ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Mikroprocesorová technika I., II.

Postgraduální studium

Ing. Jiří Chod, CSc.

1987

Ediční středisko ČVUT, Praha 1, Husova 5

© Jiří Chod, 1987

1. MIKROPOČÍTAČE

Číslicové počítače prošly za dobu svého trvání bouřlivým a složitým vývojem. Zvýšila se jejich výkonnost, snížil příkon a rozměry.

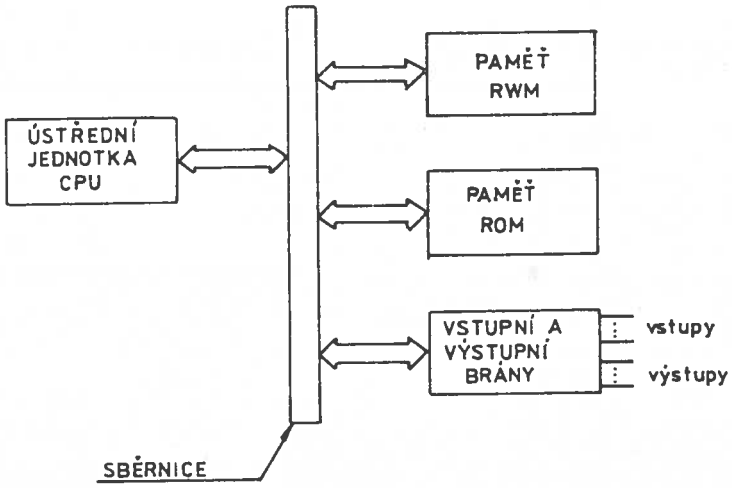
S použitím zkušeností získaných při vývoji obvodů velké integrace (LSI) byly vytvořeny nové prvky - mikroprocesory. Ty pak umožňují sestrojit rozměrově malé, na příkon nenáročné a levné počítače, které představují kvalitativní skok v možnostech aplikace a hromadného rozšíření číslicové techniky.

I když pokusy o vytvoření počítacího stroje byly již v minulých staletích, lze počátek prvních počítačů datovat až první polovinou čtyřicátých let, kdy vznikl první počítač (typ Z1 - Konrád Zuse - Německo). Vývoj počítačů poté pokračoval rychlým tempem, zejména v padesátých letech (fy.IBM). V roce 1971 se objevil nový prvek - mikroprocesor 4004 firmy Intel - USA. V roce 1974 se objevily mikroprocesory druhé generace, jejichž typickým představitelem je mikroprocesor 8080A. Tento procesor doznal značného rozšíření a v současné době je vyráběn i v ČSSR. Z mikroprocesorů druhé generace vyšli při svém návrhu i konstruktéři mikroprocesorů třetí generace (8085 - Intel, Z80 - Zilog).Zejména typ Z80 a jeho ekvivalenty, v provedení různých technologií, tvoří i dnes základ osmibitových mikropočítačových systémů na nižších úrovních.

Další vývoj mikroprocesorů směřoval jednak k mikroprocesorům s větší délkou slova - 16ti bitové a 32 bitové mikroprocesory, jednak k mikroprocesorům specializovaným na určitou činnost. V současné době řada světových výrobců vyrábí několik set různých typů mikroprocesorů, stovky typů pamětí a podpůrných obvodů. Přes uvedené výsledky dosažené při vývoji počítačů a mikropočítačů stojíme teprve na počátku jejich rozvoje a lze předpokládat, že aplikace výpočetní techniky zasáhnou podstatným způsobem do všech oborů lidské činnosti.

1.1 Struktura mikropočítače

Obvody typického mikropočítače můžeme rozčlenit do bloků dle obr.1.1.



Obr.1.1.: Základní blokové schéma mikropočítače

Jednotlivé části mikropočítače jsou seskupeny okolo společné sběrnice.

Mikropočítač sestává z centrální jednotky (Central Processing Unit) - procesoru -řídící činnost celého systému, paměti ROM (Read Only Memory), ve které jsou uloženy instrukce zajišťující realizaci daného algoritmu řízení a případně konstanty a tabulky potřebné pro činnost programu. Sestava dále obsahuje paměť RWM (Read Write Memory) pro ukládání dat, vstupní brány (Input Port) a výstupní brány (Output Port) pro styk s vnějším prostředím. Rozdělení sběrnice, propojující jednotlivé bloky, bývá obvykle členěno na datovou, adresovou a řídící sběrnici.

Data mikropočítač zpracovává po slabikách (Byte) skládajících se z jednotlivých bitů. Pro námi uvažovaný mikroprocesor 8080A je délka slova 8 bitů - jedna slabika.

Procesor řídí práci mikropočítače podle instrukcí uložených v paměti programu (ROM, RWM), ze které je postupně čte a jednu po druhé vykonává. Jednotlivá data potom ukládá zpět do paměti, nebo je vysílá na výstupní zařízení. Činnost procesoru je přitom řízena hodinovými impulsy. Každá buňka paměti, každá vstupní nebo výstupní, brána mají svoji adresu. Tuto adresu vysílá procesor po adresové sběrnici. Informace (data) mezi paměťovými buňkami nebo V/V bránami se přenáší po datové sběrnici. Signály určující správné časování přenosů po datové sběrnici se přenášejí po řídící sběrnici.

Instrukční soubor mikroprocesoru zahrnuje typicky následující bloky instrukcí:

1. Instrukce pro přesuny dat
2. Aritmetické a logické instrukce
3. Instrukce pro větvení programu (skokové)
4. Instrukce pro komunikaci s V/V branami
5. Instrukce řídicí.

Návrh přístroje řízeného mikropočítačem potom můžeme rozdělit do tří kroků :

1. Obvodové řešení (Hardware) vlastního počítače se strukturou dle obr.1.1.
2. Propojení (Hardware) počítače s ostatními obvody přístroje (klávesnice, zobrazovač, atd.)
3. Programové vybavení (Software) - tedy program - posloupnost instrukcí prováděných počítačem při řízení přístroje.

1.2 Centrální jednotka

Centrální jednotka (procesor) je jádrem mikropočítače. Obvykle je tvořena mikroprocesorovým obvodem a několika dalšími integrovanými obvody. Technologický pokrok vede k umístění centrální jednotky v jednom pouzdře integrovaného obvodu. V dalším se zaměříme již přímo na procesor řady 8080A, ale většina úvah platí obecně.

Procesor obsahuje jednak aritmetickou a logickou jednotku (ALU-Arithmetic Logic Unit), registr instrukcí (Instruction Register) a soubor univerzálních a speciálních registrů. Všechny bloky jsou propojeny vnitřní datovou sběrnicí.

Soubor obsahuje univerzální 8bitové registry (B,C,D,E,H,L), které je možno spojit v případě potřeby do dvojic. V tomto případě mají dvojice registrů symbolické jméno prvního z nich, tj. B,D,H.

Dále obsahuje hlavní osmibitový registr (A-Accumulator)-střadač. Speciální registry jsou registr příznaků F (Flag Register), ukazatel zásobníku SP (Stack Pointer) a čítač instrukcí PC (Program Counter). Registry SP a PC jsou šestnáctibitové, ostatní osmibitové. Typickou strukturu souboru registrů ukazuje obr. 1.2.

PSW	A	Střadač (Accumulator)	(8)	F Registr příznaků (flags)	(8)
B	B		(8)	C	(8)
D	D		(8)	E	(8)
H	H		(8)	L	(8)
	SP	Ukazatel zásobníku (Stack Pointer)			(16)
	PC	Čítač instrukcí (Program Counter)			(16)

INTE (1)	povolení přerušení	paměťové místo (adresované HL)	M (8)

Obr.1.2 Soubor registrů procesoru 8080A

V současné době používají mikroprocesory nejčastěji 8bitové slovo. Pro některé případy však tato délka slova nepostačuje, zejména tehdy, jestliže instrukce obsahuje adresu, nebo operand. V těchto případech je instrukce víceslabiková. První instrukce obsahuje vždy operační část instrukce (též některé odkazy na operandy), další slabiky instrukce obsahují vždy jen odkazy na operandy.

Po zapnutí napájení čte procesor vždy z takového místa paměti ROM (nulová adresa), na kterém je první slabika instrukce. Dekódováním první slabiky procesor zjistí kolika slabiková instrukce bude, je-li víceslabiková, procesor automaticky přečte i další slabiky. Poté procesor instrukci provede a dále čte další slabiku další instrukce z paměti. Čtení přitom provádí z místa následujícího bezprostředně za poslední slabikou předchozí instrukce. Řízení této činnosti zajišťuje (viz obr. 1.2) čítač instrukcí. Při každém přečtení instrukce se čítač instrukcí inkrementuje (zvýší svoji hodnotu o jedničku). Proto postačí ke správnému provedení programu instrukce uložit v paměti postupně za sebou. V některých případech je však třeba posloupnost provádění instrukcí přerušit a pokračovat na jiné adrese. K tomuto účelu slouží instrukce skokové, které způsobí, že obsah čítače instrukcí se nahradí novou adresou a procesor pak pokračuje ve čtení další instrukce z nové adresy v paměti na kterou čítač instrukcí ukazuje.

1.3 Způsoby adresování

Jak již bylo uvedeno, procesor operuje s daty a instrukcemi, které jsou ukládány nebo čteny z registrů (obecně z paměťového místa).

Každý registr má svoji adresu. Způsoby adresování nám potom ukazují, jakým způsobem je v každé instrukci vytvořena adresa registrů, kterých se činnost týká. Z tohoto hlediska můžeme rozlišit:

- přímé a nepřímé adresování
- přímé a nepřímé registrované adresování
- relativní a indexové adresování

Při přímém a nepřímém adresování je v adresové části instrukce obsažena úplná adresa nultého až n-tého řádu. Adresou nultého řádu se rozumí bezprostřední uložení operandu v adresové části instrukčního slova - přímý operand (Immediate Address). Adresa prvního řádu je tzv. přímá adresa (Absolute Address), která přímo určuje paměťové místo nebo vstup/výstup, na kterém je uložen operand.

Adresy vyšších řádů jsou tzv. nepřímé adresy (Indirect Address). Adresa druhého řádu určuje paměťové místo, na němž je zaznamenána přímá adresa. U adres vyšších řádů je odkaz na přímou adresu vícenásobný, adresa v instrukci určuje místo, v němž je nepřímá adresa, atd., až na konci posloupnosti postupně čtených adres je uložena přímá adresa.

Při přímém registrovaném adresování je v adresovém registru mikroprocesoru obsažen přímo operand.

Nepřímé registrové adresování je zvláštní případ nepřímého adresování druhého řádu. V instrukci je nepřímá adresa registru, v němž je uložena přímá adresa paměťového místa.

Při relativním a indexovém adresování se absolutní adresa získá modifikací údaje, uloženého v adresové části instrukce, pomocí obsahu modifikačních registrů. Adresová část instrukce se v případě relativního adresování interpretuje jako posunutí nebo relativní adresa, v případě indexového adresování jako index. Úplnou adresu operandu získáme přičtením relativní adresy nebo indexu k adrese, uložené v modifikačním registru, který se v případě relativního adresování nazývá bázovým registrem (Base Register), v případě indexového adresování indexovým registrem (Index Register).

1.4 Programové vybavení mikropočítače

Obecně je programování mikropočítačů stejné jako programování větších počítačů. Je-li mikropočítač vybaven dostatečnou kapacitou paměti a vhodnými přídavnými zařízeními, může provádět stejné druhy výpočtů, jako větší počítač, ovšem pomaleji. Sestavení programu pro mikropočítač klade obvykle určité specifické požadavky na programátora v důsledku možností a vlastností plynoucích z vlastního mikropočítače. Programátor se snaží především program "minimalizovat" s ohledem na kapacitu paměti mikropočítače, musí přihlédnout k rychlosti provádění instrukcí a k omezenému souboru přídavných zařízení.

Sestavení programu pro mikropočítač znamená sestavení sledu kódovaných instrukcí. U mikropočítače, stejně jako u jiných počítačů, jsou instrukce vyjádřeny v dvojkovém kódu a pomocí nich můžeme programovat. Tento způsob programování - ve strojovém jazyku - je ovšem náročný, zdlouhavý a nepohodlný, i když si jej usnadníme vyjádřením čísel v hexadecimálním kódu, ve kterém se programátor může snáze orientovat. Proto se pro programování obvykle používá **jazyk symbolických adres.** V tomto jazyku je každý operační znak mnemotechnickou zkratkou příslušné operace v angličtině, např. ADD (přičti) atd. Program napsaný v jazyku symbolických adres potom přeložíme pomocí kódové tabulky do strojového kódu, nebo lépe - pomocí speciálního programu nazývaného **asembler** přeložíme mnemotechnické zkratky do strojového jazyka. Překlad pomocí tabulky provádíme většinou pouze v případě krátkých programů určených pro jednodeskové mikropočítače. Takto přeložený program potom zavedeme do mikropočítače, který pro tento účel obvykle bývá vybaven hexadecimální klávesnicí a jednořádkovým hexadecimálním displejem. Ovládání takovýchto mikropočítačů bývá dodáváno v paměti ROM a je tvořeno programem nazývaným MONITOR. Typickým představitelem takových mikropočítačů je - viz kap.3 - školní mikropočítač ŠMS-VÚVT. Další postup ladění programu spočívající ve využití vlastností MONITORU ukážeme v kap.3.

Druhá možnost - strojový překlad - je realizovatelná buď použitím nevlastního překladače (křížového - Cross Assembler) jazyka symbolických adres na větším počítači a nebo nejkomfortněji na specializovaném **vývojovém systému.**

Vývojový systém pro práci s mikroprocesory má za základ mikropočítač s řadou podpůrných periférií a zejména s rozsáhlým programovým vybavením umožňujícím a usnadňujícím návrh programu. Vývojový systém ve svém programovém vybavení - operačním systému - má zejména:

MONITOR - soubor programů pro základní řízení systémů, změny dat v paměti, registrech, zavádění programu atd.

TEXT EDITOR - redakční program, umožňující práci s řetězci znaků, sloužící pro přípravu zdrojového programu

DEBUGGER - ladicí program pro odladění uživatelského programu

ASSEMBLER - program pro překlad uživatelského programu z jazyka symbolických adres do strojového kódu

Dále obsahuje programové vybavení překladače z vyšších programovacích jazyků (BASIC, FORTRAN, PL/M, FORTH atd.).

Odladění programů a jejich aplikace je usnadněna tzv. **obvodovým emulátorem,** což je soubor programového a technického vybavení, umožňující, aby vývojový systém sdílel s vyvíjeným vzorkem svůj mikroprocesor, část operační paměti a vstupní/výstupní brány.

Zde je třeba si uvědomit, že použití vyšších programovacích jazyků sice zrychlí návrh programu a usnadní jeho odladění, ale na druhé straně většinou znamená, že výsledný program - vlivem vlastností překladače - zabere v paměti více místa, než program, který by v jazyku symbolických adres napsal zkušený programátor. Přestože cena pamětí stále klesá, je zřejmé, že tam, kde je nutné využít maximálně nejen kapacitu paměti, ale zejména rychlost provádění programu, bude mít jazyk symbolických adres přednost před vyššími programovacími jazyky.

1.5 Jazyk symbolických adres (Assembly Language)

V dalším budou uvedeny pouze nejdůležitější zásady, bližší podrobnosti nalezne čtenář v příslušné literatuře.

Formulář pro zápis programu v jazyku symbolických adres je rozdělen do šesti vertikálních polí (obr. 1.3). První a druhé pole obsahují absolutní adresu instrukcí v paměti a strojový kód instrukce.

Absolutní adresa (Address)	Strojový kód (Object-Code)	Symbolická adresa - návěstí (Label)	Operační znak (Opcode)	Operand(y) (Operand)	Komentář (Komment)
0028	E5	RST 5:	PUSH	H	; ulož HL na ; vrchol zásobníku
0229	2AEC 83		LHLD	83EC H	; dej do HL ; adresu
002C	E3		XTHL		; vyměň HL ; a vrchol zásobníku
002D	C9		RET		; návrat

Obr. 1.3 Formulář pro zápis programu v jazyce symbolických adres

Třetí pole obsahuje návěstí (symbolické adresy) důležitých míst programu, ve čtvrtém poli je zapsána mnemotechnická značka instrukce, v pátém poli operand nebo operandy, se kterými instrukce operuje a v posledním - šestém - poli zapisujeme komentář programu . Komentář sice není třeba k provedení vlastního programu, ale je nutný pro orientaci programátora. Jak je zřejmé pro zápis programu do formuláře nepotřebujeme znát absolutní adresy jednotlivých instrukcí ani jejich strojový kód. Tyto údaje získáme až při překladu programu. Přeložení programu můžeme provádět buď "ručně" podle tabulek, nebo použít pomocný program, který překlad provede.

Pro zápis programu v jazyce symbolických adres platí určitá pravidla definovaná původcem jazyka. Na tomto místě se zmíníme alespoň o některých zásadách platných pro firemní jazyk mikroprocesoru 8080A:

1) Návěstí - Návěstí je nepovinné, nemusí tedy být na každém řádku programu. Obvykle návěstím označujeme adresy skoků, začátky podprogramů atd. Návěstí může mít maximálně 6 alfanumerických znaků.První znak musí být písmeno a za návěstím je dvojtečka.

2) Operační znak - Operační znak musí být ze souboru mnemotechnických značek instrukcí, nebo toto pole může obsahovat tzv. pseudoinstrukce, což jsou příkazy pro řízení strojního překladu programu překladačem.

3) Operand - V poli operandů je značka registru nebo registrů, s jejichž obsahem instrukce operuje, nebo přímý operand (číslo), návěstí adresy nebo i jiné operandy. Pokud instrukce vyžaduje dva operandy, první zleva určuje cílový registr, druhý registr zdrojový. Operand je od operačního znaku instrukce oddělen alespon jednou mezerou, operandy jsou navzájem odděleny čárkou. Operandem může být i znak v kódu ASCII, takovýto znak se zapíše v apostrofech.

Čísla značíme:	binární	:	0 0 0 1 1 1 1 0 B
	dekadická	:	32D nebo 32
	hexadecimální	:	18H

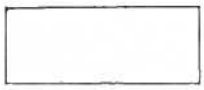
Hexadecimální číslo musí začínat číslicí, např. 0BFH a ne BFH.

4) Komentář - Pole komentářů začíná vždy středníkem. Na řádku může být pouze komentář.

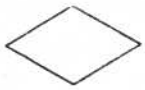
1.6 Grafické značky pro zápis algoritmů

Řešení úlohy, pro níž chceme sestavit program, je vhodné popsat tzv. vývojovým diagramem, který graficky zachycuje činnost programu. Pro sestavování vývojových diagramů se používají ustálené grafické značky, z nichž nejdůležitější jsou uvedeny dále:

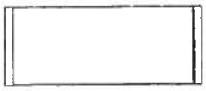
Mezní značka.
Označení mezního místa vývojového diagramu (např. začátku, konce, zastavení.



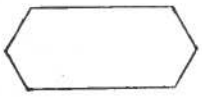
Operace nebo skupina operací.
Činnost zapisujeme do obdélníku



Rozhodování, větvení, přepínání.
Dovnitř zapisujeme podmínku, jejíž splnění ovlivňuje další průběh algoritmu.

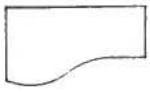


Předem definovaná činnost (Podprogram)
Představuje skupinu činností nebo kroků, které jsou specifikovány jinde a v daném vývojovém diagramu nejsou rozpracovány.

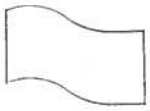


Příprava.
Představuje modifikaci činnosti nebo skupiny činností, které mění vlastní postup.

Vstup nebo výstup.
Dodání vstupních údajů pro zpracování (vstup) nebo zobrazení (záznam) výsledků (výstup).



Doklad.
Vstupní nebo výstupní operace, ve které je médiem doklad (např. výpis na tiskárně).



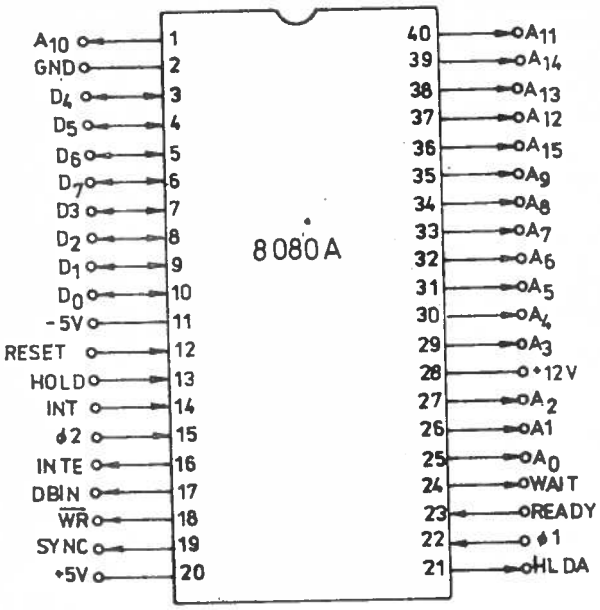
Děrná páska.
Vstupní nebo výstupní operace, ve které je médiem děrná páska.

Spojka.
Vyznačuje pevné spojení dvou vzdálených částí vývojového diagramu.

Poznámka.
Značka slouží pro zápis poznámky. Lze ji připojit ke všem ostatním značkám. Značku lze kreslit i otočenou o 180°.

Spojnice.
Slouží ke spojení jednotlivých značek vývojového diagramu. Neorientovaná spojnice určuje směr shora dolů nebo zleva doprava. Jiný směr se musí vyznačit šipkou.

2. MIKROPROCESOR 8080A



Obr. 2.1 Rozmístění vývodů 8080A

Mikroprocesor 8080A je jednočipový osmibitový paralelní mikroprocesor pro všeobecné použití. Přenáší data a stavové informace po osmibitové obousměrné sběrnici (D0 až D7). Adresy pro paměť a periferní jednotky jsou přenášeny přes oddělenou šestnáctibitovou třístavovou adresovou sběrnici (A0 až A15). Mikroprocesor 8080A má 6 výstupů pro časové a řídicí impulsy (SYNC, DBIN, WAIT, HLDA,INTE), čtyři řídicí vstupy (READY,INT, HOLD, RESET), čtyři vstupy pro napájení (+12V, +5V, -5V, ZEM) a dva vstupy pro hodinové impulsy (Ø1, Ø2).

Rozmístění vývodů pouzdra 8080A ukazuje obr.2.1, význam jednotlivých signálů tabulka 2.1.

Označení vývodů	Význam
A0 až A15	Adresová sběrnice - třístavový výstup.
D0 až D7	Datová sběrnice. Třístavový vstup/výstup.
SYNC	Synchronizační signál indikující začátek každého taktu a signalizující výstup stavového slova mikroprocesoru na datové sběrnici.
DBIN	Signál indikující, že datová sběrnice je ve vstupním režimu (Data Bus In). Aktivace vstupních dat z paměti nebo V/V na datovou sběrnici.
READY	Signál oznamující mikroprocesoru, že data na datové sběrnici jsou platná. Slouží k synchronizaci s pomalejší pamětí či periferními zařízeními. Je-li READY = 0, přechází procesor do stavu WAIT.
WAIT	Signál oznamující, že mikroprocesor je v čekacím stavu (stav WAIT)
$\overline{WR}$	Signál oznamující, že na datové sběrnici jsou platná výstupní data ($\overline{WR}$ = 0).
HOLD	Vnější signál pro převedení mikroprocesoru do stavu HOLD. V tomto stavu se zastavuje součinnost mikroprocesoru s vnějšími obvody a vývody A0 až A15 i D0 až D7 přecházejí do vysokoimpedančního stavu. Obě vnější sběrnice pak mohou být využity např. pro kanál přímého přístupu do paměti (DMA).
HLDA	Signál, kterým mikroprocesor potvrzuje splnění požadavku HOLD (Hold Acknowledge).

INTE	Signál indikující stav vnitřního klopného obvodu, který povoluje akceptovat požadavek na přerušení. Je-li INTE = 1, bude požadavek uspokojen. Tento klopný obvod lze v programu ovládat instrukcemi EI (Enable Interrupt) a DI (Disable Interrupt). Do nuly se rovněž nastavuje signálem RESET a na začátku každého přerušovacího taktu M1.
INT	Vstupní signál požadující přerušení programu (Iterrupt Reguest). Je-li INTE = 0, nebo je mikroprocesor ve stavu HOLD, pak požadavek není akceptován. Je-li požadavek přijat, pak je to vždy jen po skončení právě probíhající instrukce.
RESET	Vstupní signál vyvolávající skok na adresu 0 v hlavní paměti a vynulování signálů INTE a HLDA. Signál RESET musí trvat nejméně po dobu tří period hodinového signálu.

Tab. 2.1 Označení a funkce vývodů 8080A

2.1 Struktura mikroprocesoru 8080A

Blokové schéma mikroprocesoru 8080A ukazuje obrázek 2.2.
Mikroprocesor 8080A obsahuje následující funkční bloky:

- soubor registrů a adresovací logiku
- obvod pro řízení obousměrných datových sběrnic s třístavovým výstupem
- aritmetickou a logickou jednotku (ALU)
- instrukční (řídicí) registr a řídicí logiku

2.1.1 Registry 8080A

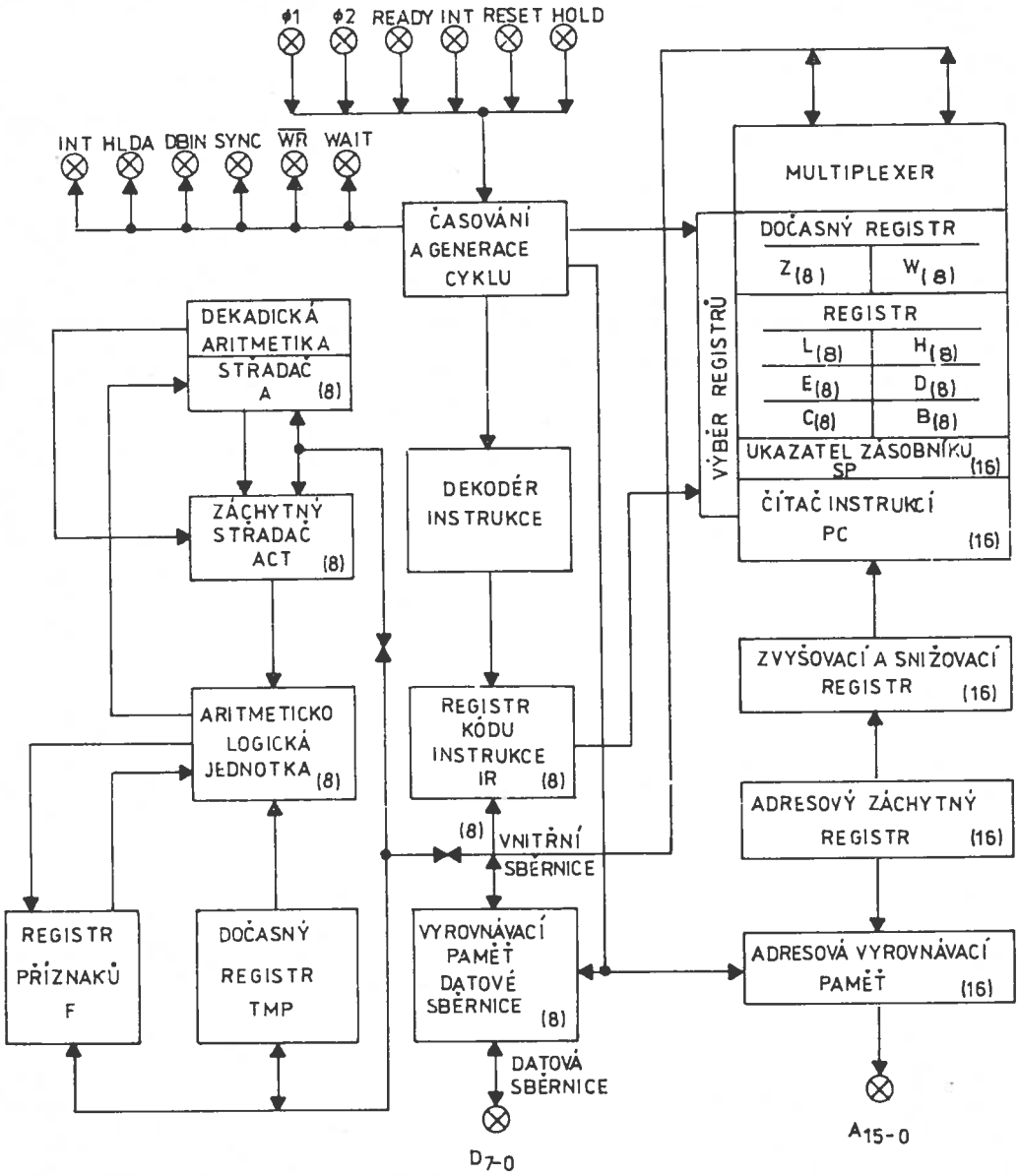
Registrová část se skládá z pole statických pamětí RAM, které jsou uspořádané do šesti šestnáctibitových registrů:

- čítač instrukcí (Program Counter - PC)
- ukazatel zásobníku (Stack-Pointer - SP)
- šest osmibitovývh registrů pro všeobecné použití, párově uspořádaných s označením B, C, D, E a H, L.
- dvojice registrů s označením W,Z pro přechodné uložení dat

Čítač instrukcí obsahuje adresu paměti právě probíhající instrukce programu a při každém vyžádání instrukce se automaticky zvětšuje o 1.Ukazatel zásobníku obsahuje adresu dalšího místa skupiny dat v paměti. Může být nastaven pro každý program na počáteční hodnotu, takže může využít libovolného rozsahu paměti RWM. Ukazatel zásobníku se snižuje vždy, když jsou do zásobníku předávána data, a zvyšuje se, jsou-li data ze zásobníku odebírána.

