
| 4A | MOV | C, D | 1 |
| 4B | MOV | C, E | 1 |
| 4C | MOV | C, H | 1 |
| 4D | MOV | C, L | 1 |
| 4E | MOV | C, M | 1 |
| 4F | MOV | C, A | 1 |
| 50 | MOV | D, B | 1 |
| 51 | MOV | D, C | 1 |
| 52 | MOV | D, D | 1 |
| 53 | MOV | D, E | 1 |
| 54 | MOV | D, H | 1 |
| 55 | MOV | D, L | 1 |
| 56 | MOV | D, M | 1 |
| 57 | MOV | D, A | 1 |
| 58 | MOV | E, B | 1 |
| 59 | MOV | E, C | 1 |
| 5A | MOV | E, D | 1 |
| 5B | MOV | E, E | 1 |
| 5C | MOV | E, H | 1 |
| 5D | MOV | E, L | 1 |
| 5E | MOV | E, M | 1 |
| 5F | MOV | E, A | 1 |
| 60 | MOV | H, B | 1 |
| 61 | MOV | H, C | 1 |
| 62 | MOV | H, D | 1 |
| 63 | MOV | H, E | 1 |
| 64 | MOV | H, H | 1 |
| 65 | MOV | H, L | 1 |
| 66 | MOV | H, M | 1 |
| 67 | MOV | H, A | 1 |

| Kód | Symbol | | Délka |
|---|---|---|---|
| 68 | MOV | L, B | 1 |
| 69 | MOV | L, C | 1 |
| 6A | MOV | L, D | 1 |
| 6B | MOV | L, E | 1 |
| 6C | MOV | L, H | 1 |
| 6D | MOV | L, L | 1 |
| 6E | MOV | L, M | 1 |
| 6F | MOV | L, A | 1 |
| 70 | MOV | M, B | 1 |
| 71 | MOV | M, C | 1 |
| 72 | MOV | M, D | 1 |
| 73 | MOV | M, E | 1 |
| 74 | MOV | M, H | 1 |
| 75 | MOV | M, L | 1 |
| 76 | HLT | | 1 |
| 77 | MOV | M, A | 1 |
| 78 | MOV | A, B | 1 |
| 79 | MOV | A, C | 1 |
| 7A | MOV | A, D | 1 |
| 7B | MOV | A, E | 1 |
| 7C | MOV | A, H | 1 |
| 7D | MOV | A, L | 1 |
| 7E | MOV | A, M | 1 |
| 7F | MOV | A, A | 1 |
| 80 | ADD | B | 1 |
| 81 | ADD | C | 1 |
| 82 | ADD | D | 1 |
| 83 | ADD | E | 1 |
| 84 | ADD | H | 1 |
| 85 | ADD | L | 1 |
| 86 | ADD | M | 1 |
| 87 | ADD | A | 1 |
| 88 | ADC | B | 1 |
| 89 | ADC | C | 1 |
| 8A | ADC | D | 1 |
| 8B | ADC | E | 1 |
| 8C | ADC | H | 1 |
| 8D | ADC | L | 1 |
| 8E | ADC | M | 1 |
| 8F | ADC | A | 1 |
| 90 | SUB | B | 1 |
| 91 | SUB | C | 1 |



| Kód | Symbol | | Délka |
|---|---|---|---|
| 92 | SUB | D | 1 |
| 93 | SUB | E | 1 |
| 94 | SUB | H | 1 |
| 95 | SUB | L | 1 |
| 96 | SUB | M | 1 |
| 97 | SUB | A | 1 |
| 98 | SBB | B | 1 |
| 99 | SBB | C | 1 |
| 9A | SBB | D | 1 |
| 9B | SBB | E | 1 |
| 9C | SBB | H | 1 |
| 9D | SBB | L | 1 |
| 9E | SBB | M | 1 |
| 9F | SBB | A | 1 |
| A0 | ANA | B | 1 |
| A1 | ANA | C | 1 |
| A2 | ANA | D | 1 |
| A3 | ANA | E | 1 |
| A4 | ANA | H | 1 |
| A5 | ANA | L | 1 |
| A6 | ANA | M | 1 |
| A7 | ANA | A | 1 |
| A8 | XRA | B | 1 |
| A9 | XRA | C | 1 |
| AA | XRA | D | 1 |
| AB | XRA | E | 1 |
| AC | XRA | H | 1 |
| AD | XRA | L | 1 |
| AE | XRA | M | 1 |
| AF | XRA | A | 1 |
| B0 | ORA | B | 1 |
| B1 | ORA | C | 1 |
| B2 | ORA | D | 1 |
| B3 | ORA | E | 1 |
| B4 | ORA | H | 1 |
| B5 | ORA | L | 1 |
| B6 | ORA | M | 1 |
| B7 | ORA | A | 1 |
| B8 | CMP | B | 1 |
| B9 | CMP | C | 1 |
| BA | CMP | D | 1 |
| BB | CMP | E | 1 |