Šest registrů pro všeobecné použití lze použít buď jako samostatné osmibitové registry, nebo jako šestnáctibitové dvojice registrů (párové registry). Dvojici registrů W a Z nelze adresovat z programu a využívá se pouze pro interní provedení instrukcí. Osmibitová slova mohou být přenášena mezi vnitřní sběrnicí a souborem registrů přes multiplexer (Register Select Multiplexer). Šestnáctibitový přenos je možný mezi souborem registrů a pamětí adresy, nebo obousměrným čítačem.

paměť adres přijímá data z jedné dvojice registrů a řídí 16 adresových výstupních oddělovacích členů (A0 až A15), stejně jeko obousměrný čítač dostává data z adresové paměti a přenáší je do souboru registrů. Šestnáctibitová data se buď přičítají nebo odečítají, nebo se jednoduše přenášejí mezi registry.



Obr.2.2 Struktura mikroprocesoru 8080A

2.1.2 Aritmeticko logická jednotka (ALU)

ALU se skládá z následujících registrů:

- osmibitový střadač
- osmibitový pomocný střadač pro přechodné uložení dat (ACT)
- pětibitový registr příznaků (Flag Register): nula, přenos, znaménko, parita a pomocný přenos
- osmibitový registr pro přechodné uložení dat (TMP)

Aritmetické, logické a další operace vykonává ALU. ALU dostává data z registru ACT a přenosového klopného obvodu. Výsledek operace se může přenášet buď do interní sběrnice, nebo do střadače. ALU dodává data rovněž do registru příznaků. Registr pro přechodné uložení dat (TMP) přijímá informace z interní sběrnice a předává je buď úplně nebo částečně dále do ALU, stavového registru a na interní sběrnici. Střadač (ACC) může být zásobován buď z ALU nebo z interní sběrnice.Předává data dále na paměť střadače (ACT) a na interní sběrnici. Obsah střadače (ACC) a pomocného přenosového klopného obvodu může být během instrukce DAA nastaven pro práci v dekadické soustavě. Výsledek je interpretován v kódu BCD ve 2 skupinách. Z hlediska programátora je důležité slovo udávající stav registru příznaků doplněného pomocnými bity. Formát slova i význam jednotlivých příznaků ukazují tab. 2.2 a tab. 2.3.

b7	b6	b5	b4	b3	b2	b1	b0
S	Z	0	AC	0	P	1	CY

Tab. 2.2 Formát slova registru příznaků

Označení příznaku	Název příznaku		Nastavuje se po aritmetické nebo logické operaci do stavu 1 za podmínky
S	Znaménko	(Sign)	stav bitu b7 výsledku je roven 1
Z	Nula	(Zero)	výsledek je nulový
AC	Pomocný přenos	(Auxiliary Carry)	při operacích sčítání nebo odečítání došlo k přenosu z bitu b3 do bitu b4
P	Parita	(Parity)	výsledek má sudou paritu
CY	Přenos	(Carry)	po operaci sčítání došlo k přenosu z bitu b7 výsledku, po operaci odečítání k přenosu z bitu b7 nedošlo (indikuje výpůjčku)

Tab. 2.3 Význam příznaků registru F

2.1.3 Řídicí registr a řízení

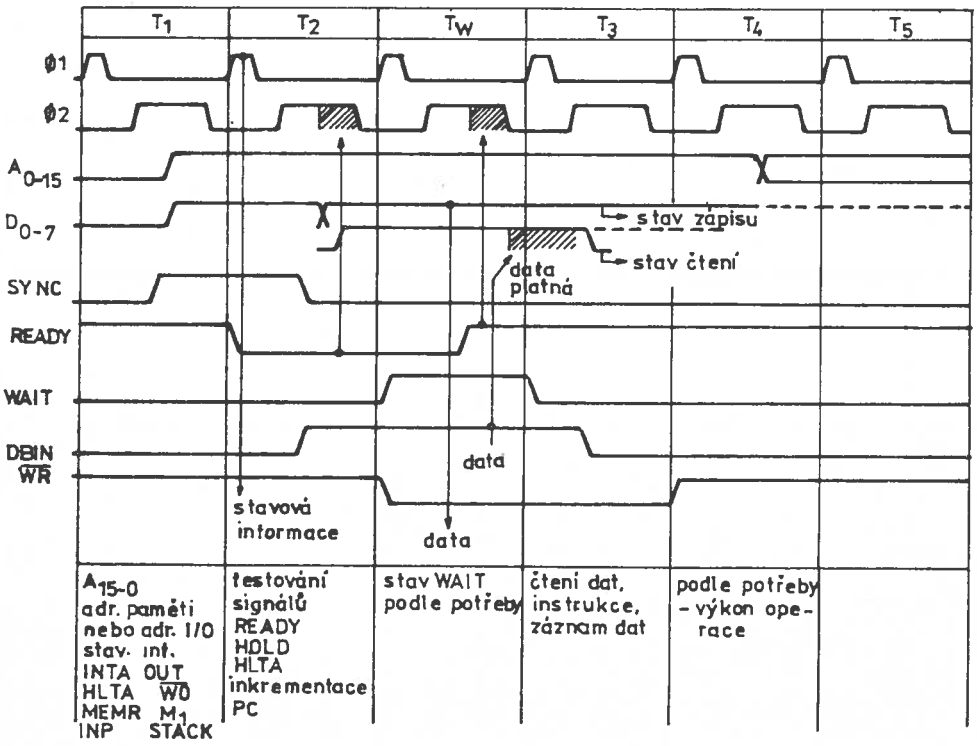
Během vyžádání instrukce je první slabika instrukce (s operačním kódem) přenášena přes interní sběrnici do osmibitového řídicího (instrukčního) registru. K obsahu řídicího registru má přístup dekodér instrukcí. Z výstupu dekodéru vycházejí řídicí signály, kombinované s různými časovými impulsy, do souboru registrů, ALU a zásobníkové paměti. Odtud vycházejí výstupní signály instrukčního dekodéru spolu s externími řídicími signály do řídicí části zapojení a odtud se dodávají časové cyklovací a stavové signály.

2.1.4 Datové oddělovací obvody pro sběrnici (Data Bus Buffer)

Tato osmibitová zásobníková paměť slouží k oddělení interní sběrnice 8080A od externí (D0 až D7). Při výstupní operaci se uloží nejprve obsah interní sběrnice do osmibitové paměti, která potom vybudí výstupní paměť datové sběrnice. Výstupní paměť je během zadávání a mimo přenosové operace odpojená. Při zadávání se přenášejí data z externí datové sběrnice na sběrnici interní.

2.2 Časování mikroprocesoru 8080A

Práce mikroprocesoru probíhá v taktech. Pro vykonání každé instrukce je v závislosti na typu instrukce zapotřebí jeden až pět taktů označených M1 až M5 . Každý takt sestává ze 3 až 5 dob T1 až T5, z nichž každá trvá právě jednu periodu hodin. Navíc ve zvláštních případech jsou možné stavy Wait, Hold a Halt, které trvají celistvý počet dob určovaný vnějšími signály. Pro obvyklou periodu hodin 500 ns se uvádí, že typická délka taktu je 2 μs. Většinou však trvá nejkratší takt jen 1.5 μs, neboť takt instrukčního cyklu běžně obsahuje jen 3 doby T1 až T3. Pokud nejsou takty cyklu prodlužovány o stavy Wait, Hold nebo Halt, pak nejkratší možný instrukční cyklus trvá 1 takt se 4 dobami (2 μs) a nejdelší 5 taktů s celkem 18 dobami (9 μs). Průběh taktu prodlouženého o dobu Tw je znázorněn na obr.2.3.



Obr.2.3 Časování mikroprocesoru 8080A

V jednotlivých dobách vykonává mikroprocesor následující činnost:

T1 : Během doby T1 se na adresovou sběrnici A0 až A15 přivede adresa slabiky v paměti nebo adresa vstupu či výstupu. Na datovou sběrnici D0 až D7 se současně v době trvání signálu SYNC odesílá tzv. informace o stavu. Je to osm jednobitových signálů, tvořících stavové slovo.

Jsou to tyto signály :

INTA Signál potvrzující akceptování požadavku na přerušení. Signálem může být řízeno přivedení instrukce RST (RESTART) z periferního zařízení na datovou sběrnici v době, kdy signál DBIN=1.

$\overline{WO}$ $\overline{WO}$=0 indikuje, že v probíhajícím taktu bude probíhat výdej dat do paměti nebo na výstup (zápis). V opačném případě budou probíhat operace typu příjem dat (čtení).

STACK Indikuje, že signál na adresové sběrnici je adresa z ukazatele zásobníku.

HLTA Potvrzující signál pro instrukci HLT.

OUT Indikuje, že adresová sběrnice obsahuje adresu výstupu, a že datová sběrnice bude obsahovat výstupní data (až nastane $\overline{WR}$=0).

M1 Potvrzuje, že mikroprocesor je v taktu M1 (Fetch cykle), ve kterém se vždy čte první slabika instrukce, tj. operační znak.

INP Indikuje, že adresová sběrnice obsahuje adresu vstupu, a že vstupní data mohou být vložena na datovou sběrnici při DBIN = 1.

MEMR Určuje, že datová sběrnice bude použita pro čtení dat z paměti.

Signály jsou platné v době T1 při SYNC = 1.

Celkem může probíhat deset různých typů taktů:

1) Čtení operačního znaku instrukce (Op Code Fetch) (M1)
2) Čtení z paměti (Memory Read)
3) Zápis do paměti (Memory Write)
4) Čtení zásobníku (Stack Read)
5) Zápis do zásobníku (Stack Write)
6) Čtení vstupní brány (Input)
7) Zápis do výstupní brány (Output)
8) Přijetí žádosti o přerušení (Interrupt)
9) Provedení instrukce HLT (Halt)
10) Přijetí žádosti o přerušení ve stavu HALT (Halt . Interrupt)

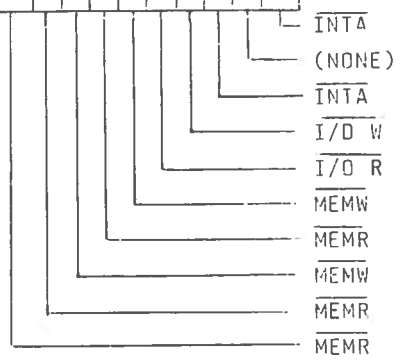
T2 : Doba T2 vždy následuje za dobou T1. V T2 inkrementuje čítač instrukcí PC a mikroprocesor testuje signály READY, HOLD a HLTA. Jestliže READY = 1, následuje doba T3. V opačném případě nastane přechod do stavu Wait, který trvá n dob Tw, dokud se nesplní podmínka READY = 1. Jestliže HOLD = 1, projde se ještě dobou T3 a pak datová i adresová sběrnice přejdou do vysokoimpedančního stavu. Pokud má vykonávaný takt obsahovat i doby T4 a T5, mohou proběhnout interně i při odpojených sběrnicích. Potom nastane doba Tw. V případě instrukce HLT (HLTA = 1) ihned po době T2 se činnost mikroprocesoru zastaví, tj. následují doby Tw, dokud tento speciální stav není zvenčí přerušen.

T3 : Doba T3 následuje po T2 nebo Tw. V taktu M1 je v průběhu T3 vždy z paměti přečtena první slabika instrukce (operační znak) a je uložena do registru instrukce. V jiných taktech než M1 se v době T3 čtou buď další slabiky instrukce, nebo data z paměti, nebo se data zapisují do paměti.

T4 a T5 : V dobách T4 a T5 probíhá vlastní vykonání instrukce, pokud neskončilo již dříve. Tyto doby nemusí nastat u všech instrukcí ani ve všech taktech.

V tabulkách instrukcí se uvádí počet taktů i dob potřebný k vykonání každé instrukce. Tyto počty můžeme používat k bilanci času, resp. rychlosti výkonu programu. Stavové slovo, vysílané v době T1, lze zachytit v registru a použít pro řízení dalších obvodů. Přehled označení stavů je uveden v tab. 2.4. Příklad řízení systému ukazují obr. 2.5 a 2.6 v odd. 2.8.

Bit datové sběrnice	Stavová informace	Pracovní režim									
		1	2	3	4	5	6	7	8	9	10
D0	INTA	0	0	0	0	0	0	0	1	0	1
D1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1
D2	STACK	0	0	0	1	1	0	0	0	0	0
D3	HLTA	0	0	0	0	0	0	0	0	1	1
D4	OUT	0	0	0	0	0	0	1	0	0	0
D5	M1	1	0	0	0	0	0	0	1	0	1
D6	INP	0	0	0	0	0	1	0	0	0	0
D7	MEMR	1	1	0	1	0	0	0	0	1	0



Poznámka:
1 Čtení operačního znaku instrukce (Op Code Fetch)
2 Čtení z paměti (Memory Read)
3 Zápis do paměti (Memory Write)
4 Čtení zásobníku (Stack Read)
5 Zápis do zásobníku (Stack Write)
6 Čtení vstupní brány (Input)
7 Zápis do výstupní brány (Output)
8 Potvrzení přerušení (Interrupt)
9 Provedení instrukce HLT (Halt)
10 Potvrzení žádosti o přerušení ve stavu HALT (Halt . Interrupt)
N Stavové slovo

Tab. 2.4 Přehled označení stavů 8080A

2.3 Průběhy přerušení

Přerušení vyžádá periferie uvedením přerušovacího vstupu (INT) na úroveň H. Vstup INT je asynchronní (požadavek na přerušení může vzniknout kdykoliv) a interní logika mikroprocesoru zabezpečuje jeho časové sladění s hodinovým kmitočtem. Soubor instrukcí obsahuje zvláštní jednoslabikovou instrukci, která zjednodušuje zpracování přerušení (obvyklá vyvolávací instrukce pro program potřebuje 3 Byte). Tuto instrukci nazýváme "RESTART". Proměnné tříbitové pole v osmičkovém instrukčním kódu RST umožňuje přerušovacímu obvodu adresovat skokem jedno z 8 pevných míst v paměti. Desítkové adresy těchto rezervovaných míst v

paměti jsou: 0, 8, 16,24, 32,40, 48 nebo 56. Tyto adresy se smí používat pro uložení první, popř. prvních instrukcí pro jednu operaci do paměti. Protože RST je vyvolávací instrukce, je po jejím ukončení uložen původní obsah čítače adres do zásobníkové paměti.

2.4 Průběh HOLD

Stav HOLD je umožněn tím, že periferie vydá signál HOLD procesoru pro přímý přístup do paměti. Procesor reaguje na tento požadavek tím, že převede adresovou sběrnici do stavu velké impedance a periferním obvodům přenechá stanovení adresy na adresové sběrnici. Procesor potvrzuje signál HOLD tím, že výstup HLDA přejde na úroveň H.
Stejně jako vstup pro přerušení, tak i vstup HOLD je interně synchronizován.

2.5 Průběh při stavu HALT

Při instrukci HALT přejde mikroprocesor 8080 po kroku T2 následujícího operačního cyklu do stavu HALT (TWH). Pro opuštění tohoto stavu existují tři možnosti:

1. Úroveň H na vodiči RESET nastaví mikroprocesor 8080 vždy zpět do stavu T1. RESET také nuluje čítač instrukcí a nastavuje registr instrukcí do nuly.
2. Signál HOLD uvede mikroprocesor 8080 vždy do stavu HOLD. Dostane-li se vodič HOLD do stavu L, je uveden mikroprocesor 8080 náběžnou hranou následujícího impulsu Ø1 do stavu HALT.
3. Instrukce pro přerušení (INT má úroveň H a INTE je nastaven pro výběr) dá mikroprocesoru 8080 podnět k opuštění stavu HALT a při příchodu náběžné hrany následujícího hodinového impulsu Ø1 přejde mikroprocesor do operačního stavu T1.

Poznámka: Je-li požadován stav HALT, musí být nastaven výstup INTE, jinak může mikroprocesor 8080 opustit stav HALT pouze přes signál RESET.

2.6 Uvedení mikroprocesoru 8080A do provozu

Mikroprocesor 8080 je schopen začít pracovat jakmile se připojí napájecí napětí. Obsah čítače adres, ukazatele zásobníku a ostatních pracovních registrů je po připojení napájení závislý na náhodných faktorech a tedy nedefinován. Procesor nastavíme do výchozího stavu pomocí externího signálu RESET , který trvá nejméně tři periody hodin (1,5 μs) a který nastaví čítač instrukcí na nulu. Program začíná potom od místa 0 v paměti.

U systémů, u kterých musí procesor čekat na určitý zvláštní startovací signál, je na prvních dvou místech v paměti uložena instrukce HALT (EI,HLT). Pro startování, které je ruční, se používá instrukce pro přerušení (INT). U ostatních systémů může procesor začít ihned se zpracováním uživatelského programu. Přitom je třeba si uvědomit, že nastavení nulové polohy signálem RESET neovlivňuje registr příznaků ani pracovní registry procesoru (střadač, registry B, C, D, E, H, L nebo ukazatel zásobníku), . Obsah těchto registrů zůstává nedefinován, dokud nejsou nastaveny do výchozího stavu programem.

2.7 Instrukční soubor mikroprocesoru 8080A

Podle údajů výrobce má instrukční soubor mikroprocesoru 8080A celkem 78 strojových instrukcí, je však možné dojít k více či méně rozdílným počtům v závislosti na způsobu, jak jsou jednotlivé operační znaky rozděleny do skupin jimž je přiřa-

MNEM	Operandy	hexa	binární kód	počet slabik	Dob	Účinek	ovlivňuje příznak	anglický popis
MOV	r1,r 2		01DDDSSS	1	5	(r1) ← (r2)		move register
MOV	r,M,H & L		01DDD110	1	7	(r) ← (M,H & L)		move from memory
MOV	M,r,H & L		01110SSS	1	7	(M,H & L) ← (r)		move to memory
MVI	r,data		00DD0110	2	7	(r) ← (B2)		move immediate
MVI	M,data,H & L	36	00110110	2	10	(M,H & L) ← (B2)		move to memory immediate
LXI	rp,data 16		00rp0001	3	10	(rh) ← (B3),(rl) ← (B2)		load register pair immediate
LDA	adr,A	3A	00111010	3	13	(A) ← (M,B3B2)		load accumulator direct
LDAX	rp,A		00rp1010	1	7	(A) ← (M,rp)		load accumulator indirect
LHLD	adr,H & L	2A	00101010	3	16	(L) ← (M,B3B2) (H) ← (M,B3 B2+1)		load H & L direct
SPHL	H & L,SP	F9	11111001	1	5	(SP) ← (H & L)		move H & L to SP
XCHG	H & L,D & E	EB	11101011	1	4	(H) ↔ (D),(L) ↔ (E)		exchange H & L with D & E
XTHL	H & L,SP,M	E3	11100011	1	18	(L) ↔ (M,SP) (H) ↔ (M,SP+1)		exchange stack top with H & L
PUSH	rp,M,SP		11rp0101	1	11	(M,SP-1) ← (rh) (M,SP-2) ← (rl)		push
PUSH PSW	M,SP,A,Flags	F5	11110101	1	11	(M,SP-1) ← (A) (M,SP-2) ← Flags		push processor status word S Z 0 AC 0 P 1 CY
POP	rp,SP,M		11rp0001	1	10	(rl) ← (M,SP) (rh) ← (M,SP+1)		pop
POP PSW	M,SP,A,Flags	F1	11110001	1	10	(Flags) ← (M,SP) (A) ← (M,SP+1)	Z,S,P,CY,AC	pop processor status word
STAX	rp,M,A		00rp0010	1	7	(M,rp) ← (A)		store accumulator indirect
STA	adr,A	32	00110010	3	13	(M,B3B2) ← (A)		store accumulator direct
SHLD	adr,H & L	22	00100010	3	16	(M,B3B2) ← (L) (M,B3B2+1) ← (H)		store H & L direct

Tabulka 2.5 : Instrukční soubor 8080A - přesunové instrukce

MNEM	Operandy	hexa	binární kód	počet slabik	Dob	účinek	ovlivňuje příznak	anglický popis
ADD	A,r		10000SSS	1	4	(A) ← (A)+(r)	Z,S,P,CY,AC	add register
ADD	A,M,H & L	86	10000110	1	7	(A) ← (A)+(M,H & L)	Z,S,P,CY,AC	add memory
ADI	A,B2	C6	11000110	2	7	(A) ← (A)+(B2)	Z,S,P,CY,AC	add immediate
ADC	A,r,CY		10001SSS	1	4	(A) ← (A)+(r)+(CY)	Z,S,P,CY,AC	add register with carry
ADC	A,M,H & L,CY	8E	10001110	1	7	(A) ← (A)+(M,H & L)+(CY)	Z,S,P,CY,AC	add memory with carry
ACI	A,data,CY	CE	11001110	2	7	(A) ← (A)+(B2)+(CY)	Z,S,P,CY,AC	add immediate with carry
DAD	rp,H & L		00rp1001	1	10	(H & L) ← (H & L)+(rh)(rl)	CY	add register pair to H & L
INR	r		00DDD100	1	5	(r) ← (r)+1	Z,S,P,AC	increment register
INR	M,H & L	34	00110100	1	10	(M,H & L) ← (M,H & L)+1	Z,S,P,AC	increment memory
INX	rp		00rp0011	1	5	(rh)(rl) ← (rh)(rl)+1		increment register-pair
SUB	r		10010SSS	1	4	(A) ← (A)-(r)	Z,S,P,CY,AC	subtract register
SUB	A,M,H & L	96	10010110	1	7	(A) ← (A)-(M,H & L)	Z,S,P,CY,AC	subtract memory
SUI	A,data	D6	11010110	2	7	(A) ← (A)-(B2)	Z,S,P,CY,AC	subtract immediate
SBB	A,r,CY		10011SSS	1	4	(A) ← (A)-(r)-(CY)	Z,S,P,CY,AC	subtract register with borrow
SBB	A,M,H & L,CY	9E	10011110	1	7	(A) ← (A)-(M,H & L)-(CY)	Z,S,P,CY,AC	subtract memory with borrow
SBI	A,data,CY	DE	11011110	2	7	(A) ← (A)-(B2)-(CY)	Z,S,P,CY,AC	subtract immediate with borrow
DCR	r		00DDD101	1	5	(r) ← (r)-1	Z,S,P,AC	decrement register
DCR	M,H & L	35	00110101	1	10	(M,H & L) ← (M,H & L)-1	Z,S,P,AC	decrement memory
DCX	rp		00rp1011	1	5	(rh),(rl) ← (rh),(rl)-1		decrement register pair

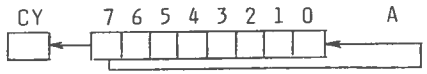
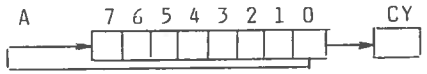
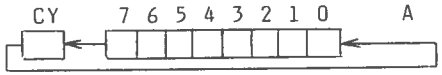
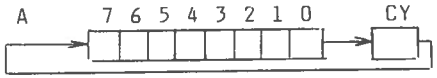
Tabulka 2.6 Instrukční kód 8080A - matematické instrukce

MNEM	operandy	hexa	binární kód	počet slabik	Dob	účinek	ovlivňuje příznak	anglický popis
ANA	A,r		10100SSS	1	4	(A) ← (A) ∧ (r)	Z,S,P,CY,AC	AND register
ANA	A,M,H & L	A6	10100110	1	7	(A) ← (A) ∧ (M,H & L)	Z,S,P,CY,AC	AND memory
ANI	A,data	E6	11100110	2	7	(A) ← (A) ∧ (B2)	Z,S,P,CY,AC	AND immediate
XRA	A,r		10101SSS	1	4	(A) ← (A) ⊻ (r)	Z,S,P,CY,AC	exlusive OR register
XRA	A,M,H & L	AE	10101110	1	7	(A) ← (A) ⊻ (M,H & L)	Z,S,P,CY,AC	exlusive OR memory
XRI	A,data	EE	11101110	2	7	(A) ← (A) ⊻ (B2)	Z,S,P,CY,AC	exlusive OR immediate
ORA	A,r		10110SSS	1	4	(A) ← (A) ∨ (r)	Z,S,P,CY,AC	OR register
ORA	A,M,H & L	B6	10110110	1	7	(A) ← (A) ∨ (M,H & L)	Z,S,P,CY,AC	OR memory
DRI	A,data	F6	11110110	2	7	(A) ← (A) ∨ (B2)	Z,S,P,CY,AC	OR immediate
CMP	A,r		10111SSS	1	4	(A)-(r)	Z,S,P,CY,AC	compare register
CMP	A,M,H & L	BE	10111110	1	7	(A)-(M,H & L)	Z,S,P,CY,AC	compare memory
CPI	A,data	FE	11111110	2	7	(A)-(B2)	Z,S,P,CY,AC	compare immediate

Tabulka 2.7 Instrukční soubor 8080A - aritmetické a logické instrukce

MNEM	operandy	hexa	binární kód	počet slabik	Dob	účinek	ovlivňuje příznak	anglický popis
JMP	PC,adr	C3	11000011	3	10	(PC) ← (B3,B2)		jump unconditional
J(CCC)	PC,adr		11CCC010	3	10	(PC) ← (B3,B2) if		jump conditional
CALL	PC,M,SP,adr	CD	11001101	3	17	(M,SP) ← (PC) (PC) ← (B3,B2)		call unconditional
C(CCC)	PC,M,SP,adr		11CCC100	3	11/17	(M,SP) ← (PC) (PC) ← (B3,B2) if		call conditional
RET	PC,M,SP	C9	11001001	1	10	(PC) ← (M,SP)		return umconditional
R(CCC)	PC,M,SP		11CCC000	1	5/11	(PC) ← (M,SP) if		return conditional
RST	M,SP,PC		11NNN111	1	11	(M,SP) ← (PC) (PC) ← (NNN)·8		restart
PCHL	PC,M,H&L	E9	11101001	1	5	(PC) ← (M,H&L)		jump H&L indirect, move H&L to PC

Tabulka 2.8 Instrukční soubor 8080A - skokové instrukce

MNEM	operandy	hexa	binární kód	počet slabik	Dob	účinek	ovlivňuje příznak	anglický popis
RLC	A	07	00000111	1	4	(An+1)←(An) (A0)← (A7) (CY)←(A7)	CY	rotate accumulator left 
RRC	A	0F	00001111	1	4	(An)←(An-1) (A7)←(A0) (CY)←(A0)	CY	rotate accumulator right 
RAL	A	17	00010111	1	4	(An+1)←(An) (CY)←(A7) (A0)←(CY)	CY	rotate accumulator left trough carry 
RAR	A	1F	00011111	1	4	(An)←(An+1) (CY)←(A0) (A7)←(CY)	CY	rotate accumulator right trough carry 

Tabulka 2.9 Instrukční soubor 8080A - posunové instrukce

MNEM	operandy	hexa	binární kód	počet slabik	Dob	účinek	ovlivňuje příznak	anglický popis
STC	CY	37	00110111	1	4	(CY) ← 1	CY	set carry
DAA	A	27	00100111	1	4		Z,S,P,CY,AC	decimal adjust accumulator
CMA	A	2F	00101111	1	4	(A) ← (A)		complement accumulator
CMC	CY	3F	00111111	1	4	(CY) ← (CY)	CY	comlement carry
IN	A,data	DB	11011011	2	10	(A) ← (V/V,B2)		input
OUT	A,data	D3	11010011	2	10	(V/V,B2) ← (A)		output
EI		FB	11111011	1	4			enable interrupt
DI		F3	11110011	1	4			disable interrupt
HLT		76	01110110	1	7			halt
NDP		00	00000000	1	4			no operation

Tabulka 2.10 Instrukční soubor 8080A - zvláštní instrukce

Význam značek a zkratek z tabulek 2.5 až 2.10 :

( ) - obsah specifikovaného registru nebo paměťového místa
← - přesun ve směru šipky
↓ - přenos z čítače instrukcí
↑ - přenos do čítače instrukcí
∧ - logický součin (AND)
∨ - logický součet (OR)
∀ - exklusivní logický součet (EXCLUSIVE OR)
+ - sčítání
- - dekrementace o hodnotu druhého komponentu
• - násobení
↔ - výměna (např. dvou obsahů registrů)
$\overline{X}$ - negace
MSB - nejvýznamnější bit
LSB - nejméně významný bit
PC - čítač instrukcí
SP - ukazatel zásobníku
M - paměťové místo

A - střadač
adr. - 16-bitová adresa
data - 8-bitová slabika dat
data 16 - dvě 8-bitové slabiky dat (16-ti bitová informace)
B2 - druhá slabika (Byte 2)
B3 - třetí slabika (Byte 3)
port - 8 bitová adresa V/V
r, r1, r2 - jeden z registrů A, B, C, D, E, H, L
DDD - cílový registr (Destination)
SSS - zdrojový registr (Source)
rp - registrový pár
rh - vyšší registr registrového páru (High)
rl - nižší registr registrového páru (Low)
rn - specifikovaný bit registru (n = 0 ÷ 7)
CCC - skokové podmínky
NNN - adresový bit 4, 5 a 6 pevné adresy skoku

Hodnota DDD nebo SSS	Specifikovaný registr
1 1 1	A
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L

Tab. 2.11 Registry 8080A

Hodnota rp	Specifikovaný registrový pár
0 0	B C
0 1	D E
1 0	H L
1 1	SP

Tab. 2.12 Registrové páry 8080A

Hodnota CCC	Význam	Příznak
0 0 0	NZ - není nula	Z = 0
0 0 1	Z - nula	Z = 1
0 1 0	NC - není přenos	CY = 0
0 1 1	C - přenos	CY = 1
1 0 0	PO - parita lichá	P = 0
1 0 1	PE - parita sudá	P = 1
1 1 0	P - plus	S = 0
1 1 1	M - minus	S = 1

Tab. 2.13 Význam PSW

hexa	MNEM	hexa	MNEM	hexa	MNEM
00	NOP	2B	DCX H&L	56	MOV D,M
01	LXI B&C	2C	INR L	57	MOV D,A
02	STAX B&C,M	2D	DCR L	58	MOV E,B
03	INX B&C	2E	MVI L,data	59	MOV E,C
04	INR B	2F	CMA	5A	MOV E,D
05	DCR B	30	-	5B	MOV E,E
06	MVI B,data	31	LXI SP	5C	MOV E,H
07	RLC	32	STA	5D	MOV E,L
0B	-	33	INX SP	5E	MOV E,M
09	DAD B	34	INR M	5F	MOV E,A
0A	LDAX B&C,M	35	DCR M	60	MOV H,B
0B	DCX B&C	36	MVI M,data	61	MOV H,C
0C	INR C	37	STC	62	MOV H,D
0D	DCR C	38	-	63	MOV H,E
0E	MVI C,data	39	DAD SP	64	MOV H,H
0F	RRC	3A	LDA addr.	65	MOV H,L
10	-	3B	DCX SP	66	MOV H,M
11	LXI D&E	3C	INR A	67	MOV H,A
12	STAX D&E,M	3D	DCR A	6B	MOV L,B
13	INX D&E	3E	MVI A,data	69	MOV L,C
14	INR D	3F	CMC	6A	MOV L,D
15	DCR D	40	MOV B,B	6B	MOV L,E
16	MVI D,data	41	MOV B,C	6C	MOV L,H
17	RAL	42	MOV B,D	6D	MOV L,L
1B	-	43	MOV B,E	6E	MOV L,M
19	DAD D	44	MOV B,H	6F	MOV L,A
1A	LDAX D&E,M	45	MOV B,L	70	MOV M,B
1B	DCX D&E	46	MOV B,M	71	MOV M,C
1C	INR E	47	MOV B,A	72	MOV M,D
1D	DCR E	48	MOV C,B	73	MOV M,E
1E	MVI E,data	49	MOV C,C	74	MOV M,H
1F	RAR	4A	MOV C,D	75	MOV M,L
20	-	4B	MOV C,E	76	HLT
21	LXI H&L	4C	MOV C,H	77	MOV M,A
22	SHLD H&L,addr.	4D	MOV C,L	78	MOV A,B
23	INX H&L	4E	MOV C,M	79	MOV A,C
24	INR H	4F	MOV C,A	7A	MOV A,D
25	DCR H	50	MOV D,B	7B	MOV A,E
26	MVI H,data	51	MOV D,C	7C	MOV A,H
27	DAA	52	MOV D,D	7D	MOV A,L
2B	-	53	MOV D,E	7E	MOV A,M
29	DAD H	54	MOV D,H	7F	MOV A,A
2A	LHLD H&L,data 16	55	MOV D,L	80	ADD B

Tabulka 2.14 Instrukční soubor B080A

hexa	MNEM		hexa	MNEM		hexa	MNEM	
81	ADD	C	AC	XRA	H	D7	RST	2
82	ADD	D	AD	XRA	L	D8	RC	
83	ADD	E	AE	XRA	M	D9	-	
84	ADD	H	AF	XRA	A	DA	JC	addr.
B5	ADD	L	B0	ORA	B	DB	IN	A,port
86	ADD	M	B1	ORA	C	DC	CC	addr.
87	ADD	A	B2	ORA	D	DD	-	
88	ADC	B	B3	ORA	E	DE	SBI	A,data
89	ADC	C	B4	ORA	H	DF	RST	3
8A	ADC	D	B5	ORA	L	E0	RPO	
8B	ADC	E	B6	ORA	M	E1	POP	H
8C	ADC	H	B7	ORA	A	E2	JPO	addr.
8D	ADC	L	BB	CMP	B	E3	XTHL	
8E	ADC	M	B9	CMP	C	E4	CPO	addr.
8F	ADC	A	BA	CMP	D	E5	PUSH	H
90	SUB	B	BB	CMP	E	E6	ANI	A,data
91	SUB	C	BC	CMP	H	E7	RST	4
92	SUB	D	BD	CMP	L	E8	RPE	
93	SUB	E	BE	CMP	M	E9	PCHL	
94	SUB	H	BF	CMP	A	EA	JPE	addr.
95	SUB	L	CD	RNZ		EB	XCHG	
96	SUB	M	C1	POP	B	EC	CPE	addr.
97	SUB	A	C2	JNZ	addr.	ED	-	
98	SBB	B	C3	JMP	addr.	EE	XRI	A,data
99	SBB	C	C4	CNZ	addr.	EF	RST	5
9A	SBB	D	C5	PUSH	B	FD	RP	
9B	SBB	E	C6	ADI	A,data	F1	POP	PSW
9C	SBB	H	C7	RST	0	F2	JP	addr.
9D	SBB	L	C8	RZ		F3	DI	
9E	SBB	M	C9	RET		F4	CP	addr.
9F	SBB	A	CA	JZ	addr.	F5	PUSH	PSW
A0	ANA	B	CB	-		F6	ORI	A,data
A1	ANA	C	CC	CZ	addr.	F7	RST	6
A2	ANA	D	CD	CALL	addr.	F8	RM	
A3	ANA	E	CE	ACI	A,data	F9	SPHL	
A4	ANA	H	CF	RST	1	FA	JM	addr.
A5	ANA	L	D0	RNC		FB	EI	
A6	ANA	M	D1	POP	D	FC	CM	addr.
A7	ANA	A	D2	JNC	addr.	FD	-	
A8	XRA	B	D3	OUT	A,port	FE	CPI	Adata
A9	XRA	C	D4	CNC	addr.	FF	RST	7
AA	XRA	D	D5	PUSH	D			
AE	XRA	E	D6	SUI	A,data			

Tab. 2.15 Instrukční soubor 8080A

MNEM		B	H	F	MNEM		B	H	F
ACI	dB	2	CE	1	DCR	D	1	15	1
ADC	A	1	8F	1	DCR	E	1	1D	1
ADC	8	1	8B	1	DCR	H	1	25	1
ADC	C	1	89	1	DCR	L	1	2D	1
ADC	D	1	8A	1	DCR	M	1	35	1
ADC	E	1	BB	1	DCX	B	1	0B	0
ADC	H	1	8C	1	DCX	D	1	1B	0
ADC	L	1	8D	1	DCX	H	1	2B	0
ADC	M	1	8E	1	DCX	SP	1	3B	0
ADD	A	1	87	1	DI		1	F3	0
ADD	B	1	B0	1	EI		1	FB	0
ADD	C	1	81	1	HLT		1	76	0
ADD	D	1	82	1	IN	aB	2	DB	0
ADD	E	1	83	1	INR	A	1	3C	1
ADD	H	1	84	1	INR	B	1	04	1
ADD	L	1	85	1	INR	C	1	0C	1
ADD	M	1	86	1	INR	D	1	14	1
ADI	d8	2	C6	1	INR	E	1	1C	1
ANA	A	1	A7	1	INR	H	1	24	1
ANA	B	1	A0	1	INR	L	1	2C	1
ANA	C	1	A1	1	INR	M	1	34	1
ANA	D	1	A2	1	INX	B	1	03	0
ANA	E	1	A3	1	INX	D	1	13	0
ANA	H	1	A4	1	INX	H	1	23	0
ANA	L	1	A5	1	INX	SP	1	33	0
ANA	M	1	A6	1	JC	a16	3	DA	0
ANI	d8	2	E6	1	JM	a16	3	FA	0
CALL	a16	3	CD	0	JMP	a16	3	C3	0
CC	a16	3	DC	0	JNC	a16	3	D2	0
CM	a16	3	FC	0	JNZ	a16	3	C2	0
CMA		1	2F	0	JP	a16	3	F2	0
CMC		1	3F	1	JPE	a16	3	EA	0
CMP	A	1	BF	1	JPO	a16	3	E2	0
CMP	B	1	BB	1	JZ	a16	3	CA	0
CMP	C	1	B9	1	LDA	a16	3	3A	0
CMP	D	1	BA	1	LDAX	B	1	DA	0
CMP	E	1	BB	1	LDAX	D	1	1A	0
CMP	H	1	BC	1	LHLD	a16	3	2A	0
CMP	L	1	BD	1	LXI	B,d16	3	01	0
CMP	M	1	8E	1	LXI	D,d16	3	11	0
CNC	a16	3	D4	0	LXI	H,d16	3	21	0
CNZ	a16	3	C4	0	LXI	SP,d16	3	31	0
CP	a16	3	F4	0	MOV	A,A	1	7F	0
CPE	a16	3	EC	0	MOV	A,B	1	78	0
CPI	dB	2	FE	0	MOV	A,C	1	79	0
CPO	a16	3	E4	0	MOV	A,D	1	7A	0
CZ	a16	3	CC	0	MOV	A,E	1	7B	0
DAA		1	27	1	MOV	A,H	1	7C	0
DAD	B	1	09	1	MOV	A,L	1	7D	0

MNEM		B	H	F	MNEM		B	H	F
DAD	D	1	19	1	MOV	A,M	1	7E	0
DAD	H	1	29	1	MOV	B,A	1	47	0
DAD	SP	1	39	1	MOV	B,B	1	40	0
DCR	A	1	3D	1	MOV	B,C	1	41	0
DCR	B	1	05	1	MOV	B,D	1	42	0
DCR	C	1	0D	1	MOV	B,E	1	43	0

Poznámky : B = počet slabik
H = první slabika hexa kódu
F = ovlivnění příznaků
(0 = ne , 1 = ano)

d8 = 8bitové slovo
d16 = 16bitové datové slovo
a8 = 8bitová adresa (V/V)
a16 = 16bitová adresa (M)

Tabulka 2.16 Instrukční soubor 8080A

MNEM		B	H	F	MNEM		B	H	F
MOV	B,H	1	44	0	ORI	d8	2	F6	1
MOV	B,L	1	45	0	OUT	a8	2	D3	0
MOV	B,M	1	46	0	PCHL		1	E9	0
MOV	C,A	1	4F	0	POP	B	1	C1	0
MOV	C,B	1	48	0	POP	D	1	D1	0
MOV	C,C	1	49	0	POP	H	1	E1	0
MOV	C,D	1	4A	0	POP	PSW	1	F1	1
MOV	C,E	1	4B	0	PUSH	B	1	C5	0
MOV	C,H	1	4C	0	PUSH	D	1	D5	0
MOV	C,L	1	4D	0	PUSH	H	1	E5	0
MOV	C,M	1	4E	0	PUSH	PSW	1	F5	1
MOV	D,A	1	57	0	RAL		1	17	1
MOV	D,B	1	50	0	RAR		1	1F	1
MOV	D,C	1	51	0	RC		1	DB	0
MOV	D,D	1	52	0	RET		1	C9	0
MOV	D,E	1	53	0	RLC		1	07	1
MOV	D,H	1	54	0	RM		1	F8	0
MOV	D,L	1	55	0	RNC		1	D0	0
MOV	D,M	1	56	0	RNZ		1	C0	0
MOV	E,A	1	5F	0	RP		1	F0	0
MOV	E,B	1	5B	0	RPE		1	E8	0
MOV	E,C	1	59	0	RPO		1	E0	0
MOV	E,D	1	5A	0	RRC		1	0F	0
MOV	E,E	1	5B	0	RST	0	1	C7	0
MOV	E,H	1	5C	0	RST	1	1	CF	0
MOV	E,L	1	5D	0	RST	2	1	D7	0
MOV	E,M	1	5E	0	RST	3	1	DF	0
MOV	H,A	1	67	0	RST	4	1	E7	0
MOV	H,B	1	60	0	RST	5	1	EF	0
MOV	H,C	1	61	0	RST	6	1	F7	0
MOV	H,D	1	62	0	RST	7	1	FF	0

MNEM		B	H	F	MNEM		B	H	F
MOV	H,E	1	63	0	RZ		1	C8	0
MOV	H,H	1	64	0	SBB	A	1	9F	1
MOV	H,L	1	65	0	SBB	B	1	98	1
MOV	H,M	1	66	0	SBB	C	1	99	1
MOV	L,A	1	6F	0	SBB	D	1	9A	1
MOV	L,B	1	6B	0	SBB	E	1	9B	1
MOV	L,C	1	69	0	SBB	H	1	9C	1
MOV	L,D	1	6A	0	SBB	L	1	9D	1
MOV	L,E	1	6B	0	SBB	M	1	9E	1
MOV	L,H	1	6C	0	SBI	d8	2	DE	1
MOV	L,L	1	6D	0	SHLD	a16	3	22	0
MOV	L,M	1	6E	0	SPHL		1	F9	0
MOV	M,A	1	77	0	STA	a16	3	32	0
MOV	M,3	1	70	0	STAX	B	1	02	0
MOV	M,C	1	71	0	STAX	D	1	12	0
MOV	M,D	1	72	0	STC		1	37	1
MOV	M,E	1	73	0	SUB	A	1	97	1
MOV	M,H	1	74	0	SUB	B	1	90	1
MOV	M,L	1	75	1	SUB	C	1	91	1
MVI	A,d8	2	3E	1	SUB	D	1	92	1
MVI	B,d8	2	06	1	SUB	E	1	93	1
MVI	C,d8	2	0E	1	SUB	H	1	94	1
MVI	D,d8	2	16	1	SUB	L	1	95	1
MVI	E,d8	2	1E	1	SUB	M	1	96	1
MVI	H,d8	2	26	1	SUI	dB	2	D6	1
MVI	L,d8	2	2E	1	XCHG		1	EB	0
MVI	M,dB	2	36	0	XRA	A	1	AF	1
NOP		1	00	0	XRA	B	1	A8	1
ORA	A	1	B7	1	XRA	C	1	A9	1
ORA	B	1	B0	1	XRA	D	1	AA	1
ORA	C	1	B1	1	XRA	E	1	AB	1
ORA	D	1	B2	1	XRA	H	1	AC	1
ORA	E	1	B3	1	XRA	L	1	AD	1
ORA	H	1	B4	1	XRA	M	1	AE	1
ORA	L	1	B5	1	XRI	d8	2	EE	1
ORA	M	1	B6	1	XTHL		1	E3	1

Poznámky : B = počet slabik
H = první slabika hexa kódu
F = ovlivnění příznaků
(0 = ne , 1 = ano)

d8 = 8bitové slovo
d16 = 16bitové datové slovo
aB = 8bitová adresa (V/V)
a16 = 16bitová adresa (M)

Tabulka 2.17 Instrukční soubor 8080A

MNEM	Kód								Dob
	D7	D6	D5	D4	D3	D2	D1	D0	
ACI	1	1	0	0	1	1	1	0	7
ADC M	1	0	0	0	1	1	1	0	7
ADC r	1	D	0	D	1	S	S	S	4
ADD M	1	0	0	D	0	1	0	1	7
ADD r	1	0	0	D	0	S	S	S	4
ADI	1	1	0	0	0	1	1	0	7
ANA M	1	0	1	0	0	1	1	0	7
ANA r	1	0	1	0	0	S	S	S	4
ANI	1	1	1	0	0	1	1	0	7
CALL	1	1	0	0	1	1	0	1	17
CC	1	1	0	1	1	1	0	0	11/17
CM	1	1	1	1	1	1	0	0	11/17
CMA	0	0	1	0	1	1	1	1	4
CMC	0	0	1	1	1	1	1	1	4
CMP M	1	0	1	1	1	1	1	0	7
CMP r	1	0	1	1	1	S	S	S	4
CNC	1	1	0	1	0	1	0	0	11/17
CNZ	1	1	0	0	0	1	0	0	11/17
CP	1	1	1	1	0	1	0	0	11/17
CPE	1	1	1	0	1	1	0	0	11/17
CPI	1	1	1	1	1	1	1	0	7
CPO	1	1	1	0	0	1	0	0	11/17
CZ	1	1	0	0	1	1	0	0	11/17
DAA	0	0	1	0	0	1	1	1	4
DAD B	0	0	0	0	1	0	0	1	10
DAD D	0	0	0	1	1	0	0	1	10
DAD H	0	0	1	0	1	0	0	1	10
DAD SP	0	0	1	1	1	0	0	1	10
DCR M	0	0	1	1	0	1	0	1	10
DCR r	0	0	D	D	D	1	0	1	5
DCX B	0	0	0	0	1	0	1	1	5
DCX D	0	0	0	1	1	0	1	1	5
DCX H	0	0	1	0	1	0	1	1	5
DCX SP	0	0	1	1	1	0	1	1	5
DI	1	1	1	1	0	0	1	1	4
EI	1	1	1	1	1	0	1	1	4
HLT	0	1	1	1	0	1	1	0	7
IN	1	1	0	1	1	0	1	1	10
INR M	0	0	1	1	0	1	0	0	10
INR r	0	0	D	D	D	1	0	0	5
INX B	0	0	0	0	0	0	1	1	5
INX D	0	0	0	1	0	0	1	1	5
INX H	0	0	1	0	0	0	1	1	5
INX SP	0	0	1	1	0	0	1	1	5
JC	1	1	0	1	1	0	1	0	10
JM	1	1	1	1	1	0	1	0	10
JMP	1	1	0	0	0	0	1	1	10

MNEM	Kód								Dob
	D7	D6	D5	D4	D3	D2	D1	D0	
JNC	1	1	0	1	0	0	1	0	10
JNZ	1	1	0	0	0	0	1	0	10
JP	1	1	1	1	0	0	1	0	10
JPE	1	1	1	0	1	0	1	0	10
JPO	1	1	1	0	0	0	1	0	10
JZ	1	1	0	0	1	0	1	0	10
LDA	0	0	1	1	1	0	1	0	13
LDAX B	0	0	0	0	1	0	1	0	7
LDAX D	0	0	0	1	1	0	1	0	7
LHLD	0	0	1	0	1	0	1	0	16
LXI B	0	0	0	0	0	0	0	1	10
LXI D	0	0	0	1	0	0	0	1	10
LXI H	0	0	1	0	0	0	0	1	10
LXI SP	0	0	1	1	0	0	0	1	10
MVI M	0	0	1	1	0	1	1	0	10
MVI r	0	0	D	D	D	1	1	0	7
MOV M,r	0	1	1	1	0	S	S	S	7
MOV r,M	0	1	D	D	D	1	1	0	7
MOV r1,r2	0	1	D	D	D	S	S	S	5
NOP	0	0	0	0	0	0	0	0	4
ORA M	1	0	1	1	0	1	1	0	7
ORA r	1	0	1	1	0	S	S	S	4
ORI	1	1	1	1	0	1	1	0	7
OUT	1	1	0	1	0	0	1	1	10
PCHL	1	1	1	0	1	0	0	1	5
POP B	1	1	0	0	0	0	0	1	10
POP D	1	1	0	1	0	0	0	1	10
POP H	1	1	1	0	0	0	0	1	10
POP PSW	1	1	1	1	0	0	0	1	10
PUSH B	1	1	0	0	0	1	0	1	11
PUSH D	1	1	0	1	0	1	0	1	11
PUSH H	1	1	1	0	0	1	0	1	11
PUSH PSW	1	1	1	1	0	1	0	1	11
RAL	0	0	0	1	0	1	1	1	4
RAR	0	0	0	1	1	1	1	1	4
RC	1	1	0	1	1	0	0	0	5/11
RET	1	1	0	0	1	0	0	1	10
RCL	0	0	0	0	0	1	1	1	4
RM	1	1	1	1	1	0	0	0	5/11
RNC	1	1	0	1	0	0	0	0	5/11
RNZ	1	1	0	0	0	0	0	0	5/11
RP	1	1	1	1	0	0	0	0	5/11
RPE	1	1	1	0	1	0	0	0	5/11
RPO	1	1	1	0	0	0	0	0	5/11
RRC	0	0	0	0	1	1	1	1	4
RST	1	1	A	A	A	1	1	1	11

MNEM	Kód								Dob
	D7	D6	D5	D4	D3	D2	D1	D0	
RZ	1	1	0	0	1	0	0	0	5/11
SBB M	1	0	0	1	1	1	1	0	7
SBB r	1	0	0	1	1	S	S	S	4
SBI	1	1	0	1	1	1	1	0	7
SHLD	0	0	1	0	0	0	1	0	16
SPHL	1	1	1	1	1	0	0	1	5
STA	0	0	1	1	0	0	1	0	13
STAX B	0	0	0	0	0	0	1	0	7
STAX D	0	0	0	1	0	0	1	0	7
STC	0	0	1	1	0	1	1	1	4
SUB M	1	0	0	1	0	1	1	0	7
SUB r	1	0	0	1	0	S	S	S	4
SUI	1	1	0	1	0	1	1	0	7
XCHG	1	1	1	0	1	0	1	1	4
XRA M	1	0	1	0	1	1	1	0	7
XRA r	1	0	1	0	1	S	S	S	4
XRI	1	1	1	0	1	1	1	0	7
XTHL	1	1	1	0	0	0	1	1	18

Tabulka 2.18 Instrukční soubor 8080A - kódování instrukcí

zen pojem instrukce. V dalším se proto přidržíme rozdělení tak, jak je uvádí výrobce.

Instrukční soubor můžeme rozdělit na instrukce:

1. pro přesuny dat (přesunové)
2. aritmetické, logické, posunové a jiné
3. skokové (pro větvení programu)
4. pro komunikaci s V/V
5. řídicí a zvláštní instrukce

Z těchto hledisek jsou instrukce rozděleny v tabulkách 2.5 až 2.13. Tabulky 2.17 a 2.18 uvádějí instrukční soubor dle hexa kódu a tabulky 2.19 a 2.20 instrukce seřazené dle abecedy.

2.8 Mikropočítačová stavebnice MCS-80

Již první mikroprocesory byly nabízeny především ve formě stavebnic. Tato situace trvá i v současné době. Např. firma Intel nabízí stavebnici MCS-48/80/85, která zahrnuje pomocné součástky pro výstavbu mikropočítačů s osmibitovými mikroprocesory 8080, 8085, 8048/8748/8035 a 8041/8741.

V dalším si všimneme základní stavebnice pro mikroprocesor 8080A s označením MCS-80. Mikropočítačová stavebnice zahrnuje nejenom vlastní mikroprocesor 8080A s podpůrnými obvody (tzv. skupina mikroprocesoru), ale zejména paměti ROM, EPROM, PROM, RMW a obvody vstup/výstup. Minimální sestavu mikropočítače tvoří mikroprocesor s podpůrnými obvody, paměti a některý z obvodů seriového nebo paralelního vstupu/výstupu. Typickou sestavu mikropočítače ukazuje obr. 2.4.

2.8.1 Skupina mikroprocesoru

Skupinu mikroprocesoru tvoří mikroprocesor 8080A spolu s obvody 8224 a 8228. Mikroprocesor 8080A byl již v předchozím podrobně charakterizován, a proto se zaměříme na ostatní prvky souboru.

2.8.1.1 Generátor hodinového signálu 8224

Integrovaný obvod 8224 představuje generátor dvoufázového hodinového signálu a obsahuje i budiče, které zajišťují dosažení nestandardních napěťových úrovní signálu, jak je požaduje mikroprocesor 8080A (viz obr. 2.5).

Pro činnost obvodu 8224 je nezbytné k němu připojit vnější dvojpól, určující kmitočet hodin. Může to být člen RC nebo rezonátor typu LC, ale obvykle to bývá miniaturní piezoelektrický rezonátor. Vyžaduje se na něm seriová rezonance na devítinásobku požadovaného hodinového kmitočtu a ekvivalentní odpor v rozmezí 20 až 75 Ohmů. Maximální kmitočet krystalu může být až 27 MHz. Nejčastěji se používá křemenný krystal s kmitočtem asi 18 MHz.