| Kód | Symbol | | Délka |
|---|---|---|---|
| BC | CMP | H | 1 |
| BD | CMP | L | 1 |
| BE | CMP | M | 1 |
| BF | CMP | A | 1 |
| C0 | RNZ | | 1 |
| C1 | POP | B | 1 |
| C2 | JNZ | a | 3 |
| C3 | JMP | a | 3 |
| C4 | CNZ | a | 3 |
| C5 | PUSH | B | 1 |
| C6 | ADI | d | 2 |
| C7 | RST | Ø | 1 |
| C8 | RZ | | 1 |
| C9 | RET | | 1 |
| CA | JZ | a | 3 |
| CB | | | |
| CC | CZ | a | 3 |
| CD | CALL | a | 3 |
| CE | ACI | d | 2 |
| CF | RST | 1 | 1 |
| D0 | RNC | | 1 |
| D1 | POP | D | 1 |
| D2 | JNC | a | 3 |
| D3 | OUT | d | 2 |
| D4 | CNC | a | 3 |
| D5 | PUSH | D | 1 |
| D6 | SUI | d | 2 |
| D7 | RST | 2 | 1 |
| D8 | RC | | 1 |
| D9 | | | |
| DA | JC | a | 3 |
| DB | IN | d | 2 |
| DC | CC | a | 3 |
| DD | | | |
| DE | SBI | d | 2 |
| DF | RST | 3 | 1 |
| E0 | RPO | | 1 |
| E1 | POP | H | 1 |
| E2 | JPO | a | 3 |
| E3 | XTHL | | 1 |
| E4 | CPO | a | 3 |
| E5 | PUSH | H | 1 |

| Kód | Symbol | | Délka | Kód | Symbol | | Délka |
|---|---|---|---|---|---|---|---|
| E6 | ANI | d | 2 | F3 | DI | | 1 |
| E7 | RST | 4 | 1 | F4 | CP | a | 3 |
| E8 | RPE | | 1 | F5 | PUSH | PSW | 1 |
| E9 | PCHL | | 1 | F6 | ORI | d | 2 |
| EA | JPE | a | 3 | F7 | RST | 6 | 1 |
| EB | XCHG | | 1 | F8 | RM | | 1 |
| EC | CPE | a | 3 | F9 | SPHL | | 1 |
| ED | | | | FA | JM | a | 3 |
| EE | XRI | d | 2 | FB | EI | | 1 |
| EF | RST | 5 | 1 | FC | CM | a | 3 |
| F0 | RP | | 1 | FD | | | |
| F1 | POP | PSW | 1 | FE | CPI | d | 2 |
| F2 | JP | a | 3 | FF | RST | 7 | 1 |

b) abecední řazení

| Symbol | | Kód | Délka | Takty | Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|---|---|---|---|---|
| ACI | d | CE | 2 | 7 | ANA | E | A3 | 1 | 4 |
| ADC | A | 8F | 1 | 4 | ANA | H | A4 | 1 | 4 |
| ADC | B | 88 | 1 | 4 | ANA | L | A5 | 1 | 4 |
| ADC | C | 89 | 1 | 4 | ANA | M | A6 | 1 | 7 |
| ADC | D | 8A | 1 | 4 | ANI | d | E6 | 2 | 7 |
| ADC | E | 8B | 1 | 4 | CALL | a | CD | 3 | 17 |
| ADC | H | 8C | 1 | 4 | CC | a | DC | 3 | 11/17 |
| ADC | L | 8D | 1 | 4 | CM | a | FC | 3 | 11/17 |
| ADC | M | 8E | 1 | 7 | CMA | | 2F | 1 | 4 |
| ADD | A | 87 | 1 | 4 | CMC | | 3F | 1 | 4 |
| ADD | B | 80 | 1 | 4 | CMP | A | BF | 1 | 4 |
| ADD | C | 81 | 1 | 4 | CMP | B | B8 | 1 | 4 |
| ADD | D | 82 | 1 | 4 | CMP | C | B9 | 1 | 4 |
| ADD | E | 83 | 1 | 4 | CMP | D | BA | 1 | 4 |
| ADD | H | 84 | 1 | 4 | CMP | E | BB | 1 | 4 |
| ADD | L | 85 | 1 | 4 | CMP | H | BC | 1 | 4 |
| ADD | M | 86 | 1 | 7 | CMP | L | BD | 1 | 4 |
| ADI | d | C6 | 2 | 7 | CMP | M | BE | 1 | 7 |
| ANA | A | A7 | 1 | 4 | CNC | a | D4 | 3 | 11/17 |
| ANA | B | A0 | 1 | 4 | CNZ | a | C4 | 3 | 11/17 |
| ANA | C | A1 | 1 | 4 | CP | a | F4 | 3 | 11/17 |
| ANA | D | A2 | 1 | 4 | CPE | a | EC | 3 | 11/17 |