Při připojení rezonátoru začne v obvodu 8224 kmitat oscilátor a získaný signál se zavádí jednak na výstup OSC, jednak do vnitřního generátoru hodin. Tento generátor dělí základní kmitočet devíti a tvaruje dvě posloupnosti nepřekrývajících se impulsů Ø 1 a Ø 2. Dolní hodnota napětí hodinového signálu je nejvýše 0,45V a horní hodnota nejméně 9,4V. Kromě toho obvod 8224 dodává na výstup signál Ø2 (TTL) se standardními úrovněmi TTL. Máme tedy na výstupech čtyři signály, z nichž Ø1 a Ø2 se zavádějí na hodinové vstupy mikroprocesoru 8080 a signály OSC a Ø2 (TTL) se mohou použít pro synchronizaci procesů v mikropočítači (viz obr. 2.5).

Dále má obvod 8224 tři výstupy označené $\overline{STSTB}$, RESET a READY, které se používají k řízení vzorkování stavu mikroprocesoru v době T2 (Status Strobe), k zahájení činnosti (Reset) a k zavádění čekacího stavu Tw (Ready). Generování těchto signálů závisí na vstupních signálech SYNC (z 8080A), $\overline{RESIN}$ a RDYIN a je uvnitř 8224 synchronizováno hodinovým signálem.

2.8.1.2 Řídicí obvod sběrnice 8228 (System Controller)

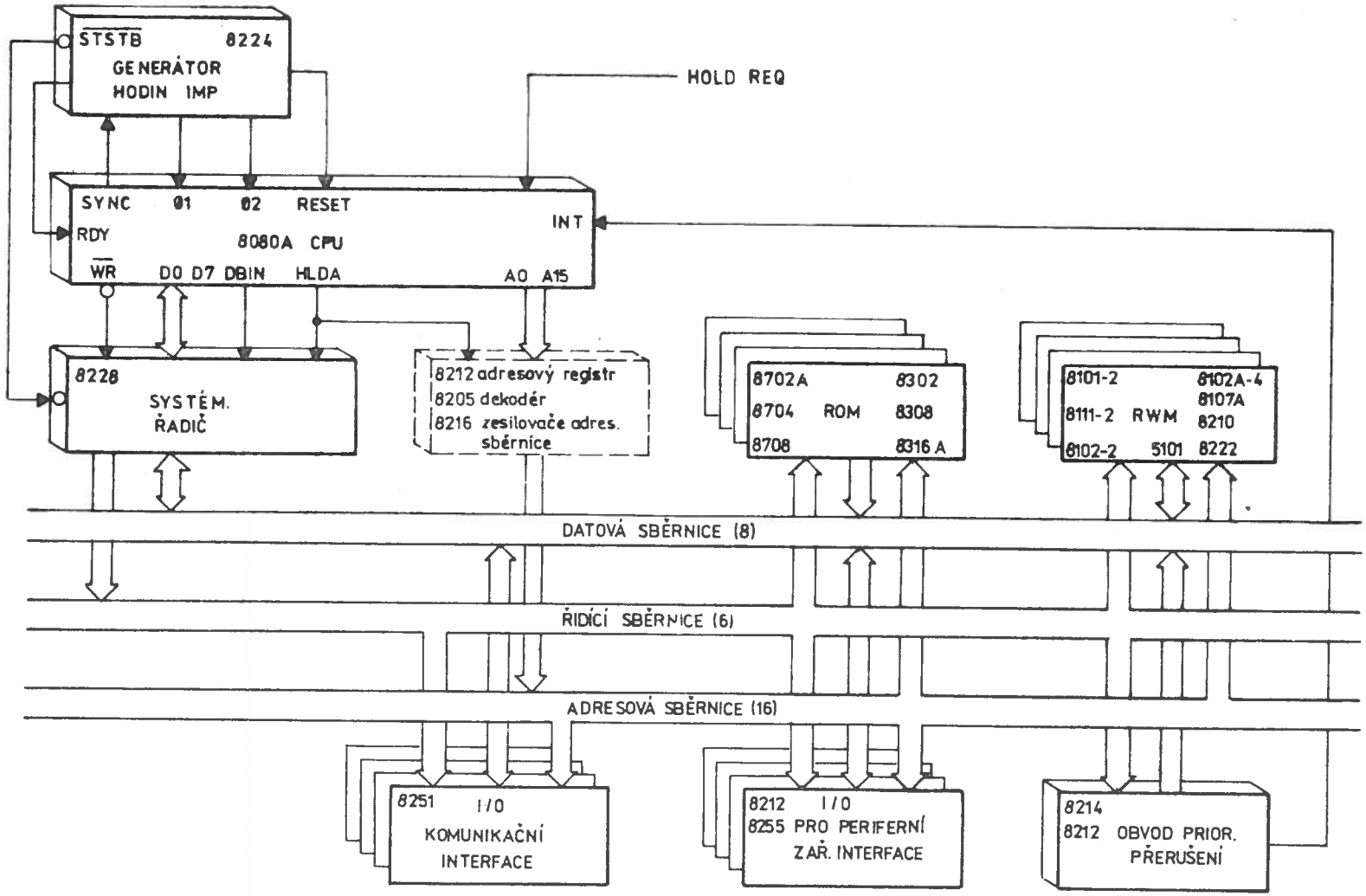
Obvod 8228 (8238) je kombinovaný budič systémové sběrnice a budič sběrnice automatů s mikroprocesorem 8080A (viz obr. 2.5).

Obsahuje dvě části:

a) osmibitový oboustranný budič sběrnice

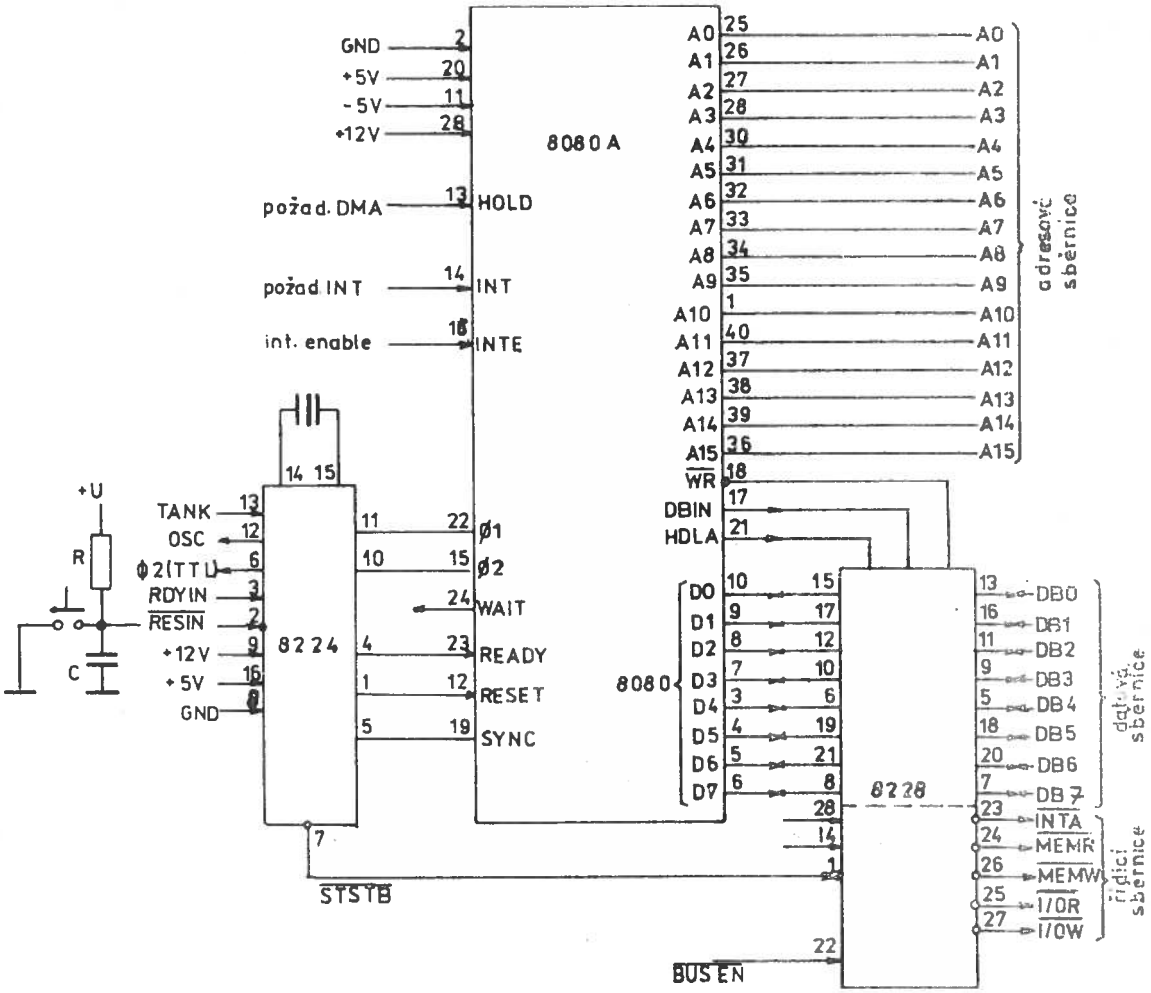
b) kombinovaný dekodér taktu.

V době T1 každého taktu mikroprocesor 8080A na datovou sběrnici vysílá stavové slovo, ve kterém je zakódována informace o typu taktu. Stavové slovo se v obvodech



Obr. 2.4 Blokové schéma typického mikropočítačového systému s mikroprocesorem 8080A

typu 8228/8238 zavádí do stavového registru a po příchodu signálu $\overline{STSTB}$ z obvodu 8224 se dekóduje. Na výstupu dekodéru, resp. na "řídicí sběrnici" pak máme 5 signálů. Jsou to 2 signály pro řízení zápisu do paměti nebo výstupu ($\overline{MEMW}$ a $\overline{IOW}$), 2 signály pro řízení čtení z paměti nebo vstupu ($\overline{MEMR}$ a $\overline{IOR}$) a 1 signál ($\overline{INTA}$), který oznamuje přijetí požadavku na přerušení mikroprocesorem (viz obr. 2.5 a 2.6).



Obr. 2.5 Zapojení obvodů skupiny mikroprocesoru

Zesilovače v obvodech 8228/8238 postačí k buzení sběrnic malých a středních automatů s 8080A. V rozsáhlejších aplikacích je nutné sběrnice posílit budiči typu 8216 nebo 8226.

Obvod 8238 se od obvodu 8228 liší jen tím, že impulsy $\overline{IOW}$ a $\overline{MEMW}$ dodává na řídicí sběrnici přibližně o 1 dobu dříve. Tím usnadňuje zavádění čekacího stavu Wait (Tw) v případě velkých systémů, kdy rozsáhlá datová sběrnice je vlivem kapacitních zátěží příliš pomalá.

Na obr. 2.5 si povšimneme i obvodů pro nastavení mikroprocesoru do nulté polohy při připojení napájecího napětí, resp. při ručním nulování. Tyto obvody tvoří člen RC připojený na vstup RESIN, resp. tlačítko.

Další možnost vytvoření řídicích signálů ukazuje obr. 2.6, kde je použito registru 8212 a obvodů nízkého stupně integrace.

2.8.2 Paměti

Ve stavebnici je obsažena řada statických i dynamických pamětí RWM . Dynamické paměti je nutno v předepaných časových intervalech obnovovat, a proto stavebnice obsahuje i obvod 8202 - řadič zotavení paměti RWM.

Součástí stavebnice jsou dále paměti ROM (PROM, EPROM). Řízením vlastní paměti ani jednotlivými typy pamětí se zde nebudeme zabývat vzhledem k tomu, že lze použít prakticky libovolných typů , a proto odkazujeme případného zájemce na lit. /12/, /18/, /25/, /28/. Připomeňme pouze, že k aktivování paměti využijeme výběrové signály paměti (CHIP SELECT). Řízení čtení, případně čtení/zápis, bude aktivováno signály $\overline{MEMR}$ a $\overline{MEMW}$. Pro dekódování adresy pamětí i vstupních/výstupních bran se používá řada dalších obvodů. Mohou to být běžné obvody nižšího stupně integrace (hradla) nebo přímo specializované obvody. Typickým představitelem je např. dekodér 8205 (3205) - rychlý dekodér 1 z 8 s malým vstupním proudem a vysokým logickým ziskem.

2.8.3 Obousměrný zesilovač datové sběrnice a logika řízení systému

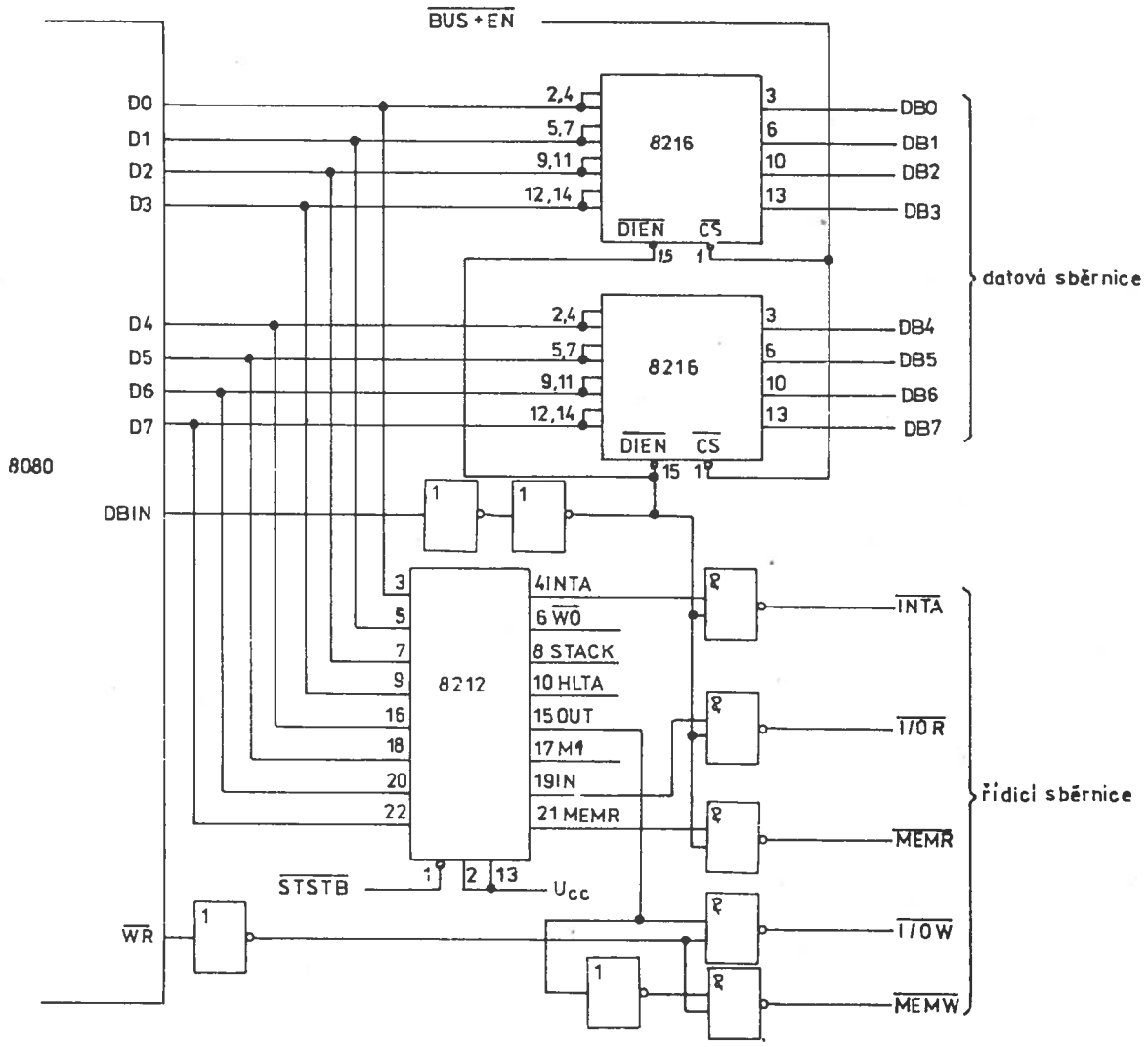
Paměťové jednotky systému a jednotky V/V jsou propojené s mikroproccesorem 8080A přes obousměrnou datovou sběrnici a sběrnici řídicí. Řídicí sběrnice se používá k tomu, aby byla přenášena data ve správném časovém okamžiku. Datové vodiče, paměti a členy V/V mají tři možné stavy. Selektivním připojováním nebo odpojováním (třístavová logika) paměti nebo V/V jednotek pomocí signálů z řídicí sběrnice lze jednoduchým způsobem přenášet informace na nebo ze sběrnice.

Odvození řídicích signálů sběrnice a její řízení již bylo uvedeno v odst. 2.8.1.2 v popisu řídicího obvodu sběrnice 8228. Zde pouze připomeneme, že datová sběrnice mikroprocesoru 8080A má požadavek na minimální výstupní napětí úrovně log. 1 (3,3V) a povoluje maximální odběr pouze 1,7mA. Těmto hodnotám však neodpovídají vstupní a výstupní obvody většiny pamětí a periférních zařízení. Přizpůsobení lze řešit (viz obr. 2.5 a obr. 2.6) pomocí obvodů 8216 nebo (pro menší systémy) přímo obvodem 8228. V dalším tyto obvody stručně charakterizujeme.

2.8.3.1 Obvody 8216 a 8226

Tyto obvody (totožné s označením 3216 a 3226) jsou vyrobeny technologií STTL. Obvod 8216 (3216) slouží jako třístavový neinvertující obousměrný budič sběrnice s nízkým vstupním proudem, velkým budícím ziskem a malým zpožděním signálu. Obvod 8226 (3226) je jinak shodný, ale invertující budič.

Obvody jsou schopny zajistit požadované napěťové a proudové úrovně i pro výstavbu rozsáhlých mikropočítačových systémů.



Obr. 2.6 Řízení systému 8080A

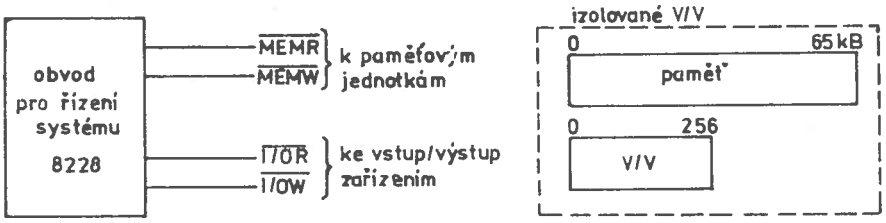
2.8.4 Obvody vstup/výstup

Stejně jako centrální procesor každého výpočetního systému, tak i mikroprocesor 8080A musí mít možnost styku s prvky nebo systémy mimo vlastní výpočetní obvody. Přístroje jako děrovače pásky, klávesnice, flopy - disky, tiskárny, převodníky A/D a D/A a jiné se používají pro zadávání nebo zobrazení informací mikropočítače. Výsledky operací se potom ukládají buď do paměti a nebo jsou přímo předávány periferii.Pro tyto účely se používají obvody vstup/výstup. Existuje řada možností volby uspořádání vstupů a výstupů celého systému tak, aby bylo dosaženo co největších výkonů při co nejnižších nárocích na obvody. V podstatě však jsou dvě základní možnosti zapojení obvodů vstup/výstup do mikropočítačového systému:

-Izolované vstupy/výstupy

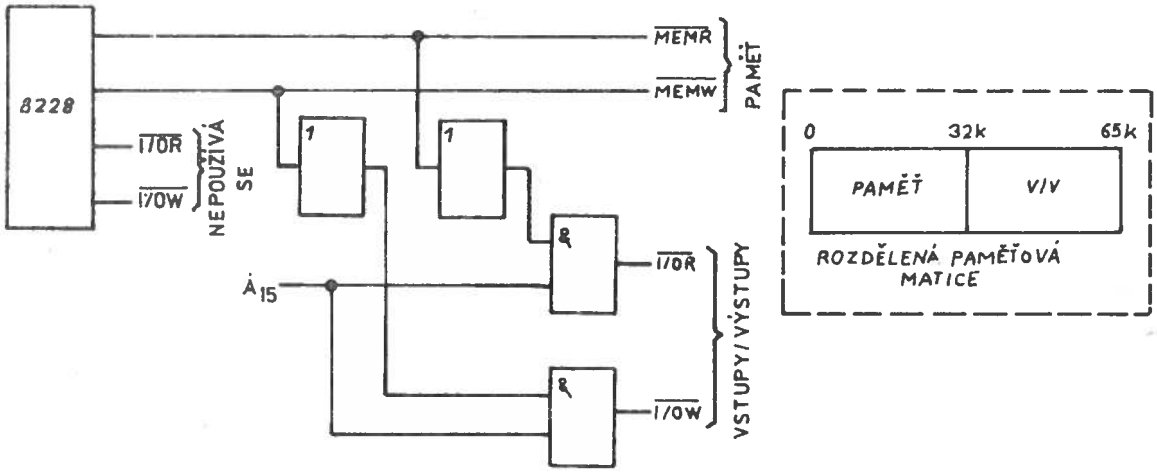
-Paměťově mapované vstupy/výstupy

V prvém případě **-izolované V/V-** ("Isolated I/O")jsou vstupy a výstupy zapojeny dle obr 2.7. Pro řízení přenosu (IN, OUT) má 8080A zvláštní instrukce, které rozlišují od sebe paměťové jednotky a jednotky V/V, takže nemohou být chybně adresována místa v paměti místo jednotek V/V. V úplné sestavě systému má paměť kapacitu 65 kB a na její osmibitové buňky se lze odvolávat adresami v rozsahu 0 až 65535 (adresy jsou obsaženy v šestnáctibitových slovech). Adresový prostor vstupů/výstupů má rozsah 256 B a lze se tedy na něj odvolat adresami v rozsahu 0 až 255. V tomto případě (instrukce IN a OUT) mají obvody V/V přístup **pouze** ke střadači, ale celkový adresní prostor 65 kB není nijak omezen.



Obr.2.7 Izolované V/V

V druhém případě, při **paměťovém mapování V/V** (Memory Mapped I/O), rezervujeme (viz obr. 2.8) určitý rozsah adresního prostoru paměti pro obvody V/V. Tím lze ovšem tyto obvody ovládat stejnými instrukcemi, které se používají pro přístup do paměti. Jak je zřejmé z obr. 2.8, vznikají nové řídicí signály spojením signálů MEMR a MEMW na vhodnou kombinaci adresových vodičů. Pro tento účel se většinou se používá bit A15.



Obr. 2.8 Paměťové mapování obvodů V/V

Použijeme-li bit A 15 jako označení příznaku V/V, dostaneme jednoduché přiřazení: je-li A15 = 0 , je aktivní paměť,
je-li A15 = 1 , je aktivní obvod V/V.
Jak již bylo řečeno, pro tuto funkci lze použít i jiných adresovacích bitů. Použití bitu A15 je však výhodné, protože může být snadno řízen programem a ještě umožňuje adresování paměti do 32 K slabik.

Jinou metodou adresování obvodů V/V je tzv. **"lineární volba"** (Linear select). Při této metodě se nedekóduje adresová sběrnice, nýbrž každý jednotlivý bit aktivuje jeden obvod V/V. Při této metodě je omezený počet jednotek V/V, které mohou být adresované, ale odpadá dekodér, což je důležitý bod zejména při navrhování menších systémů.

Pro obvody V/V lze používat řadu prvků z nichž některé jsou programovatelné, tzn., že funkce obvodu není dána jen strukturou a zapojením, ale i tzv. řídicími slovy, která do jeho řídicího registru odesílá mikroprocesor. O některých těchto obvodech bude dále pojednáno v následujícím odstavci.

2.8.5 Doplňující obvody stavebnice MCS 80.

K těmto obvodům řadíme jak univerzální registry, např. 8212, tak plně programovatelné obvody sériového a paralelního styku 8255 a 8251 a dále programovatelné časovače 8253, řadič DMA 8257 a řadič přerušení 8259. Do této skupiny můžeme dále zařadit obvod 8271 - řadič styku s pružnými disky, obvod 8273 - řadič SDLC protokolu, obvod 8275 - řadič styku s obrazovkovým terminálem atd. Z tohoto sortimentu si povšimneme pouze obvodů nejčastěji užívaných, ostatní - s ohledem na jejich specifické použití nalezne čtenář v příslušné literatuře.

2.8.5.1 Univerzální registr 3212 (8212)

Obvod 3212 (8212) je univerzální paralelní osmibitový registr (Latch) s oddělovacími výstupními budiči a třístavovým výstupem a s doplňujícími obvody. Obsahuje 8 klopných obvodů typu D s oddělovacími třístavovými zesilovači na výstupu a doplňkové obvody s následujícími vstupy: 2 vstupy pro aktivaci (DS - Device Select), 1 vstup vzorkovací (STB-Strobe), 1 vstup pro řízení stavu výstupních zesilovačů a volbu zdroje hodinových impulsů klopných obvodů (MD) a 1 vstup pro nulování klopných obvodů (CLR).

Využití tohoto obvodu je zcela univerzální. Nejobvyklejší aplikace však jsou: hradlovaný oddělovací registr, vstupní-výstupní brána, jednosměrný nebo i obousměrný sběrnicový zesilovač (méně výkonný než s 8216, ale s registry). Bližší viz např. lit. /18/, /24/, /25/.

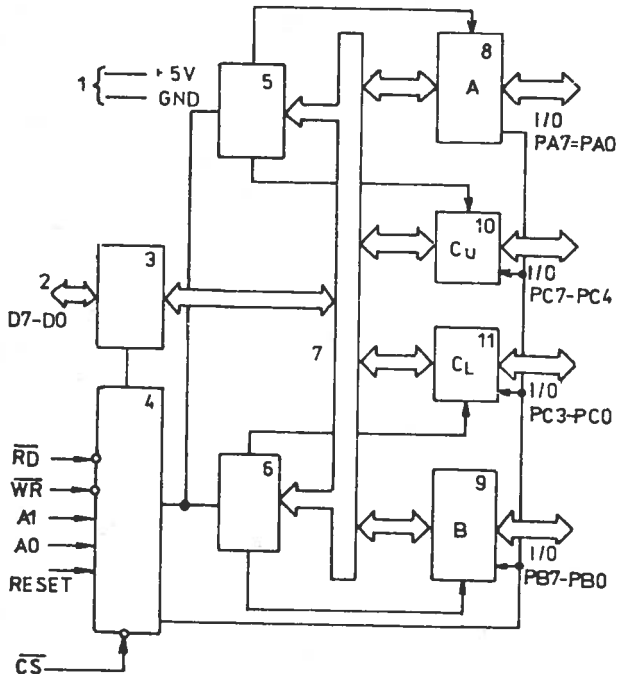
2.8.5.2 Programovatelný obvod 8255

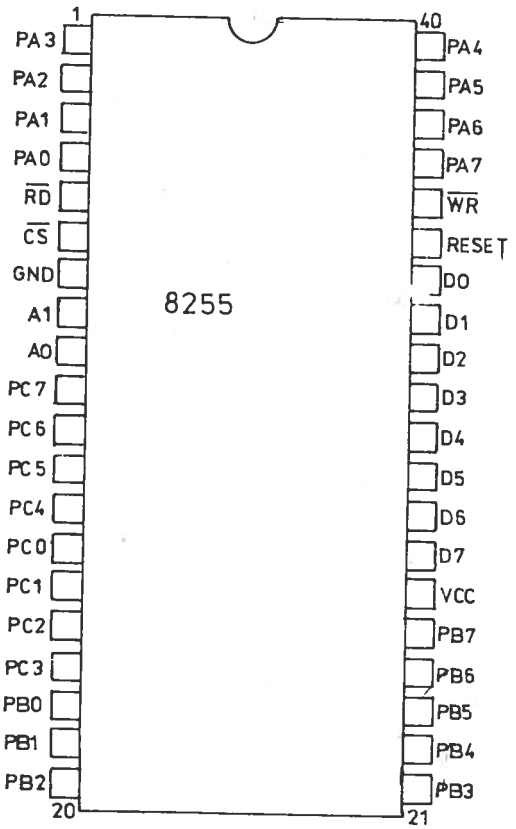
Obvod 8255 je programovatelný víceúčelový obvod pro připojení vstupů/výstupů k mikroprocesoru 8080A. Má 24 vstupů/výstupů, které mohou být naprogramované odděleně ve dvou skupinách po dvanácti a mohou pracovat ve třech režimech:

Při prvním druhu provozu (**režim** 0) může být každá skupina 12 vstupů/výstupů naprogramována po 4 jako vstup nebo výstup. Při druhém druhu provozu (**režim** 1) může být naprogramováno 8 vstupů/výstupů v každé skupině jako vstup nebo výstup. Ze zbývajících 4 vývodů se použijí 3 pro výměnu potvrzení přerušení a pro signály k řízení přerušení. Třetí druh provozu (**režim 2**) můžeme označit jako obousměrný provoz sběrnice, při kterém se používá pro jednu obousměrnou sběrnici 8 vývodů. 5 dalších vývodů, z nichž jeden patří už do druhé skupiny, se v tomto případě používá pro výměnu potvrzení přerušení. Kromě toho je možné přímé nastavení a nulování

jednotlivých bitů. Funkční vlastnosti obvodu 8255 jsou dány softwarem, takže pro připojení periferních zařízení nebo obvodů nejsou obvykle třeba žádné další logické obvody.

Schéma zapojení obvodu 8255 ukazuje obr. 2.9, pohled na rozmístění vývodů obr. 2.10. Označení vývodů udává tab. 2.4, a shrnutí základních druhů provozu tab. 2.5.





Obr. 2.9 Blokové schéma zapojení 8255

Obr. 2.10 Rozmístění vývodů obvodu 8255

1 - napájecí napětí, 2 - datová sběrnice
3 - buffer datové sběrnice
4 - řídicí logika pro čtení a zápis
5 - řízení skupiny A, 6 - řízení skupiny B
7 - vnitřní datová sběrnice, 8 - skupina A, obvod A
9 - skupina B, obvod B, 10 - skupina A, obvod C (vyššího řádu)
11 - skupina B, obvod C (nižšího řádu)

Označení	Funkce
D0 - D7	datová sběrnice (obousměrná)
RESET	nulování (vstup)
$\overline{CS}$	aktivace obvodu
$\overline{RD}$	čtení
$\overline{WR}$	zápis
A0, A1	adresa brány a CWR
PA0 - PA7	brána A (bit 0 až 7)
PB0 - PB7	brána B (bit 0 až 7)
PC0 - PC7	brána C (bit 0 až 7)
V_{CC}	napájecí napětí + 5 V
GND	zem (0 V)

Tab. 2.4 Označení vývodů obvodu 8255

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	ČTENÍ (vstupní operace)
0	0	0	1	0	brána A → Datová sběrnice
0	1	0	1	0	brána B → Datová sběrnice
1	0	0	1	0	brána C → Datová sběrnice
					ZÁPIS (výstupní operace)
0	0	1	0	0	Datová sběrnice → brána A
0	1	1	0	0	Datová sběrnice → brána B
1	0	1	0	0	Datová sběrnice → brána C
1	1	1	0	0	Datová sběrnice → CWR
					PASIVNÍ FUNKCE
X	X	X	X	1	Datová sběrnice → 3. stav
1	1	0	1	0	bez funkce
X	X	1	1	0	Datová sběrnice → 3. stav

Tab. 2.5 Základní druhy provozu obvodu 8255

Všech 24 vstupů/výstupů obvodu 8255 je sdruženo do tří 8bitových bran (A, B, C). Podle způsobu naprogramování mohou splňovat různé funkce. Mají různé možnosti, které rozšiřují oblast použití a flexibilitu obvodu 8255. Funkční konfigurace každé brány se nastavuje programem tak, že mikroprocesor vyšle do 8255 vhodné řídicí slovo CW. Existují dva formáty CW, jeden pro definici druhu provozu (režimu, módu) a druhý pro řízení bitů brány C. Do registru CWR lze pouze zapisovat, nelze z něho číst. Brány A, B, C lze nastavit dle potřeby při respektování toho, že jednotlivé brány mohou pracovat jako:

Brána A : jeden 8bitový datový výstupní latch/buffer a jeden 8bitový vstupní latch.

Brána B : jeden 8bitový vstupní/výstupní datový latch/buffer a jeden 8bitový vstupní latch.

Brána C : jeden osmibitový datový výstupní latch/buffer a jeden osmibitový vstupní buffer

Brána C může být rozdělena na dvě čtyřbitové brány. Každá obsahuje jeden 4bitový latch a může být použita pro výstupy ve spojení s branami A a B. Jak již bylo řečeno výše, naprogramováním systému jsou pevně dané tři základní druhy provozu (režimy):

druh provozu 0 - jednoduché vstupy/výstupy
druh provozu 1 - strobované vstupy/výstupy
druh provozu 2 - obousměrná sběrnice.

Je-li na vstupu RESET úroveň log "1" (HIGH), nastaví se všechny brány do stavu pro vstup dat (tzn., že 24 vodičů je ve stavu s velkou impedancí). Po skončení signálu RESET zůstává 8255 v tomto výchozím stavu, aniž by bylo třeba nějakého dalšího nastavení. Během systémového programu (OUT) může být zvolen kterýkoli z druhů provozu. To umožňuje, že jedním obvodem 8255 mohou být ovládána různá periferní zařízení jednoduchým programem.

Druhy provozu bran A a B mohou být definované nezávisle na sobě, zatímco brána C je rozdělena podle požadavků bran A a B na dvě části. **Změní-li se druh provozu,jsou všechny výstupní registry včetně stavových klopných obvodů resetovány.** Druhy provozu lze kombinovat. Např. skupina B může být naprogramována na provoz 0, zatímco skupina A může být naprogramována na provoz 1.

Definice řídicího slova CW pro volbu druhu provozu ukazuje obr. 2.11, typické propojení pro všechny druhy provozu obr. 2.12.

Nastavení jednotlivých bitů. Každý z 8 bitů brány C může být instrukcí (OUT) nastaven na 1 (setován) nebo na 0 (resetován). Nstavení/nulování se uskuteční (viz obr. 2.13) zápisem do CWR (A1A0 =11) řídicího slova CW, které má D7 = 0 a v bitech D3 D2 D1 zakódované číslo bitu brány C, který má být řízen a bit D0 má hodnotu rovnou hodnotě požadované.

Přerušovací funkce. Je-li 8255 naprogramován na druh provozu 1 nebo 2, máme k dispozici řídicí signály, které můžeme použít jako signály pro přerušení pro 8080A. Signály pro požadavek přerušení z brány C mohou být setováním nebo resetováním příslušného klopného obvodu INTE akceptovány nebo nikoli (použitím funkce "Bit set/reset" brány C).

Definice pro klopný obvod INTE:

Bit - SET - INTE je nastaven - přerušení umožněno
Bit - RESET - INTE je nulován - přerušení není umožněno

Poznámka: Všechny klopné obvody se automaticky při výběru druhu provozu a při systémovém resetu nulují.

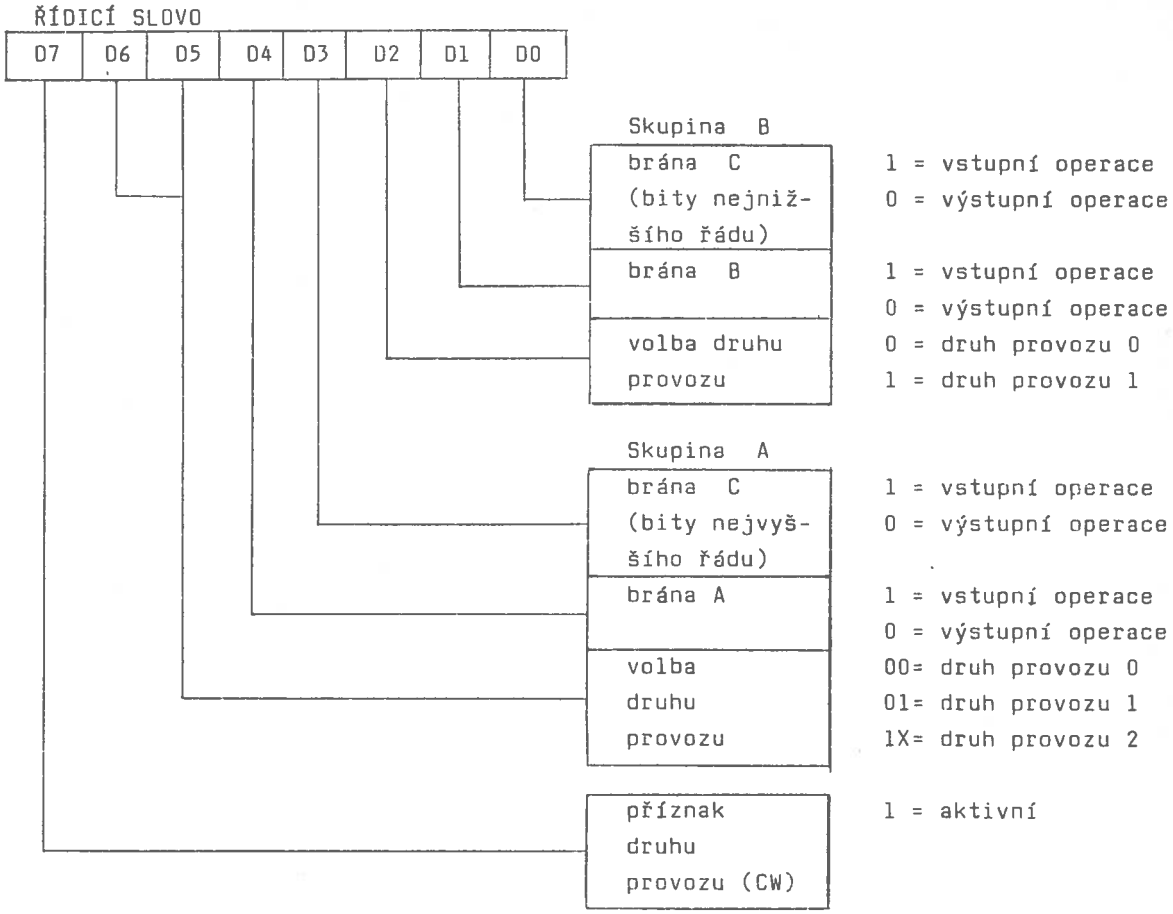
V dalším budeme stručně charakterizovat všechny druhy provozu (režimy):

Druh provozu 0 (základní vstupní/výstupní operace)

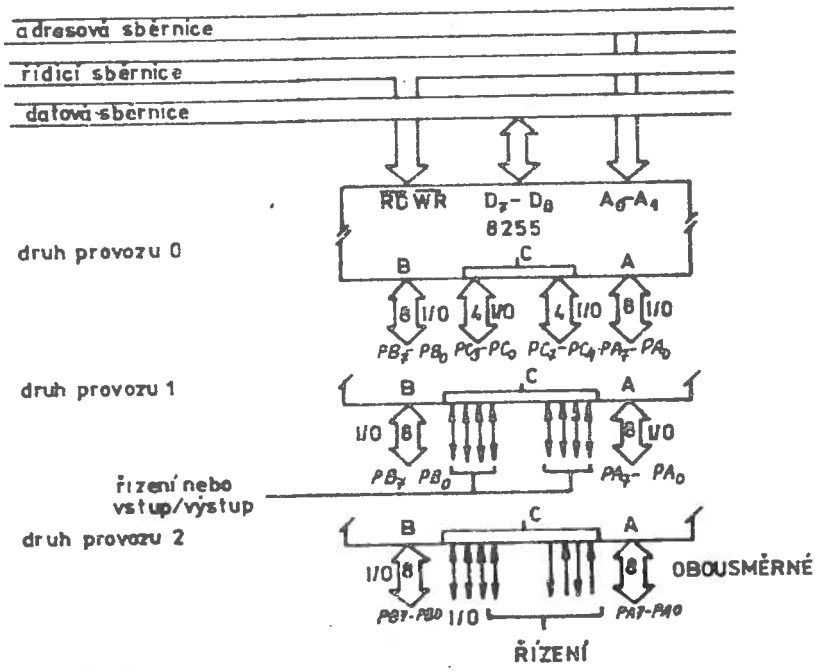
Funkční uspořádání umožňuje jednoduché vstupní a výstupní operace pro každou ze tří bran. Není nutná výměna potvrzení, neboť data jsou prostě do zvolené brány zapsána nebo z ní čtena. Definici bran v tomto případě zachycuje tab. 2.6.

Definici provozu 0 odpovídá:

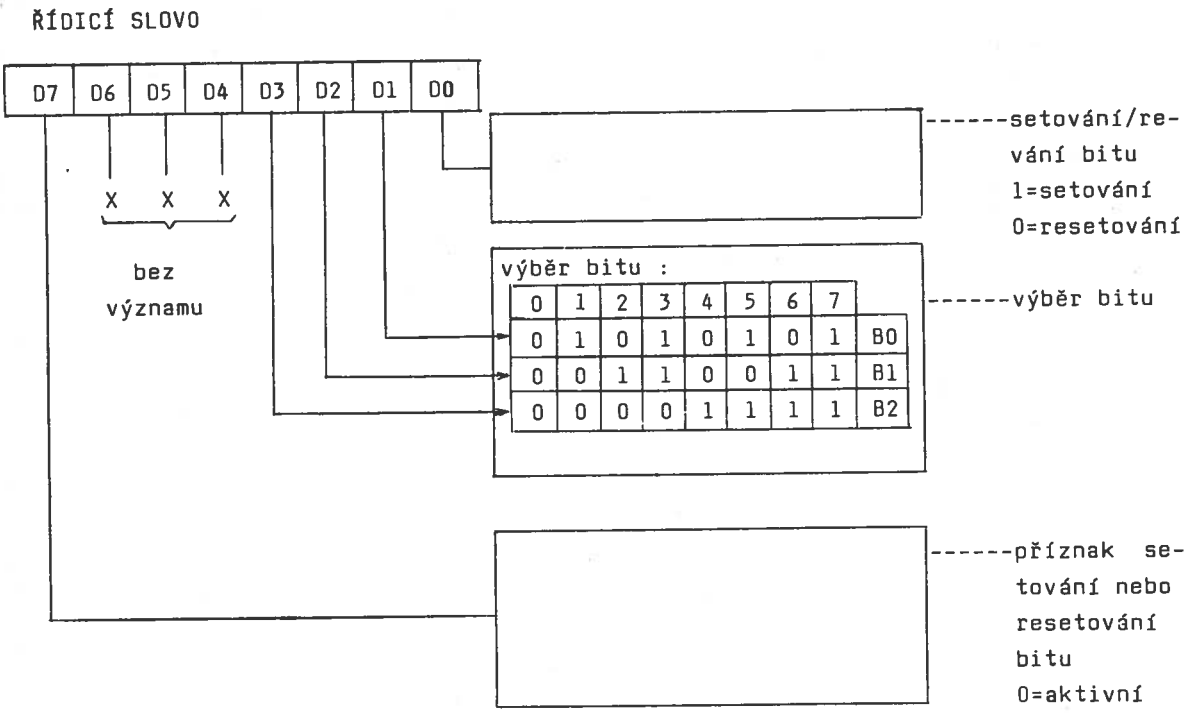
- dvě 8bitové a dvě 4bitové brány
- každá brána může být vstupní nebo výstupní
- výstupy jsou s pamětí (latch)
- vstupy jsou bez paměti
- může být definováno 16 samostatných vstupů/výstupů



Obr.2.11 Definice formátu řídícího slova 8255



Obr.7.12 Definice druhů provozu a interface pro sběrnici



Obr. 2.13 Formát pro ovládání bitu brány C

A		B		Skupina A			Skupina B	
D4	D3	D1	D0	brána A	brána C (bity vyššího řádu)	# č.	brána B	brána C (bity nižšího řádu)
0	0	0	0	výstup	výstup	0	výstup	výstup
0	0	0	1	výstup	výstup	1	výstup	vstup
0	0	1	0	výstup	výstup	2	vstup	výstup
0	0	1	1	výstup	výstup	3	vstup	vstup
0	1	0	0	výstup	vstup	4	výstup	výstup
0	1	0	1	výstup	vstup	5	výstup	vstup
0	1	1	0	výstup	vstup	6	vstup	výstup
0	1	1	1	výstup	vstup	7	vstup	vstup
1	0	0	0	vstup	výstup	8	výstup	výstup
1	0	0	1	vstup	výstup	9	výstup	vstup
1	0	1	0	vstup	výstup	10	vstup	výstup
1	0	1	1	vstup	výstup	11	vstup	vstup
1	1	0	0	vstup	vstup	12	výstup	výstup
1	1	0	1	vstup	vstup	13	výstup	vstup
1	1	1	0	vstup	vstup	14	vstup	výstup
1	1	1	1	vstup	vstup	15	vstup	vstup

Tab. 2.6 Definice bran pro druh provozu 0

Druh provozu 1 (strobovaný vstup/výstup)

Tato funkční konfigurace umožňuje vstup nebo výstup zvolenou bránou v korespondenčním režimu (handshaking), resp. se strobovacími signály. Tyto bitové signály jsou v režimu 1 pro bránu A nebo B generovány nebo přijímány vývody brány C. Definici režimu 1 odpovídají:

- dvě skupiny (A a B)
- každá skupina představuje jednu 8bitovou bránu a 4bitovou bránu
- 8bitová brána může být vstupní i výstupní, vstup i výstup je s pamětí (latch)
- 4bitová brána se užívá pro řízení a kontrolu stavu 8bitové brány

Definice signálů řízení vstupu:

- STB (Strobe Input) - STB = L strobuje vstup, tj. převádí data ze vstupních svorek do vstupního registru
- IBF (Input Buffer Full F/F) - IBF = H indikuje, že data již byla převzata do vstupního registru. Jinak řečeno IBF -> L pro STB -> H a opět IBF -> L po vzestupné hraně signálu RD.
- INTR (Interrupt Reguest) - INTR = H vyvolá v CPU přerušení, pokud to vstupní zařízení vyžaduje. INTR přejde na úroveň H při $\overline{STB}$ = H, IBF = H a INTE = H. Nuluje se sestupnou hranou $\overline{RD}$. Tím se umožňuje, aby vstupní zařízení si žádalo o obsloužení mikroprocesorem pouhým strobováním dat do brány.

Působení signálů INTE A a INTE B je ovládáno nastavením a nulováním bitů PC4 a PC2. Dva bity PC6 a PC7, nevyužité pro řízení v režimu 1, mohou být naprogramovány na vstup nebo výstup a využity libovolně s kteroukoliv bránou nebo samostatně.

- $\overline{OBF}$ (Output Buffer Full F/F) - výstupní bit OBF přechodem na úroveň L indikuje, že CPU již předal data do příslušné brány. Klopný obvod OBF se nastaví vzestupnou hranou vstupu WR a nuluje se přechodem vstupu ACK na úroveň L.
- $\overline{ACK}$ (Acknowledge Input) - ACK = L indikuje, že data z brány A nebo B již byla převzata periferním zařízením.
- INTR (Interrupt Request) - úroveň H na tomto bitovém výstupu může být použita k přerušení CPU při převzetí dat periferním zařízením.
- INTR přejde na úroveň H při ACK = OBF = INTE = H. Nuluje se sestupnou hranou WR.

Působení signálů INTE A INTE B je ovládáno bity PC6 a PC2.

K volnému využití tentokráte zůstávají bity PC4 a PC5.

Jednotlivé kombinace zachycují obr. 2.14 až 2.16.

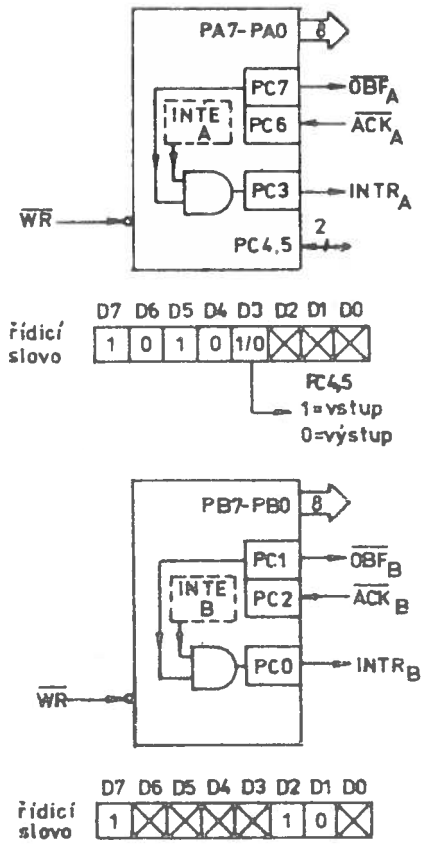
Druh provozu 2 (strobovaná obousměrná brána)

Tento režim se vztahuje pouze k bráně A a k pěti bitům brány C. Ostatní brány resp. bity mohou být využity v režimu 0 nebo 1 . Definici režimu 2 odpovídá:

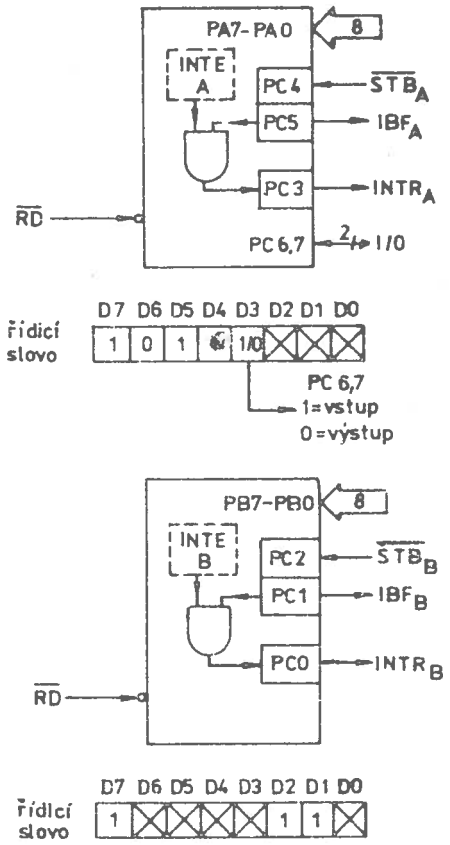
- práce jen se skupinou A
- jedna 8bitová obousměrná brána A a pět bitů z brány C pro řízení a stav brány A
- vstupy i výstupy jsou s pamětí

Definice řídicích signálů obousměrné brány:

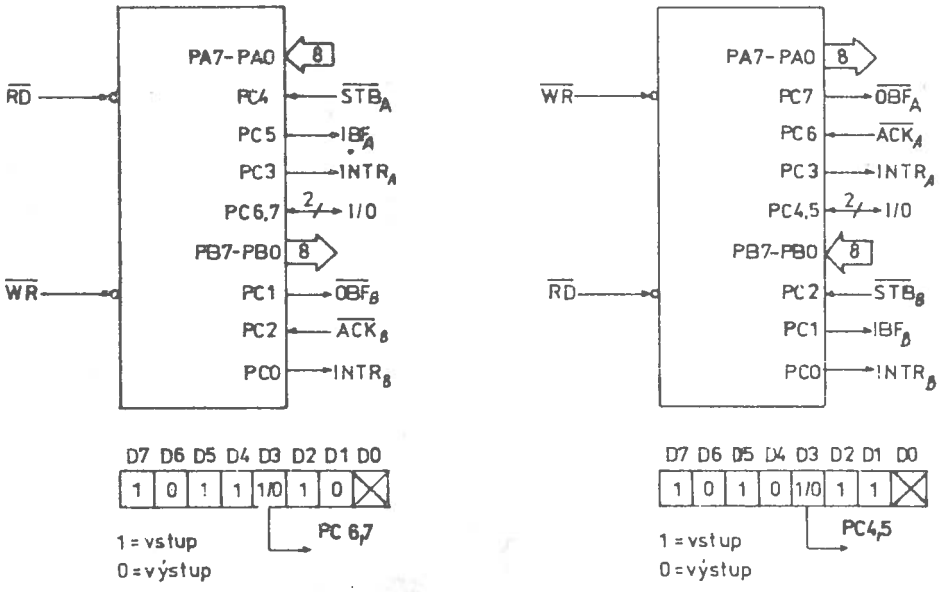
- INTR (Interrupt Request) - požadavek o přerušení CPU pro vstup i výstup bránou A
- $\overline{OBF}$ (Output Buffer Full) - úroveň L oznamuje, že CPU již předal data do registru brány A



Obr. 2.14 Druh provozu 1 - výstupy



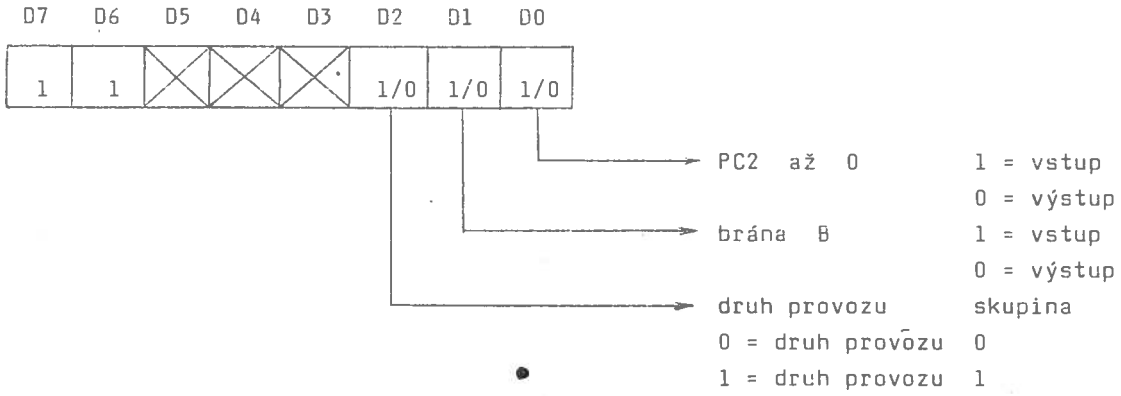
Obr. 2.15 Druh provozu 1 - vstupy



Obr. 2.16 Varianty při druhu provozu 1

- $\overline{ACK}$ (Acknowledge) - úroveň L umožňuje třístavovému výstupu brány A předat data ven. V opačném případě výstup A bude ve 3. stavu.
- INTE 1 (klopný obvod INTE 1 přiřazený k OBF) - nastavuje se bitem PC6.
- $\overline{STB}$ (Strobe Input) - úroveň L převádí data z periferní strany do vstupního registru brány A
- IBF (Input Buffer Full F/F) - úroveň H indikuje, že data již byla převzata do vstupního registru
- INTE 2 (klopný obvod INTE 2 přiřazený k IBF) - nastavuje se bitem PC4

Situaci opět zachycují obr. 2.17 až 2.22, možné kombinace zachycuje tabulka 2.7.