| Symbol | | Kód | Délka | Takty | Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|---|---|---|---|---|
| CPI | d | FE | 2 | 7 | JPE | a | EA | 3 | 10 |
| CPO | a | E4 | 3 | 11/17 | JPO | a | E2 | 3 | 10 |
| CZ | a | CC | 3 | 11/17 | JZ | a | CA | 3 | 10 |
| DAA | | 27 | 1 | 4 | LDA | a | 3A | 3 | 13 |
| DAD | B | 09 | 1 | 10 | LDAX | B | 0A | 1 | 7 |
| DAD | D | 19 | 1 | 10 | LDAX | D | 1A | 1 | 7 |
| DAD | H | 29 | 1 | 10 | LHLD | a | 2A | 3 | 16 |
| DAD | SP | 39 | 1 | 10 | LXI | B, w | 01 | 3 | 10 |
| DCR | A | 3D | 1 | 5 | LXI | D, w | 11 | 3 | 10 |
| DCR | B | 05 | 1 | 5 | LXI | H, w | 21 | 3 | 10 |
| DCR | C | 0D | 1 | 5 | LXI | SP, w | 31 | 3 | 10 |
| DCR | D | 15 | 1 | 5 | MOV | A, A | 7F | 1 | 5 |
| DCR | E | 1D | 1 | 5 | MOV | A, B | 78 | 1 | 5 |
| DCR | H | 25 | 1 | 5 | MOV | A, C | 79 | 1 | 5 |
| DCR | L | 2D | 1 | 5 | MOV | A, D | 7A | 1 | 5 |
| DCR | M | 35 | 1 | 10 | MOV | A, E | 7B | 1 | 5 |
| DCX | B | 0B | 1 | 5 | MOV | A, H | 7C | 1 | 5 |
| DCX | D | 1B | 1 | 5 | MOV | A, L | 7D | 1 | 5 |
| DCX | H | 2B | 1 | 5 | MOV | A, M | 7E | 1 | 7 |
| DCX | SP | 3B | 1 | 5 | MOV | B, A | 47 | 1 | 5 |
| DI | | F3 | 1 | 4 | MOV | B, B | 40 | 1 | 5 |
| EI | | FB | 1 | 4 | MOV | B, C | 41 | 1 | 5 |
| HLT | | 76 | 1 | 7 | MOV | B, D | 42 | 1 | 5 |
| IN | d | DB | 2 | 10 | MOV | B, E | 43 | 1 | 5 |
| INR | A | 3C | 1 | 5 | MOV | B, H | 44 | 1 | 5 |
| INR | B | 04 | 1 | 5 | MOV | B, L | 45 | 1 | 5 |
| INR | C | 0C | 1 | 5 | MOV | B, M | 46 | 1 | 7 |
| INR | D | 14 | 1 | 5 | MOV | C, A | 4F | 1 | 5 |
| INR | E | 1C | 1 | 5 | MOV | C, B | 48 | 1 | 5 |
| INR | H | 24 | 1 | 5 | MOV | C, C | 49 | 1 | 5 |
| INR | L | 2C | 1 | 5 | MOV | C, D | 4A | 1 | 5 |
| INR | M | 34 | 1 | 10 | MOV | C, E | 4B | 1 | 5 |
| INX | B | 03 | 1 | 5 | MOV | C, H | 4C | 1 | 5 |
| INX | D | 13 | 1 | 5 | MOV | C, L | 4D | 1 | 5 |
| INX | H | 23 | 1 | 5 | MOV | C, M | 4E | 1 | 7 |
| INX | SP | 33 | 1 | 5 | MOV | D, A | 57 | 1 | 5 |
| JC | a | DA | 3 | 10 | MOV | D, B | 50 | 1 | 5 |
| JM | a | FA | 3 | 10 | MOV | D, C | 51 | 1 | 5 |
| JMP | a | C3 | 3 | 10 | MOV | D, D | 52 | 1 | 5 |
| JNC | a | D2 | 3 | 10 | MOV | D, E | 53 | 1 | 5 |
| JNZ | a | C2 | 3 | 10 | MOV | D, H | 54 | 1 | 5 |
| JP | a | F2 | 3 | 10 | MOV | D, L | 55 | 1 | 5 |

| Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|
| MOV | D, M | 56 | 1 | 7 |
| MOV | E, A | 5F | 1 | 5 |
| MOV | E, B | 58 | 1 | 5 |
| MOV | E, C | 59 | 1 | 5 |
| MOV | E, D | 5A | 1 | 5 |
| MOV | E, E | 5B | 1 | 5 |
| MOV | E, H | 5C | 1 | 5 |
| MOV | E, L | 5D | 1 | 5 |
| MOV | E, M | 5E | 1 | 7 |
| MOV | H, A | 67 | 1 | 5 |
| MOV | H, B | 60 | 1 | 5 |
| MOV | H, C | 61 | 1 | 5 |
| MOV | H, D | 62 | 1 | 5 |
| MOV | H, E | 63 | 1 | 5 |
| MOV | H, H | 64 | 1 | 5 |
| MOV | H, L | 65 | 1 | 5 |
| MOV | H, M | 66 | 1 | 7 |
| MOV | L, A | 6F | 1 | 5 |
| MOV | L, B | 68 | 1 | 5 |
| MOV | L, C | 69 | 1 | 5 |
| MOV | L, D | 6A | 1 | 5 |
| MOV | L, E | 6B | 1 | 5 |
| MOV | L, H | 6C | 1 | 5 |
| MOV | L, L | 6D | 1 | 5 |
| MOV | L, M | 6E | 1 | 7 |
| MOV | M, A | 77 | 1 | 7 |
| MOV | M, B | 70 | 1 | 7 |
| MOV | M, C | 71 | 1 | 7 |
| MOV | M, D | 72 | 1 | 7 |
| MOV | M, E | 73 | 1 | 7 |
| MOV | M, H | 74 | 1 | 7 |
| MOV | M, L | 75 | 1 | 7 |
| MVI | A, d | 3E | 2 | 7 |
| MVI | B, d | 06 | 2 | 7 |
| MVI | C, d | 0E | 2 | 7 |
| MVI | D, d | 16 | 2 | 7 |
| MVI | E, d | 1E | 2 | 7 |
| MVI | H, d | 26 | 2 | 7 |
| MVI | L, d | 2E | 2 | 7 |
| MVI | M, d | 36 | 2 | 10 |
| NOP | | 00 | 1 | 4 |
| ORA | A | B7 | 1 | 4 |

| Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|
| ORA | B | B0 | 1 | 4 |
| ORA | C | B1 | 1 | 4 |
| ORA | D | B2 | 1 | 4 |
| ORA | E | B3 | 1 | 4 |
| ORA | H | B4 | 1 | 4 |
| ORA | L | B5 | 1 | 4 |
| ORA | M | B6 | 1 | 7 |
| ORI | d | F6 | 2 | 7 |
| OUT | d | D3 | 2 | 10 |
| PCHL | | E9 | 1 | 5 |
| POP | B | C1 | 1 | 10 |
| POP | D | D1 | 1 | 10 |
| POP | H | E1 | 1 | 10 |
| POP | PSW | F1 | 1 | 10 |
| PUSH | B | C5 | 1 | 11 |
| PUSH | D | D5 | 1 | 11 |
| PUSH | H | E5 | 1 | 11 |
| PUSH | PSW | F5 | 1 | 11 |
| RAL | | 17 | 1 | 4 |
| RAR | | 1F | 1 | 4 |
| RC | | D8 | 1 | 5/11 |
| RET | | C9 | 1 | 10 |
| RLC | | 07 | 1 | 4 |
| RM | | F8 | 1 | 5/11 |
| RNC | | D0 | 1 | 5/11 |
| RNZ | | C0 | 1 | 5/11 |
| RP | | F0 | 1 | 5/11 |
| RPE | | E8 | 1 | 5/11 |
| RPO | | E0 | 1 | 5/11 |
| RRC | | 0F | 1 | 4 |
| RST | Ø | C7 | 1 | 11 |
| RST | 1 | CF | 1 | 11 |
| RST | 2 | D7 | 1 | 11 |
| RST | 3 | DF | 1 | 11 |
| RST | 4 | E7 | 1 | 11 |
| RST | 5 | EF | 1 | 11 |
| RST | 6 | F7 | 1 | 11 |
| RST | 7 | FF | 1 | 11 |
| RZ | | C8 | 1 | 5/11 |
| SBB | A | 9F | 1 | 4 |
| SBB | B | 98 | 1 | 4 |
| SBB | C | 99 | 1 | 4 |



| Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|
| SBB | D | 9A | 1 | 4 |
| SBB | E | 9B | 1 | 4 |
| SBB | H | 9C | 1 | 4 |
| SBB | L | 9D | 1 | 4 |
| SBB | M | 9E | 1 | 7 |
| SBI | d | DE | 2 | 7 |
| SHLD | a | 22 | 3 | 16 |
| SPHL | | F9 | 1 | 5 |
| STA | a | 32 | 3 | 13 |
| STAX | B | 02 | 1 | 7 |
| STAX | D | 12 | 1 | 7 |
| STC | | 37 | 1 | 4 |
| SUB | A | 97 | 1 | 4 |
| SUB | B | 90 | 1 | 4 |
| SUB | C | 91 | 1 | 4 |
| SUB | D | 92 | 1 | 4 |

| Symbol | | Kód | Délka | Takty |
|---|---|---|---|---|
| SUB | E | 93 | 1 | 4 |
| SUB | H | 94 | 1 | 4 |
| SUB | L | 95 | 1 | 4 |
| SUB | M | 96 | 1 | 7 |
| SUI | d | D6 | 2 | 7 |
| XCHG | | EB | 1 | 4 |
| XRA | A | AF | 1 | 4 |
| XRA | B | A8 | 1 | 4 |
| XRA | C | A9 | 1 | 4 |
| XRA | D | AA | 1 | 4 |
| XRA | E | AB | 1 | 4 |
| XRA | H | AC | 1 | 4 |
| XRA | L | AD | 1 | 4 |
| XRA | M | AE | 1 | 7 |
| XRI | d | EE | 2 | 7 |
| XTHL | | E3 | 1 | 18 |

# Literatura


[1] Component data Catalog 78 – INTEL

[2] *Dědina, B. – Valášek, P.:* Mikroprocesory a mikropočítače. Praha, SNTL 1983.

[3] *Drápal, V.:* Technický popis minipočítače ADT 4000/4100. ZPA Trutnov 1976.

[4] *Dvořák, V. a kol.:* Sbírka řešených úloh z automatizační techniky. TESLA ELTOS, IMA Praha 1983.

[5] *Hladík, D. a kol.:* Obvodový emulátor 8080. Uživatelská příručka ČSVTS, Bratislava 1984.

[6] *Hlavička, J. a kol.:* Diagnostika elektronických číslicových obvodů. Praha, SNTL 1982.

[7] *Janovský:* Mikroprocesorové vývojové soupravy

[8] Katalog součástek TESLA 1983–1984, 1. díl. Praha, TESLA Eltos.

[9] *Kottek, E.:* Testy integrovaných obvodů řady MH74 a MH743 vycházející ze struktury. Sdělovací technika 4 (1976).

[10] *Kučera, J.:* Diagnostický software ADT 4400, ZPA Trutnov 1980.

[11] Návod pro použití, instalaci a obsluhu ADT. ZPA Trutnov 1975.

[12] *Novák, S.:* Simulační program SIM 80/85. Příloha časopisu Amatérské radio č. 10 1983 až č. 7 1984.

[13] Popis školního mikropočítače TEMS 8003, TESLA Vráble 1978.

[14] *Starý, J.:* Mikropočítač a jeho programování. Praha, SNTL 1984.




ING. PAVEL PŘÍVĚTIVÝ A KOLEKTIV

## Provoz a údržba počítačů


DT 681.32(075.3)

Vydalo SNTL – Nakladatelství technické literatury, n. p., Spálená 51, 113 02 Praha 1, v roce 1989 jako svou 10 869. publikaci. Redakce elektrotechnické literatury. Odpovědný redaktor Ing. Milan Veselý. Vazbu navrhl Metoděj Sychra. Grafická úprava a technická redakce Dana Břízová. Ze sazby monofoto vytiskl ofsetem Tisk, knižní výroba, n. p., Brno, závod 1. 256 stran, 51 obrázků, 17 tabulek. Typové číslo L25-C2-IV-31f/55775. Vydání první. Náklad 4000 výtisků. 12,04 AA, 12,33 VA

05/40
Cena vázaného výtisku Kčs 16,–

507/23,856

Učebnice je určena pro předmět Provoz a údržba počítačů vyučovaný na středních průmyslových školách s výukou studijního oboru 26-60-6 Elektronická a sdělovací zařízení, AB: Elektronické počítačové systémy

441 2007

04-538-89 Kčs 16,–