Obr. 2.17 Řídicí slovo pro druh provozu 2

Pokyny pro kombinace druhů provozu (režimů)

Jsou kombinace režimů, při kterých nejsou všechny bity brány využity pro řízení nebo pro instrukci o stavu. Zbývající bity mohou být použity takto:

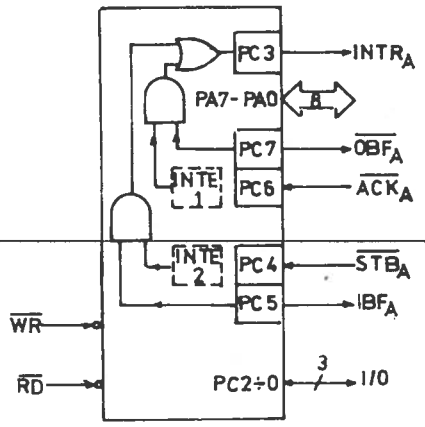
a) jsou-li naprogramovány jako vstupní - všechny vstupní vývody jsou přístupné při obyčejném čtení brány C.
b) jsou-li naprogramovány jako výstupní - bity brány C_H (PC7 až PC4) jsou individuálně přístupné bitovými operacemi (bit set/reset), bity brány C_L (PC3 až PC0) jsou individuálně přístupné bitovými operacemi nebo hromadně zápisem do brány C.

Zatižitelnost vývodů bran B a C :

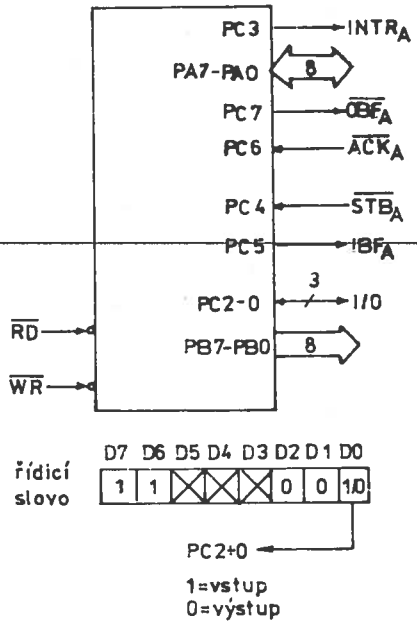
Jakákoli kombinace o s m i výstupů, náhodně vybraných z bran B a C může dodávat proudy až 1 mA při 1,5 V. To umožňuje z 8255 přímo napájet budiče, které tak velké proudy vyžadují.

Čtení stavu brány C

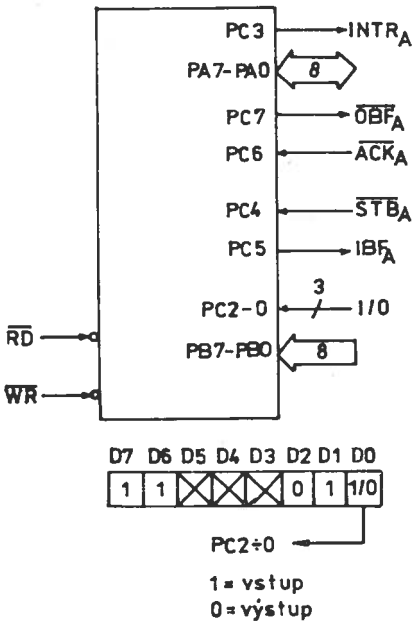
Brána C přenáší při druhu provozu 0 data od nebo do periferních zařízení. Je-li obvod 8255 naprogramován na druh provozu 1 nebo 2, generuje nebo přijímá potvrzovací signály od periferních zařízení. Programátor může vyhodnocením obsahu brány C otestovat nebo přezkoušet stav každé periferie . Stavová informace z brány C se čte bez jakékoliv speciální instrukce.



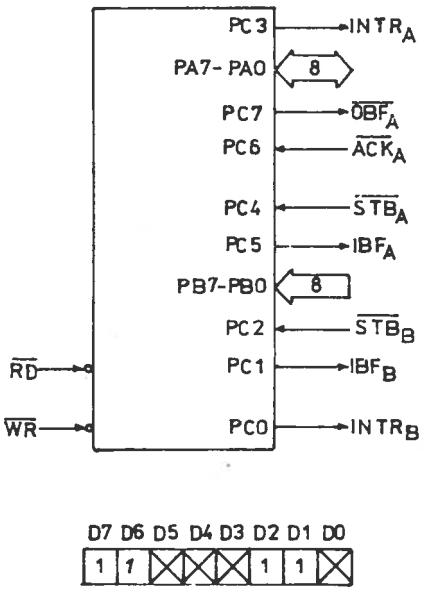
Obr. 2.18 Druh provozu 2



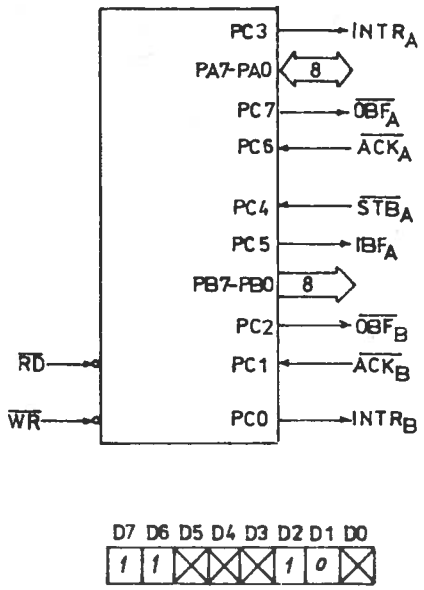
Obr. 2.19 Druh provozu 2 a 0 (výstupní operace)



Obr. 2.20 Druh operace 2 a 0 (vstupní operace)



Obr. 2.21 Druh provozu 2 a 1 (vstupní operace)



Obr. 2.22 Druh provozu 2 a 1 (výstupní operace)

	Druh provozu 0		Druh provozu 1		Druh provozu 2
	vstup	výstup	vstup	výstup	pouze skupina A
PA0	vstup	výstup	vstup	výstup	↔
PA1	vstup	výstup	vstup	výstup	↔
PA2	vstup	výstup	vstup	výstup	↔
PA3	vstup	výstup	vstup	výstup	↔
PA4	vstup	výstup	vstup	výstup	↔
PA5	vstup	výstup	vstup	výstup	↔
PA6	vstup	výstup	vstup	výstup	↔
PA7	vstup	výstup	vstup	výstup	↔
PB0	vstup	výstup	vstup	výstup	
PB1	vstup	výstup	vstup	výstup	
PB2	vstup	výstup	vstup	výstup	pouze druhy
PB3	vstup	výstup	vstup	výstup	
PB4	vstup	výstup	vstup	výstup	provozu 0 a 1
PB5	vstup	výstup	vstup	výstup	
PB6	vstup	výstup	vstup	výstup	
PB7	vstup	výstup	vstup	výstup	
PC0	vstup	výstup	$INTR_B$	$INTR_B$	V/V
PC1	vstup	výstup	IBF_B	$\overline{OBF}_B$	V/V
PC2	vstup	výstup	$\overline{STB}_B$	$\overline{ACK}_B$	V/V
PC3	vstup	výstup	$INTR_A$	$INTR_A$	$INTR_A$
PC4	vstup	výstup	$\overline{STB}_A$	V/V	$\overline{STB}_A$
PC5	vstup	výstup	IBF_A	V/V	IBF_A
PC6	vstup	výstup	V/V	$\overline{ACK}_A$	$\overline{ACK}_A$
PC7	vstup	výstup	V/V	$\overline{OBF}_A$	$\overline{OBF}_A$

Tab. 2.7 Přehledná tabulka definic druhů provozu

Formát stavového slova pro pracovní režim 1

Uspořádání vstupů

D7	D6	D5	D4	D3	D2	D1	D0
V/V	V/V	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
skupina A					skupina B		

Uspořádání výstupů

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBF}_A$	$INTE_A$	V/V	V/V	INR_A	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
skupina A					skupina B		

Formát stavového slova pro pracovní režim 2

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBF}_A$	$INTE_1$	IBF_A	$INTE_2$	$INTR_A$			
skupina A					skupina B		

(Definováno výběrem pracovního režimu 0 nebo 1)

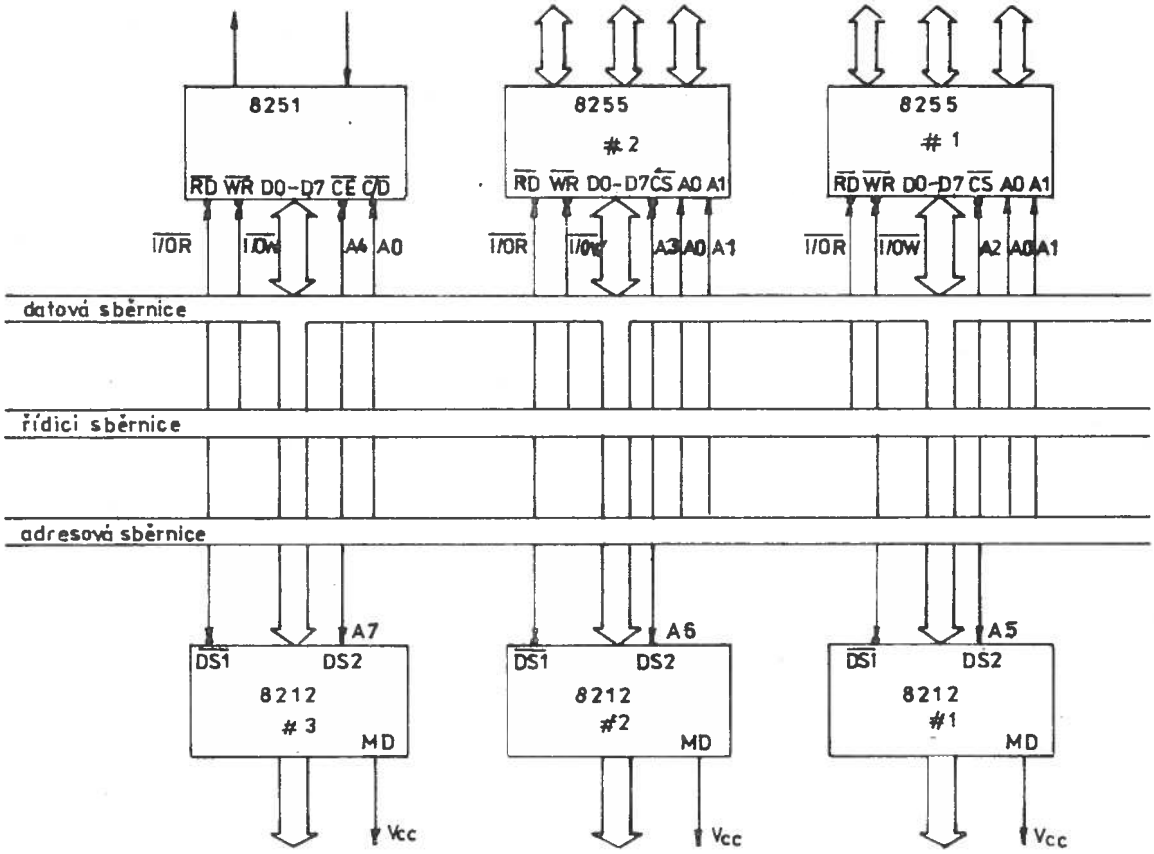
Obr. 2.23 Formát stavového slova pro druh provozu 1 nebo 2

2.8.5.3 Programovatelný seriový interface 8251

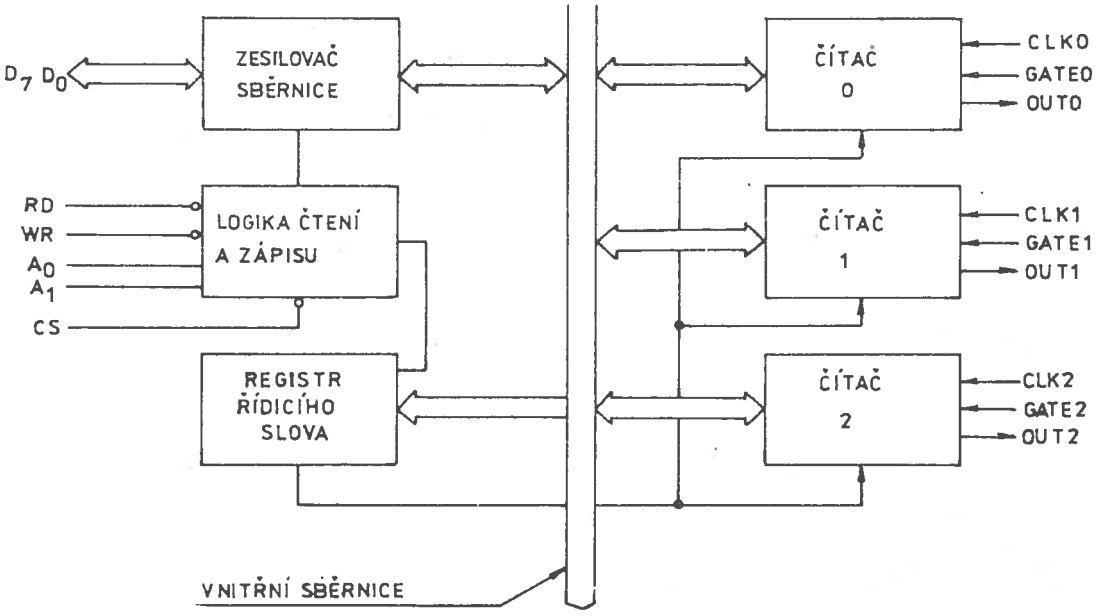
Obvod 8251 je univerzální vysilač/přijimač (výkonový buffer) pro synchronní a asynchronní provoz pro přenos dat v mikropočítačových systémech. Je určen jako ostatní obvody pro periferie mikroprocesoru a je naprogramován tak, že může být použit prakticky pro všechny dnes používané druhy seriového přenosu dat. Výkonový buffer přijme paralelní znaky z mikroprocesoru a pro přenos je převede na spojitý seriový tok dat. Současně může data i přijímat a převádět je na paralelní znaky pro mikroprocesor. Výkonový buffer hlásí mikroprocesoru, kdy může nový znak pro přenos přijmout nebo předat a mikroprocesor může v každém okamžiku "přečíst" stav výkonového bufferu včetně chyby při přenosu dat a včetně řídicích signálů, jako např. SYNDET a TxE. Obvod je vyráběn technologií n-kanálu na křemíku. Vzhledem k tomu, že v cvičeních na školním počítači nebude tento obvod používán, odkazujeme zájemce na příslušnou literaturu.

Příklad propojení obvodů vstup/výstup

Na obr. 2.24 je zapojení typického systému V/V v rozsáhlé kombinaci s jinými obvody (8212, 8251 a 8255).



Obr. 2.24 Připojení obvodů V/V



Obr. 2. 25 Blokové schéma obvodu 8253

2.8.5.4 Programovatelný časovač 8253

Programovatelný časovač 8253 je mimořádně univerzální, neboť obsahuje tři 16bitové čítače s předvolbou, které mohou být využity k nejrůznějším účelům, ale nejčastěji se uplatní při časování (Interval Timer).

Obvod charakterizují následující základní údaje:

- 3 nezávislé 16bitové čítače s předvolbou
- frekvence čítání 0 až 2 MHz
- čítání binární i dekadické
- napájecí napětí +5V
- pouzdro DIL o 24 vývodech
- přímá slučitelnost s mikroprocesorem 8080A.

Obvod 8253 pomáhá řešit problémy generování časových intervalů, které vznikají téměř v každém mikropočítačovém systému. Namísto realizace programových smyček, programátor zajistí vhodnou inicializaci 8253 tak, aby jeden nebo více čítačů, obsažených v 8253, nezávisle na CPU, odpočítával impulsy a přerušil činnost CPU (INT) po uplynutí žádoucího intervalu. Existují přitom i jiné funkce, které lze snadno plnit obvodem 8253. Např. programovatelný generátor rychlosti, čítač impulsů, dělič, generování hodin pro reálný čas atd.

Schéma obvodu 8253 ukazuje obr. 2.25. Obvod 8253 spolupracuje s datovou sběrnicí 8080A přes vyrovnávací registr instrukcemi IN a OUT. Vyrovnávací registr zabezpečuje programování režimu, zavádění předvolby do čítačů i jejich čtení.

Logika čtení a záznamu má pět bitových vstupů z mikropočítače a generuje vnitřní řídicí signály pro 8253. Jsou to vstupy:

- $\overline{RD}$ - úroveň L oznamuje 8253, že CPU čte data, tj. stavy čítačů
- $\overline{WR}$ - úroveň 1 oznamuje 8253, že CPU předává na sběrnici data, tj. řídicí slovo CW nebo hodnotu předvolby čítačů
- A0,A1 - tyto vstupy jsou obvykle připojeny k adresové sběrnici. Slouží k volbě jednoho ze tří čítačů a k adresování registru řídicího slova CWR při určování režimu (módu)
- $\overline{CS}$ - úroveň L aktivuje 8253 pro operaci čtení nebo zápisu z CPU. Vstup $\overline{CS}$ však nemá žádný vliv na průběžnou činnost čítačů.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	
0	1	0	0	0	předvolba čítače č. 0
0	1	0	0	1	předvolba čítače č. 1
0	1	0	1	0	předvolba čítače č. 2
0	1	0	1	1	zápis CW módu
0	0	1	0	0	čtení stavu čítače č. 0
0	0	1	0	1	čtení stavu čítače č. 1
0	0	1	1	0	čtení stavu čítače č. 2
0	0	1	1	1	3. stav
1	X	X	X	X	desaktivace - 3. stav
0	1	1	X	X	3. stav

Tab. 2.8 Definice funkcí obvodu 8253

Registr řídicího slova CWR je adresován při A0= A1= 1, kdy přijímá a uchovává slabiku z vyrovnávacího registru datové sběrnice. Tato informace určuje operační režim každého čítače, výběr binárního nebo dekadického způsobu čítání a předvolbu registru. Do CWR lze pouze psát. Jeho obsah nelze zpětně zjišťovat čtením.

Čítače č.0, 1 a 2 jsou funkčně identické a tak popíšeme jen jeden z nich. Každý čítač je 16bitový, s předvolbou a čítá dolů buď binárně nebo dekadicky. Funkce jeho vstupu (CLK), hradla (GATE), výstupu (OUT) jsou určeny naprogramovaným režimem (módem), uloženým v CWR.

Čítače jsou navzájem nezávislé a každý může mít odlišný režim způsobu čítání. Zjištění obsahu kteréhokoliv čítače může programátor zajistit jednoduchou operací čtení při skončení čítání a také je k dispozici speciální povel a logika umožňující čtení stavu čítače "za chodu", aniž by se narušovalo čítání ze vstupu CLK.

Režim činnosti 8253 se volí programově prostou výstupní operací mikropočítače, přičemž každý z čítačů se nastavuje individuálně zápisem CW do CWR při A0 A1 = 11. Situaci zachycují tabulky 2.9 až 2.13.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Symbol	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Tab. 2.9 Formát CW 8253

SC1	SC0	
0	0	čítač č. 0
0	1	čítač č. 1
1	0	čítač č. 2
1	1	neplatný

Tab. 2.10 Volba čítače (SC-Select Counter)

M2	M1	M0	režim
0	0	0	0
0	0	1	1
X	1	0	2
X	1	1	3
1	0	0	4
1	0	1	5

Tab. 2.11 Nastavení režimu (módu) čítače (M - Mode)

R e ž i m 0: Přerušení na konci čítání.

Výstup OUT čítače je po operaci volby režimu napřed na úrovni L. Poté, co je do čítače vložena hodnota předvolby, OUT zůstane na úrovni L a čítač zahájí čítání. Po dočítání přejde OUT na úroveň H a zůstane na ní, dokud registr uvažovaného čítače není znovu předvolen.

Změna předvolby v průběhu čítání způsobí:

a) zavedení 1. slabiky předvolby zastaví probíhající čítání

b) zavedení 2. slabiky předvolby odstartuje nové čítání

Vstup GATE povoluje čítání, je-li na úrovni H a zabraňuje čítání při úrovni L.

RL1	RL0	
0	0	Operace s pomocným registrem - viz operace čtení
1	0	Čtení/zápis jen vyšší slabiky
0	1	Čtení/zápis jen nižší slabiky
1	1	Čtení/zápis nižší slabiky a pak vyšší slabiky

Tab. 2.12 Čtení/předvolba (RL - Read/Load)

BCD	
0	Binární čítání - 16bitový čítač
1	Dekadické čítání - čítač o 4 dekádách

Tab. 2.13 Základ čítání (BCD)

Režim 1: Programovatelný monostabilní generátor

Výstup OUT přejde na úroveň L po čítání následujícím za vzestupnou hranou na vstupu GATE.

Výstup OUT přejde na úroveň H po dočítání. Jestliže byla vložena nová předvolba, zatímco OUT = L, až do dalšího přepnutí to nikterak neovlivní délku osamoceného impulsu (One-Shot). Průběžný stav čítače může být čten kdykoliv, aniž by to ovlivnilo generování impulsu.

Impuls je možné znovu spouštět i v době jeho trvání, neboť OUT zůstane na úrovni L po celou dobu čítání po vzestupné hraně vstupu GATE.

Režim 2: Dělení frekvence číslem N

Výstup OUT bude na úrovni L po jednu periodu průběhu na hodinovém vstupu CLK (Clock). Rozteč mezi výstupními signály se rovná číslu N zadanému do registru čítače (Count Register - není totožný s čítačem). Jestliže byl obsah registru čítače znovu změněn mezi dvěma výstupními impulsy, probíhající výstupní perioda se nezmění. Následující perioda však už bude odpovídat nové předvolbě.

Vstup GATE, pokud je na úrovni L, převede OUT na úroveň H. Jakmile nastane GATE = H, zahájí čítač čítání z počáteční hodnoty. Vstup GATE proto může být využíván k synchronizaci čítače.

V případě nastavení režimu 2 zůstane OUT = H, dokud není registr čítače naplněn. Výstup proto může být synchronizován i programem.

Režim 3: Generování obdélníkových průběhů

Činnost v režimu 3 je podobná režimu 2, až na to, že výstup OUT zůstane na úrovni L, dokud neproběhne polovina všech čítání (pro sudé N), a přejde na úroveň H na dobu druhé poloviny z celkového počtu čítání. V případě, že číslo N je liché, bude OUT = H po dobu (N + 1)/2 čítání a OUT = L po dobu (N - 1)/2 čítání.

Pokud však registru čítače zadáme novou hodnotu v průběhu čítání, nová hodnota se uplatní ihned po té změně výstupu OUT, Která odpovídala předchozímu obsahu.

Režim 4: Programem ovládané strobování

Po nastavení režimu je OUT = H a po zavedení N začne čítač pracovat. Po dočítání přejde výstup OUT na úroveň L na dobu jedné periody CLK a pak se vrací na H.

Jestliže je obsah registru čítače změněn v době mezi výstupními impulsy, průběžná perioda výstupu se tím neovlivní. Nová hodnota N se však uplatní ihned v další periodě. Čítání se zastaví, je-li vstup GATE = L.

Režim 5: Obvodově ovládané strobování

Čítač zahájí čítání po vzestupné hraně ovládacího vstupu GATE a po dočítání přejde výstup OUT na úroveň L na dobu 1 periody CLK.

Výstup OUT neklesne na úroveň L, dokud po poslední vzestupné hraně GATE neodpočítá úplné číslo N.

Operace zápisu a čtení do 8253

Při zápisu musí program mikropočítače každý z čítačů 8253 inicializovat, tj. zapsat řídicí slovo CW (režim) a hodnotu N (1 nebo 2 slabiky) do registru čítače dříve, než přistoupí ke skutečnému použití vybraného čítače. Přitom zapsání CW je možné v libovolném pořadí. (Každý CWR má vlastní adresu).

Zavedení čísla N do registru čítače však musí probíhat přesně v pořadí naprogramovaném v CW módu, ale jakmile začne být registr čítače plněn, musí být naplněn tím počtem slabik, které je určeno v CW módu.

Všechny čítače čítají dolů, tj. jejich obsah se čítaním zmenšuje a čítání se končí dosažením nuly (terminal count). Zavedení nuly do registru čítače vede k maximálnímu počtu čítání (2^{16} binárně, 10^4 dekadicky). V režimu 0 nové čítání nezačne, dokud není dokončeno plnění registru čítače, který očekává jednu nebo dvě slabiky v závislosti na CW módu (RL0, RL1). Pak se pokračuje operací nulování (restart).

V četných aplikacích je třeba zjišťovat průběžnou hodnotu v čítači za chodu a rozhodovat podle ní. Čítání událostí (impulsů) je patrně nejobvyklejší aplikací, kdy je to zapotřebí. 8253 zahrnuje i logiku, která dovoluje číst obsah kréhokoliv ze tří čítačů bez narušování jeho činnosti.

Existují dvě metody. První metoda spočívá v použití prosté vstupní operace čtení CPU z čítače, kdy prostřednictvím adresových bitů A1, A0 programátor zvolí žádoucí čítač. Připomeňme zde však, že při A1 A0 = 11 (CW módu) žádné čtení nemůže nastat. Jediné omezení této metody spočívá v tom, že pro zajištění ustálené hodnoty v čítači se musí na dobu čtení zastavit čítání ve zvoleném čítači. Zajistí se to snadno s pomocí řídicího vstupu GATE nebo externí logikou, která zablokuje vstup CLK. Obsah čítače se čte nadvakrát. Jako první se čte LSB a jako druhý MSB.

Je zcela nezbytné uskutečnit obě čtení. Jestliže se naprogramuje čtení dvou slabik, pak obě se musí přečíst dříve, než se do téhož čítače opět zapisuje.

Druhým způsobem je čtení okamžitého obsahu čítače bez jakéhokoliv zásahu do operace čítání. 8253 je k tomu vybaven speciální logikou, která se zpřístupní zavedením jednoduchého povelu do CWR. Když si programátor přeje zjistit stav určitého čítače za chodu, pošle do příslušného CWR slabiku CW módu následujícího formátu:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

kde specifikují:

- SC1 a SC0 - čítač, jehož stav má být zachycen do pomocného registru (latch)
- 00 v D5 a D4 označují operaci zachycení stavu
- ostatní bity CW (D3 až D0) jsou bez významu

Jako u předešlé metody, i u tohoto způsobu je nutné uskutečnit úplnou operaci čtení, tak jak byla programována.

2.9 Zásady pro tvorbu programů a jejich ladění

Sestavení programu pro mikropočítač znamená sestavení sledu kódovaných instrukcí. Při návrhu programu musíme přihlédnout k reálným možnostem mikropočítače (schopnost testovámí, způsoby adresování atd.). Řešení programu přitom bývá vhodné rozdělit na menší, dobře přehledné a snadno odladitelné části, tedy zpracovat ve formě podprogramů.

Tato forma je natolik výhodná, že mnohé programy pro mikroprocesorem řízené přístroje či zařízení jsou organizovány tak, že hlavní program tvoří pouze posloupnost volání podprogramů, každý z těchto podprogramů volá opět další podprogramy atd. Je ovšem pravda, že tato struktura programů má i své nevýhody, např. v kritických aplikacích v reálném čase může zpomalovat dobu výpočtu či odezvy.

Mezi hlavní výhody řešení programového vybavení pomocí podprogramů patří:

1) Každý z podprogramů můžeme vytvořit jako relativně krátký, dobře srozumitelný a relativně dobře testovatelný a odladitelný blok celkového programu.

2) Vazby mezi všemi částmi celkového programu (tj. mezi volajícím a volaným podprogramem) můžeme vytvořit přehledné a dobře definovatelné.

Z těchto hledisek si v dalším všimneme základních vlastností mikropočítače a jejich využití při sestavování programů.

2.9.1 Testování

Schopnost testovat booleovské (tj. dvouhodnotové) proměnné patří k velmi důležitým vlastnostem počítače. Pro testování obvykle používáme zvláštní skupinu registrů (viz odst. 2.1), u mikroprocesoru 8080A je to registr příznaků. Hodnoty bitů tohoto registru můžeme pomocí některých instrukcí testovat a podle výsledku testu větvit program. Přitom testy prováděné v programu můžeme rozdělit:

a) jednoduché testy řešící dvouhodnotová rozhodování

b) testy řešící mnohohodnotová rozhodování

ad a) **Jednoduché testy:**

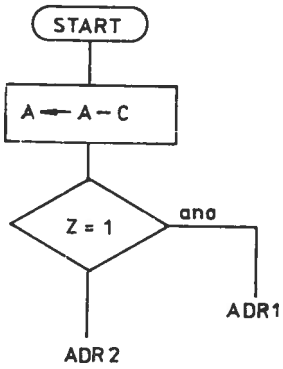
V těchto případech nás zajímá pouze hodnota jedné proměnné testované počítačem, např. hodnota příznakového bitu nulového výsledku (Z). Příkladem může být test na obsah střadače po provedení určité operace (např. po odečtení obsahu registru). Příklad takového programu ukazuje obr. 2.26. Shodným způsobem můžeme použít i dalších příznakových bitů aplikací příslušné instrukce. Podobnou záležitostí jsou i programy realizující zpoždění určitou částí programu.

Situaci ukazují obr. 2.27 a 2.28.

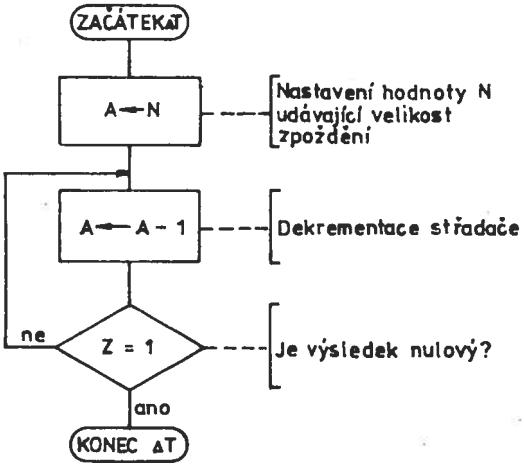
V prvním případě je využit střadač, v druhém případě použijeme některého z univerzálních registrů procesoru k uložení čísla udávajícího požadovaný počet průchodů cyklem. V případech, ve kterých máme za úkol testovat určitou množinu prvků, je vhodné použití tzv. "masky", tj. vhodně zvoleného čísla, které při operaci AND vyloučí nepotřebné bity. Příklad takového programu ukazuje obr. 2.29, kde máme za úkol zjistit, zda jsou bity b0 a b4 v registru B jedničkové. Do střadače vložíme masku, která vyloučí nepotřebné bity a pomocí příznaku Z větvíme program na ADR1 nebo ADR2.

ad b) **Testy řešící mnohohodnotová rozhodování**

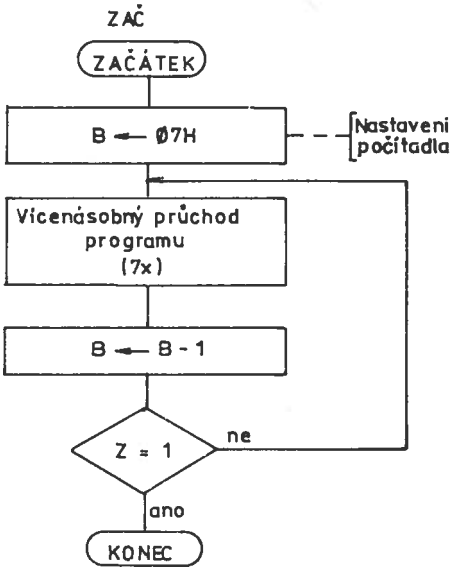
Pokud posloupnost testů, řešících mnohohodnotová rozhodování není příliš dlouhá, lze použít kaskádu jednoduchých rozhodování obdobných principu z příkladu na obr. 2.29.



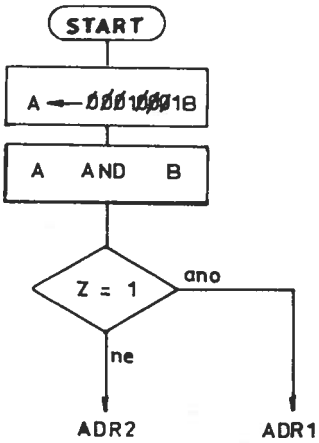
Obr. 2.26 Test nulového výsledku



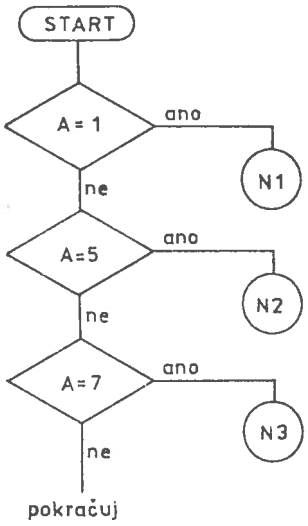
Obr. 2.27 Realizace zpoždění pomocí cyklu



Obr. 2.28 Vícenásobný průchod programu skupinou instrukcí



Obr. 2.29 Určení, zda alespoň jeden ze dvou bitů má hodnotu 1



Obr. 2.30 Mnohohodnotové rozhodování

Příklad: Rozhodněte, zda se obsah střadače shoduje s některým z čísel 1, 5, 7. Pokud se rovná 1, pokračuje program na návěstí N1, pokud 5 na N2, pokud 7 na N3.

Řešení úkolu ukazuje obr. 2.30. Pro porovnání je použita instrukce CPI, která porovnává obsah střadače s operandem v druhé slabice instrukce a v případě jejich shody nastaví Z =1.

Obsah střadače zůstává po provedení instrukce nezměněn. Uvedený princip je vhodný pouze do určitého, omezeného, počtu rozhodování, při větším počtu je vhodnější použití tabulek (viz obr.2.31).Principem je prohledávání tabulky referenčních znaků, kde podle pořadového čísla znaku nalezeného v tabulce se v jiné tabulce vyhledá adresa příslušného skoku.

2.9.2 Zásobníková paměť

Při návrhu programu často používáme podprogramů. Instrukce pro volání podprogramu (CALL) obsahuje adresu, která nahradí obsah čítače instrukcí. Po provedení podprogramu se však počítač musí vrátit na to místo programu, odkud byl program volán (resp. na místo následující). Znamená to, že dříve, než nahradíme obsah čítače instrukcí počáteční adresou podprogramu, musíme stávající adresu z čítače instrukcí dočasně uložit na nějaké známé místo v paměti. Aby se pak počítač po provedení podprogramu mohl vrátit do volajícího programu, stačí, aby instrukce návratu z podprogramu (RET), umístěná v podprogramu jako poslední, přesunula zpět do čítače instrukcí dříve uloženou návratovou adresu.
Ve výše uvedeném případu jsme používali paměťové místo jako součást struktury volání a návratu z podprogramu. Struktura pro volání a návrat z podprogramu používá zásobníkové paměti (Stack). Zásobníková paměť je typu poslední dovnitř - první ven (LIFO - Last In First Out).

U mikroprocesoru 8080A je zásobníková paměť vytvořena v paměti RWM a v procesoru je pouze registr zvaný **ukazatel zásobníku** (Stack Pointer - SP), který ukazuje na poslední obsazenou pozici zásobníku - tzv. vrchol zásobníku.

Při volání podprogramu je návratová adresa uložena na vrchol zásobníku a při návratu z podprogramu je adresa z vrcholu zásobníku přesunuta do čítače instrukcí. Tato struktura zásobníku (zásobníkové paměti) umožňuje, aby hlavní program volal podprogram a tento podprogram volal další podprogram atd. Toto vnořování podprogramů je zajištěno právě uložením každé návratové adresy na právě platný vrchol zásobníku. Adresování zásobníku, tj. regulace ukazatele zásobníku (SP) tak, aby ukazoval vždy na vrchol zásobníku, je automatické při volání i při návratu z podprogramů.

Dokumentace objasňující vazby mezi podprogramem a některým z volajících programů musí obsahovat:

1) ve kterých registrech hledá volaný podprogram hodnoty parametrů předávaných

volajícím programem (vstupní parametry)

2) ve kterých registrech předává volaný podprogram výsledky (výstupní parametry)

3) které registry jsou ovlivněny volaným podprogramem

4) jména podprogramů, které volaný podprogram sám volá

Informace o podprogramu je vhodné zapisovat formou komentáře vždy **před** vlastní zápis podprogramu v jazyce symbolických adres.

2.9.3 Tabulky

Tabulky se při programování mikropočítačů často používají a jejich vhodným použitím můžeme program zjednodušit nebo zkrátit dobu výpočtu. Při vytváření algoritmu programu rozvážíme, které aspekty problému jsou více náhodné než pravidelné a ty náhodnější se snažíme vyřešit použitím tabulek.

Abychom mohli vyzvednout správně určitou položku z tabulky, musíme znát:

1) počáteční adresu celé tabulky

2) posunutí adresy dané položky oproti počáteční adrese, nebo pořadí položky v tabulce

3) velikost položky (tzn. kolika slabikami je tvořena)

Pokud jsou všechny položky v tabulce stejné, mohou být za sebou uloženy bez oddělení a k určení vstupní adresy tabulky nám stačí znalost její počáteční adresy, pořadí položky v tabulce a velikost položky. V opačném případě oddělíme jejich konce koncovým znakem. V takovémto případě musíme znát buď všechny počáteční adresy položek, nebo pouze počáteční adresu tabulky a posun jednotlivých položek proti této adrese.

Jako příklad použití tabulky můžeme uvést např. kódování adres podmíněných skoků při mnohonásobném rozhodování (srovnej s postupem v odst. 2.9.1). Bližší je zřejmé z následujícího příkladu (lit. /24/).

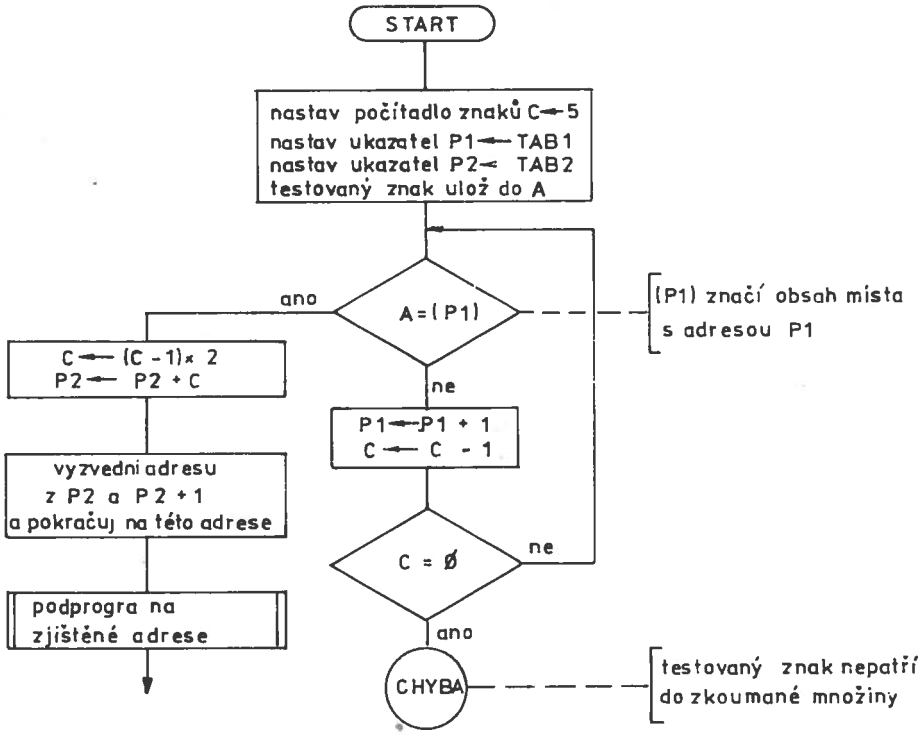
Příklad: Máme rozhodnout, zda určitý znak je shodný s některým ze znaků dané množiny. Každému znaku z této množiny je přiřazena počáteční adresa podprogramu. Pokud se testovaný znak shoduje s některým ze znaků této množiny, máme adresu odpovídající znaku vyzvednout z tabulky.

Situaci ukazuje obr. 2.31. Testovaný znak uložíme do střadače. Program zjišťuje, zda-li je znak ve střadači shodný s některým ze znaků A, B, C, D, E. Počáteční adresa tabulky znaků je TAB 1. Znakům z této tabulky jsou pořiřazeny dvouslabičné adresy a počáteční adresa tabulky adres je TAB 2.

Nastavíme počítadlo znaků C, ukazatel P1 na počáteční adresu tabulky znaků a ukazatel P2 na počáteční adresu tabulky adres. Algoritmus prohledává tabulku znaků, nenajde-li žádnou shodu, končí a hlásí, že znak nepatří do povolené množiny. Najde-li shodu, vypočítá vstupní adresu tabulky adres a adresu přiřazenou nalezenému znaku vyzvedne z tabulky adres. Program pak může pokračovvat skokem na zjištěnou adresu, kde začíná podprogram volený právě testovaným znakem.

2.9.4 Bitové operace

Při realizaci určitého problému, resp. zařízení řízeného mikroprocesorem, se snažíme klást hlavní důraz na programové zabezpečení (software). Při manipulaci s obvody V/V často zjišťujeme, že slovo mikropočítače je buď příliš dlouhé, nebo naopak nepostačující pro potřebný počet vstupních (výstupních) signálů. V takovém případě sdružujeme jednotlivé, vzájemně nesouvisející, signály do ucelených slov. Tím zároveň minimalizujeme i technické řešení (hardware), neboť se sníží počet

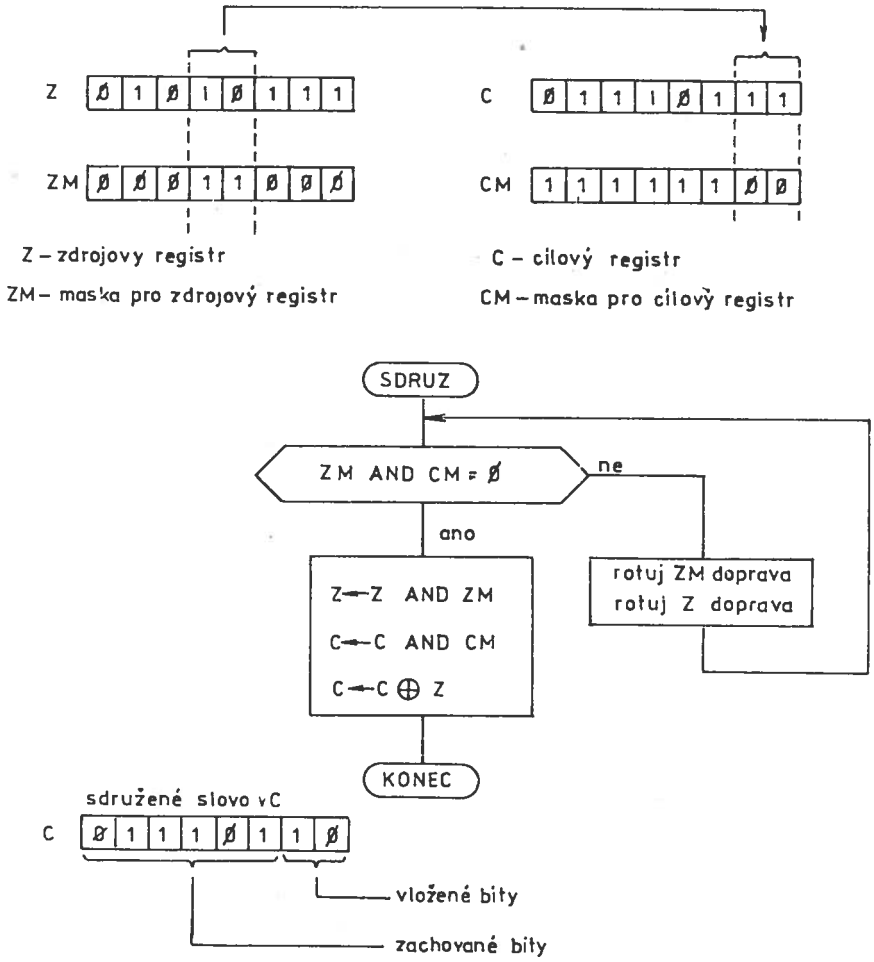


Obr. 2.31 Použití tabulek při mnohonásobném rozhodování

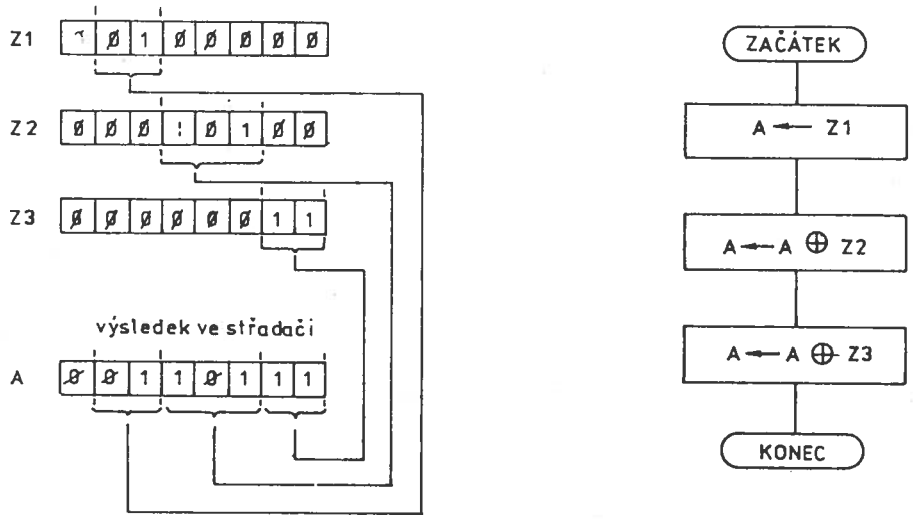
vstupních/výstupních bran. Např. 20 vstupních signálů a 11 výstupních signálů sloučíme do 4 bran vhodně naprogramovaných obvodů 8255. Pro roztřídění jednotlivých signálů potom použijeme programové zabezpečení. Stačí, když si zapamatujeme pozici signálů ve vstupní (výstupní) bráně. Pomocí programu tedy budeme bity buď sdružovat, nebo naopak třídit.

Základní postup při sdružování bitů spočívá v odebrání určitého počtu bitů ze zdrojového slova Z, jejich posunutí do nové polohy uvnitř slova a uložení do cílového slova C bez ovlivnění jeho zbývajících bitů. Pro ilustraci použijeme program SDRUZ, převzatý z lit. /24/. Situaci ukazuje obr. 2.32.

Před použitím podprogramu si připravíme zdrojové slovo i cílové slovo. Zdrojové slovo přesuneme do registru označeného Z, cílové slovo do registru C. Do dalších dvou registrů (ZM a CM) připravíme masku. V registru ZM budou jedničky pouze na pozicích, odkud odebíráme sdružované bity, v registrech CM budou nuly pouze na pozicích, kam bity ze zdrojového slova neukládáme. Podprogram SDRUZ nejprve zkoumá, zdali jsou pozice sdružovaných bitů ve zdrojovém i cílovém slově shodné. Test se provede pomocí operace AND mezi oběma maskami. Pokud nejsou pozice shodné, posouváme pomocí instrukce pro kruhový posun doprava o jeden bit obsah zdrojového registru i jeho masku, až dojde ke shodě. Pak operací Z AND ZM vynulujeme ve zdrojovém registru všechny bity, kterých se sdružování netýká a naopak v cílovém registru operací C AND CM vynulujeme ty pozice, na které přesouváme sdružené bity. Nakonec operací EX-OR sloučíme obě sdružené části do jednoho slova.



Obr. 2.32 Sdružování bitu do jednoho slova



Obr. 2.33 Sdružování předem připravených slov

Výše popsaný algoritmus vyžaduje použití mnoha přesunů. Zjednodušení můžeme dosáhnout vynulováním cílového registru před zahájením sdružování a současně připravením bitů zdrojového registru do cílových pozic spolu s vynulováním nepotřebných bitů. Pak stačí pouze přesunout cílové slovo do střadače a pomocí EX-OR mezi střadačem a zdrojovým slovem slova sdružit. Tuto situaci zachycuje příklad programu na obr. 2.32.

Princip **třídění** bitů slova spočívá ve dvou krocích:

1) všechny nepotřebné bity slova vynulujeme pomocí masky
2 zbylé bity posuneme vlevo nebo vpravo, jak to aplikace vyžaduje

2.9.5 Pole a použití ukazatelů

Častým úkolem při realizaci určitého úkolu je manipulace s velkým množstvím slov (např. ukládání dat z A/D převodníku, textové řetězce atd.). Pole je charakteristické tím, že jeho prvky čteme nebo ukládáme jeden po druhém tak, jak jdou za sebou z jednoho konce pole na druhý a naopak. Přístup k jednotlivým prvkům zajišťuje ukazatel, obsahující adresu prvků pole a jehož dekrementací nebo inkrementací můžeme jednotlivé prvky vyzvednout. Bližší podrobnosti viz např. lit. /24/.

2.9.6 Styk mikropočítače s vnějším prostředím

Pro styk mikropočítače s vnějším prostředím musí být k němu připojena příslušná periferní zařízení. K nim můžeme zařadit např. alfanumerickou nebo hexadecimální klávesnici, displej, tiskárnu, snímač, děrovač, převodníky D/A a A/D atd. Všechna tato periferní zařízení jsou připojena přes stykové obvody - interfejsy (interface) a obvody vstupů/výstupů k mikropočítači.

Z hlediska jejich řízení můžeme rozlišit :

1) **Řešení programovým řízením**, kdy zařízení, které má aktivovat nějakou operaci, použije vhodný stavový bit (nebo jejich soubor) a mikropočítač, který periodicky tyto bity testuje rozhodne o další činnosti. Tímto způsobem vlastně periferie sděluje počítači svůj stav (připraven, porucha atd.).
2) **Řešení pomocí systému přerušení**, kdy zařízení oznámí, pomocí speciálního vstupního signálu, požadavek na přerušení hlavního programu. Procesor přeruší svoji činnost (dokončí právě prováděnou instrukci), provede požadovaný úkon (obslužný program přerušení) a vrátí se zpět tam, kde byl přerušen a pokračuje ve vykonávání původního programu.
3) **Řešení pomocí přímého přístupu do paměti - DMA** (Direct Memory Access).

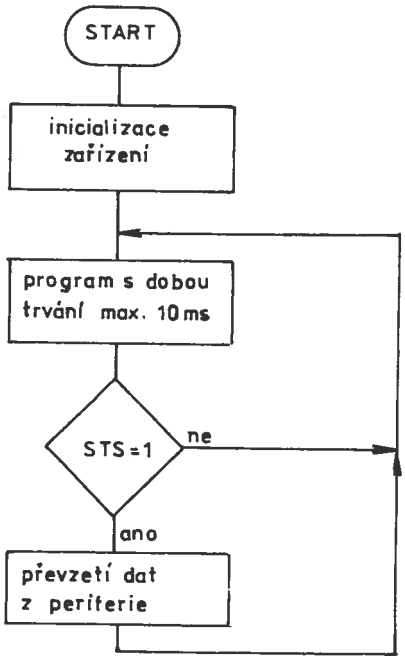
V tomto případě spočívá operace v přenosu bloků dat do paměti nebo naopak s vyloučením činnosti procesoru. Tohoto způsobu se používá zejména tehdy, jestliže situace vyžaduje velmi rychlou odezvu.

V dalším si všimneme zejména dvou prvních způsobů styku mikropočítače s periferními jednotkami, neboť třetí způsob (DMA) vyžaduje určitá technická zabezpečení a svým rozsahem se vymyká ze základních aplikací.

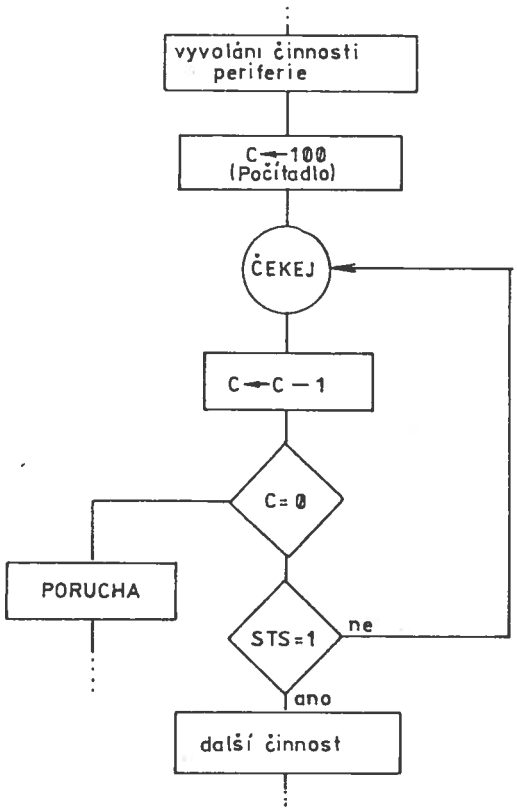
Programové řízení

Použití stavových bitů, případně sdružených do davového slova, představuje pružný a účinný postup s minimálními požadavky na technické zabezpečení. Používáme jej zejména tehdy, jestliže periferie nevyžaduje (z hlediska rychlosti mikropočítače) častou obsluhu, nebo naopak vyžaduje obsluhu v určitých časových úsecích. Typickým příkladem je vstup dat z klávesnice. Je zřejmé, že zadávání dat operátorem

bude podstatně pomalejší než rychlost mikroprocesoru a že postačí, když bude mikropočítač testovat stav klávesnice pouze v určitých časových úsecích (např. každých 10 ms) a ve zbývajícím čase bude vykonávat jinou činnost. Strukturu takového programu ukazuje obr. 2.34.



Obr.2.34 Struktura programu při periodickém testování



Obr.2.35 Řízení periferie

Obdobou této úlohy je např. ovládání motoru nebo jiného zařízení, např. ovládacích magnetů brzdy snímače děrné pásky. Tyto úlohy patří do skupiny úloh, kdy mikropočítač spustí určitou operaci v periferním zařízení a čeká na odezvu, kterou vyšle periferie nastavením určitého stavového bitu indikujícího ukončení činnosti. V tomto případě mikropočítač periodicky testuje stavový bit v tzv. čekací smyčce. Protože však očekávaná odezva také nemusí nastat (porucha) je nutné zabránit uváznutí programu ve smyčce např. způsobem, naznačeným na obr. 2.35. Zde vycházíme ze znalosti časového okamžiku, který má maximálně uplynout než se uskuteční požadovaný úkon. Do smyčky zařadíme počítadlo nastavené tak, aby program čekal ve smyčce jen o málo delší dobu než nastane očekávaná odezva. Pokud do této doby nedojde k nastavení stavového bitu, indikuje počítač poruchu, může spustit diagnostický test atd.

3. ŠKOLNÍ MIKROPOČÍTAČ ŠMS - VÚVT

Školní mikropočítač ŠMS-VÚVT představuje prostředek pro výuku základních poznatků o mikroprocesoru 8080A. Svojí konstrukcí představuje standardní zapojení mikroprocesoru 8080A rozšířené tak, aby byla možná, pro školní účely, realizace maximálního počtu typických úloh. Blokové schéma mikropočítače je uvedeno na obr. 3.1, pohled na desku mikropočítače ukazuje obr. 3.2, mapu paměti zachycuje obr. 3.3.

Školní mikropočítač ŠMS-VÚVT obsahuje základní paměť PROM (max. 4 kB) s monitorem, paměť RWM (max. 2 kB), vstupní/výstupní obvody (28 volných V/V), časovač (8253), A/D a D/A převodník (8 bitů), modem pro kazetový magnetofon (100 - 300 Baud), interfejs RS 232, V 24), klávesnici a displej. Další možnosti rozšíření systému jsou zřejmé z obr. 3.1. Konstrukčně je systém uspořádán ve dvou kufřících, z nichž jeden obsahuje vlastní mikropočítač se zdrojem (viz obr. 3.2), druhý uživatelské periferie (magnetofon MK27, motorek, reproduktor). Spojení s okolím systému lze fyzicky provést přes devět 16-pólových kontaktových polí, dva 5-pólové konektory (MK27 a SM7202) a tři 16-pólové konektory IO pro V/V. Školní mikropočítač je doplněn o interfejs pro dálnopis T 100 a dále je doplněn pomocným monitorem rozšiřujícím možnosti jeho použití.

Pro ovládání mikropočítače slouží 9 příkazových kláves (červené) a 16 hexadecimálních kláves, které jsou zároveň určeny pro vyvolávání obsahu registrových párů na displej.

Monitor (základní) zabezpečuje obsluhu uživatelských přerušení, zápis a čtení dat do paměti atd. V použité verzi je vestavěn rozšiřující (pomocný) monitor, který zabezpečuje obsluhu dálnopisu T 100 použitého ve funkci tiskárny a děrovače a využívajícího jeho klávesnice. Start pomocného monitoru se provádí z adresy 0800 H.

Význam příkazových kláves je následující:

[A] (Adr) ...Přivolává uživatelský čítač instrukcí a zobrazuje jej spolu s adresou na displeji. Následuje-li po stlačení "A" vstup hexadecimálních kláves, zobrazí se nová adresa. Vstup končí stiskem některé z příkazových kláves.

[M] (MEM) ...Přivolá režim vstupu dat na právě zobrazenou adresu, v případě opakování se adresa dekrementuje. (Druhá možnost ukládání dat do paměti je představována klávesou "N", která však adresu inkrementuje!).
Režim zadávání dat indikuje tečka v 6-té pozici displeje. Klávesu "M" lze použít pro prohlížení a vkládání dat ve směru klesajících adres. Dále ji lze použít pro ukončení žádosti o zobrazení registrů, vrchu zásobníku a ukazatele zásobníku.

[N] (NEXT) ...Zápis dat do paměti a její inkrementování.
Při zobrazení registrů vybírá další registr.

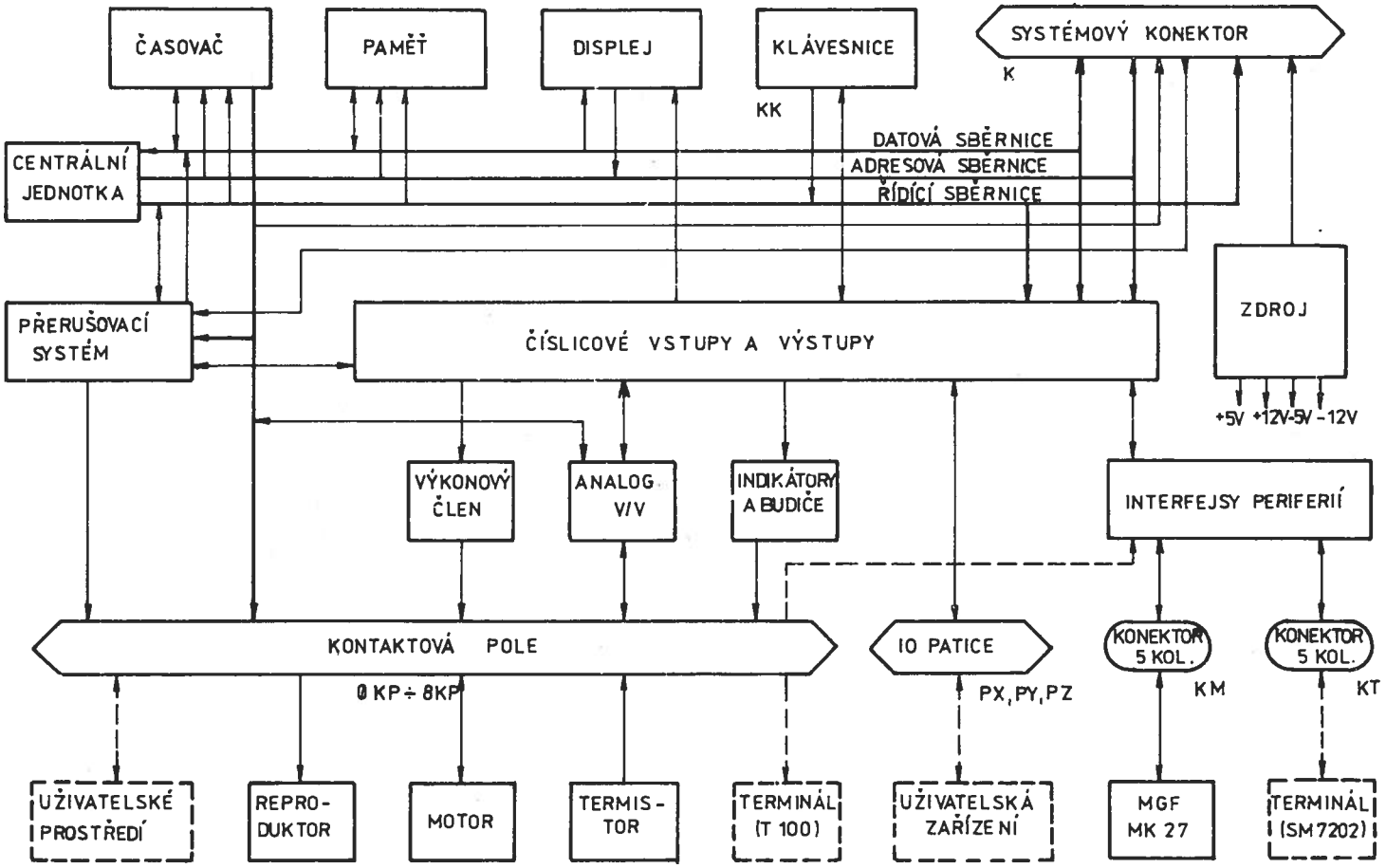
[C] (CLR) ...Ruší data vložená při předcházejícím stlačení příkazové klávesy. Je-li stlačena při zadávání adresy, znovu zobrazí programový čítač. Jde-li o vstup dat do paměti nebo registru, obnoví jejich předcházející obsah.

[R] (REG) ...Stlačení "R" následované hexadecimální klávesou registru zobrazí programový čítač spolu s názvem a obsahem registru.

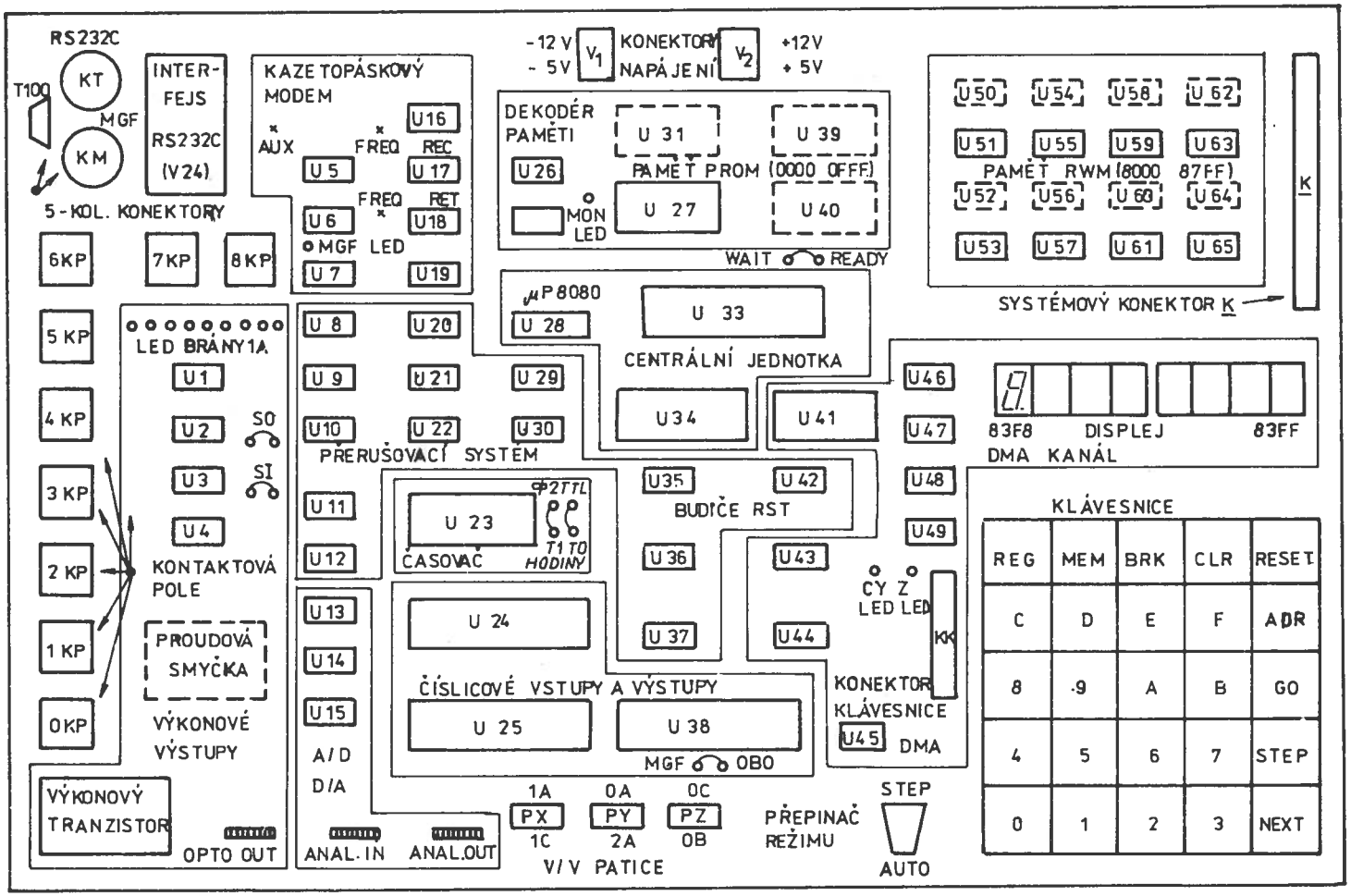
[G] (GO) ...Start programu

[B] (BRK) ...Zobrazí adresu naposled aktivovaného ladícího bodu

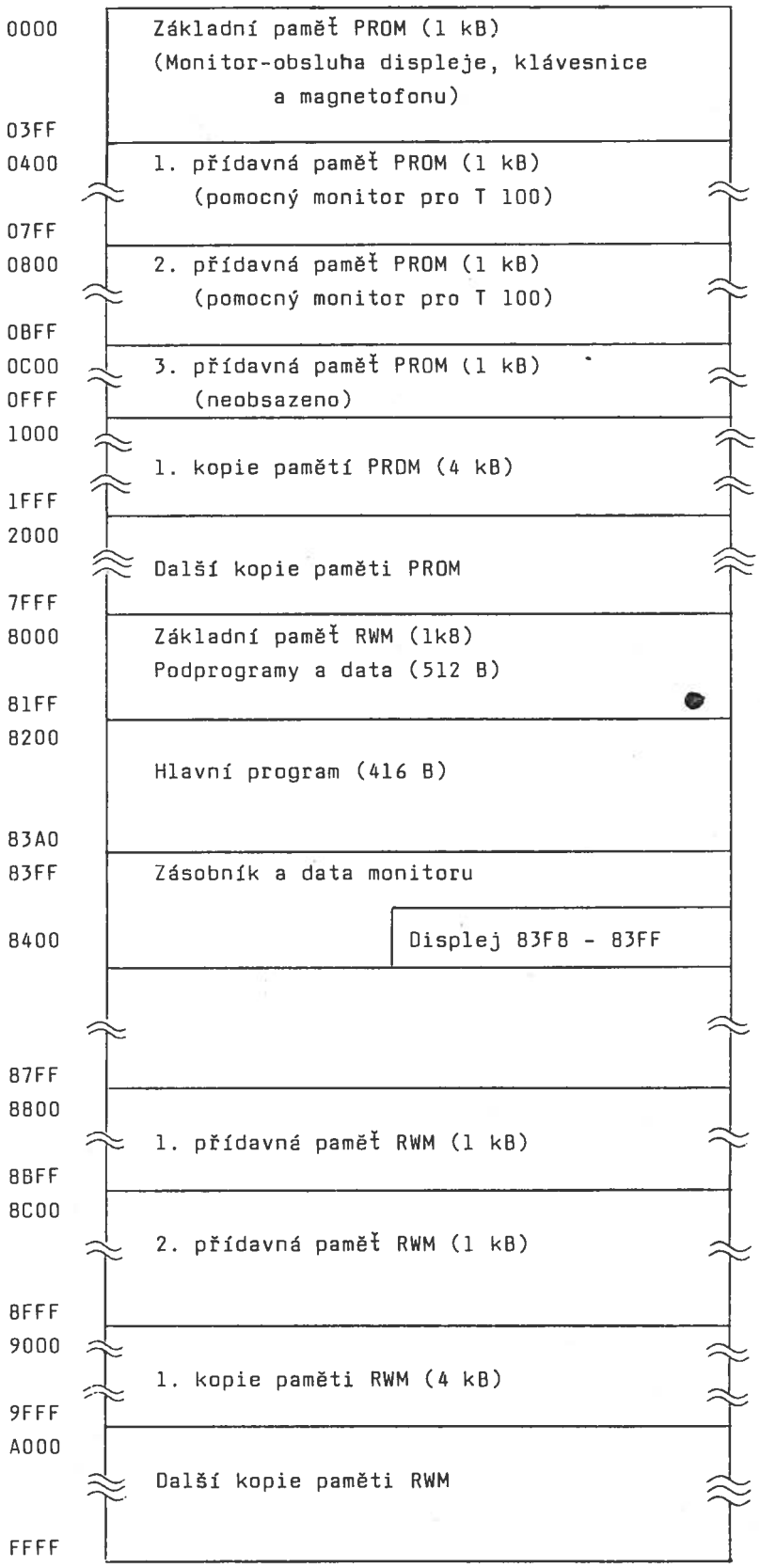
[RESET] ..."Studený start" monitoru na adrese 0000 H



Obr. 3.1 Blokové schéma ŠMS-VÚVT



Obr. 3.2 Rozložení funkčních bloků na desce ŠMS-VÚVT



Obr. 3.3 Mapa paměti ŠMS - VÚVT

3.1 Popis činnosti ŠMS-VÚVT

3.1.1 Monitor - základní funkce

V první etapě, po přeložení programu do strojového kódu, je nutno uložit program do příslušné paměťové buňky. Nejprve oznámíme adresu začátku ukládání klávesou "A" a poté následuje čtyřmístná, hexadecimálně vyjádřená adresa pomocí hexa-kláves. Dalším krokem je povolení změny obsahu paměti - klávesa "M". Adresovaná buňka paměti (je současně zobrazena v levé části displeje) se naplní hodnotou zadanou z klávesnice. Zvýšení adresy se dosáhne stlačením klávesy "N". Lze též ukládat do paměti i v opačném pořadí (i když je to nezvyklé) tzn. od vyšších adres směrem k nižším, pomocí krokování klávesou "M". Ukládání přitom kontroluje monitor a je možné:

- při plnění paměťové buňky lze zadat více údajů (chyba při zadání), ale významné jsou pouze poslední dva
- po zadání nového obsahu paměťové buňky je možné původní obsah obnovit klávesou "C", za předpokladu, že nebyla stisknuta žádná řídicí klávesa
- monitor kontroluje, zda jsou zadané údaje uloženy, v opačném případě vyvolá na displeji text "Err" (např. při zadání neexistující paměťové buňky)

3.1.2 Ladění programu

Po uložení programu do paměti můžeme kontrolovat správnou funkci programu - program ladit. Při ladění je vhodné nechat program probíhat po instrukcích a kontrolovat obsah paměti a obsah registrů. Této funkce dosáhneme zadáním startovací klávesy ("A"), následované 4místnou hexaadresou a postupným krokováním programem klávesou "S". Přitom musí být přepínač STEP-AUTO v poloze STEP. Na displeji se zobrazí adresa následující instrukce a její operační kód. Po každé instrukci je možné si prohlédnout obsah paměti a registrů (A, B, C, D, E, H, L, SP a vrchol zásobníku). Kromě toho je možno průběžně pozorovat, na diodách LED, hodnotu příznaku C a Z. Naposled vybraný registr se zobrazuje po provedení každé instrukce na displeji. Posledně vybranou buňku paměti zobrazíme klávesou "M". Vzhledem k tomu, že tyto funkce jsou dosaženy pomocí přerušení procesoru, je nezbytné dát do každého podprogramu instrukci EI (povolení přerušení). V režimu STEP je možno měnit jak obsahy registrů nebo paměti, tak i je možná změna startovací adresy.

Zobrazení obsahu registru je možné klávesou "R" následovanou klávesou se jménem registru (A, B, C, D, E, F, H, L), klávesou "N" zobrazíme další registr. Párové registry zobrazujeme následovně:

BC	"A" , "B" , "M"
DE	"A" , "D" , "M"
HL	"A" , "B" , "M"
SP	"A" , "1" , "M"
vrchol zásobníku	"A" , "2" , "M"

Současně se zobrazením registrového páru se zobrazí obsah paměťové buňky, na kterou registr ukazuje. Obsah této buňky je možno též měnit.

Po odladění programu můžeme přistoupit k jeho skutečnému běhu. Režim dosáhneme v poloze AUTO přepínače STEP/AUTO shodným způsobem, jako v předchozí části s tím rozdílem, že místo klávesy "S" stiskneme klávesu "G".

Procesor v tomto případě bude trvale vykonávat uživatelský program. Uživatelský program se ukončí při skoku do monitoru na adresu 0030H (instrukce RST 4) - tzv. **"teplý start"** nebo skokem na adresu 0000H - tzv. **"studený start"**. V prvém případě zůstanou obsahy registrů uchovány, stejně tak body přerušení (viz dále), hodnota SP atd. a je možno si je prohlédnout.

V druhém případě (adresa 0000H) je tento skok ekvivalentní stisknutí klávesy

"RESET" a výše uvedené informace nejsou uchovány.

Nevýhodou zmíněného ladění programu je skutečnost, že nedůležité části programu musíme pracně překrokovat klávesou "S". Proto ŠMS-VÚVT umožňuje zadání "bodů přerušení" k některým instrukcím. Jestliže potom dojde v programu k jejich aktivaci, přejde ŠMS-VÚVT na teplý start monitoru. Zde je možno uskutečnit podobnou činnost jako v režimu STEP a poté stiskem "S" nebo "G" (podle požadovaného režimu) pokračovat ve vykonávání programu. Přitom je možno zadat počet průchodů bodem přerušení než se činnost programu přeruší.

Zadání bodů přerušení se řídí následujícími pravidly:

- bod zastavení zadáme posloupností "A", 4 místná hexa adresa instrukce na které je bod přerušení, "B", dvojmístné číslo v kódu hexa = počet povolených opakovaných průchodů daným bodem
- při každém průchodu se počet povolených opakování zmenší o jeden (není-li již nula)
- je-li počet průchodů nula, přechází se na teplý start monitoru, bod přerušení se tím však neruší
- je možno zadat maximálně osm bodů přerušení, každý s maximálně 255 průchody
- bod přerušení možno zobrazit (adresa a aktuální počet povolených průchodů) stiskem klávesy "B" a další body prohlížet klávesou "N" (jen pokud neběží uživatelský program)
- zrušení bodu přerušení dosáhneme jeho zobrazením a stiskem klávesy "C"
- klávesa RESET, resp. skok na 0000 H, ruší všechny body přerušení
- vzhledem k tomu, že způsob realizace je obdobný jako u režimu STEP, musí být přepinač STEP/AUTO v poloze STEP. Je tedy zřejmé, že při funkci "G" se bude program realizovat podstatně pomaleji, než pro případ, kdy by byl přepínač STEP/AUTO v poloze AUTO
- s ohledem na technickou realizaci mikropočítače ŠMS-VÚVT je použití bodů přerušení účinné až po víceslabikové instrukci, tedy že neúčinná bude funkce bodu přerušení ukazující na jednoslabikovou instrukci.

3.1.3 Simulace činnosti procesoru pro RST 5 a RST 6

Skutečný mikropočítač s mikroprocesorem 8080A přechází po RST 5 a RST 6 na instrukci na adrese 0028H, resp. 0030H. V ŠMS-VÚVT je tato část paměti (viz obr. 3.3) obsazena pamětí EPROM. Proto monitor zabezpečí přechod na jiné adresy: pro RST 5 se přechází na adresu, která je uložena na adresy 83ECH, 83EDH (dvě slabiky). Při studeném startu sem monitor ukládá hodnotu 8228H. Pro RST 6 se potom použije adresa 83EAH a 83EBH, na které je při studeném startu uložena hodnota 8230H.

3.1.4 Archivace programu na magnetofonové pásce

ŠMS-VÚVT umožňuje přenést obsah paměti na magnetofonovou pásku běžného magnetofonu, resp. magnetofonu MK 27 tvořícího příslušenství školního mikropočítače (viz obr. 3.2). Určená oblast paměti se přenese postupně, po jednotlivých bitech, do magnetofonu namodulována na akustické kmitočty. Před nahráváním nahrajeme krátký komentář (název, funkce programu, datum, atd.). Poté propojíme ŠMS-VÚVT a MK 27.

Posloupnost ukládání je následující:

"A", adresa začátku přenosu, "M", "A", adresa konce přenosu, "B", "A", "0371". V této fázi generuje mikropočítač nemodulovaný tón určený pro nastavení úrovně nahrání. Poté stiskneme klávesu "G". Přenos je indikován zhasnutím displeje, ukončení přenosu jeho rozsvícením.

P ř i z p ě t n é m č t e n í dat z magnetofonu nejprve ručně nastavíme začátek záznamu na magnetofon a poté aktivujeme příjem dat posloupností: "A", adresa začátku přenosu, "M", "A", "03AE" a zapneme magnetofon. Po skončení slovního komentáře , při přehrávání modulačního tónu, stiskneme klávesu "G". Mikropočítač kontroluje bezchybnost přenosu dat pomocí střadače, ve kterém musí být po ukončení přenosu nula. V opačném případě rozsvítí nápis "Err".

Při přenosu dat musí být přepinač STEP/AUTO v poloze AUTO !

3.1.5 Monitor - rozšiřující funkce

Jak již bylo uvedeno dříve, je použitá verze školního mikropočítače ŠMS-VÚVT doplněna dalším pomocným monitorem. Jeho úkolem je rozšíření základních funkcí o ovládání dálnopisu T 100 a operace s bloky dat.

Pomocný monitor uvedeme v činnost, po propojení dálnopisu přes konektor a sběrnici laboratoře dle pokynů vedoucího cvičení, vyvoláním startovací adresy 0800H. (Posloupnost "A", 0800 , "G").

Monitor se ohlásí (na T 100):

pomocný monitor pro sms s tty
prikazy: d, f, m, s, w, r, b

Po vytisknutí znaku "." očekává zadání příkazu z klávesnice T 100.
Při vstupu do monitoru se uchová obsah všech registrů, vyjma uživatelského PC.

3.1.6 Příkazy pomocného monitoru

Příkaz O <adr.1> ␣ <adr.2> <CR> <LF>

Vytiskne obsah paměti od adresy <adr.1> do adresy <adr.2> . Obě adresy musí být zadány jako dvě hexadecimální čísla, oddělená mezerou a zakončena návratem vozíku a novým řádkem.

Příkaz F <adr.1> ␣ <adr.2> ␣ ⌊konst.⌋ <CR> <LF>

Obsah paměti RWM od adresy <adr.1> do adresy <adr.2> se naplní konstantou ⌊konst.⌋

Příkaz M <adr.1> ␣ <adr.2> ␣ <adr.3> <CR> <LF>

Oblast paměti od <adr.1> do <adr.2> se přesune do paměti RWM počínaje adresou <adr.3>

Příkaz S <adr> ␣ xx-xx- ... ␣ <CR> <LF>

Zobrazuje a mění se obsah paměti od zadané adresy. Po zapsání S <adr> se po stisknutí klávesy "mezera" vytiskne obsah adresy s nápovědným znakem "-". Zadáme-li z klávesnice (T 100) nový obsah zobrazené paměťové buňky, bude automaticky uložen při přechodu na další buňku paměti stiskem klávesy "mezera". Při stisknutí klávesy "mezera" se vždy vytiskne obsah následující adresy. Činnost příkazu zakončuje <CR><LF>

Příkaz W <adr.1> ␣ <adr.2> <CR> <LF>

Přenese obsah paměti od <adr.1> do <adr.2> na děrnou pásku. Při jeho aktivaci postupujeme dle nápovědního textu, který vyšle monitor na T 100.

Příkaz R

Čte a zavádí do paměti data z děrné pásky vyděrovaná dříve příkazem W. Zapnutí a vypnutí čtečky se provede dle nápovědného textu, který monitor vytiskne při aktivaci příkazu.

Příkaz B

Provede návrat do základního monitoru, obnoví obsah uživatelských registrů, s výjimkou PC, vynuluje se displej a očekávají se příkazy základního monitoru z

klávesnice ŠMS-VÚVT.

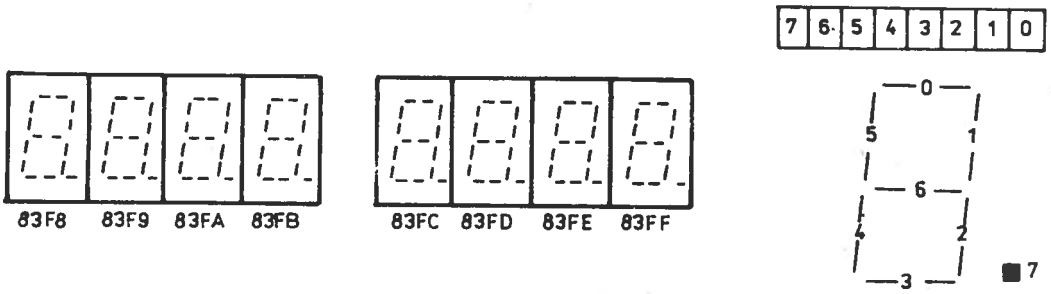
Poznámka: Pokud se monitor ohlásí černě vytisknutým znakem "zvonek" před nápovědným znakem ".", došlo při činnosti k nějaké chybě (např. neznámý znak, chyba na zaváděcí pásce). Při nalezení chyby na pásce při příkazu "R" se zavádění programu do paměti okamžitě přeruší, čeká se na konec pásky a po dalším spuštění se monitor ohlásí znakem "zvonek" před znakem " ".

3.2 Technické prostředky ŠMS - VÚVT

Základní údaje o technickém uspořádání (hardware) ŠMS-VÚVT byly uvedeny v úvodu kap. 3. V dalším si všimneme jednotlivých částí mikropočítače.

3.2.1 Displej

Školní mikropočítač používá 8-místný sedmisegmentový displej zobrazující v hexadecimálním kódu. Pro zobrazení dat se používá přímý přístup do paměti (DMA). Data, která se mají vyslat na displej, jsou zapsána do příslušných paměťových buněk (viz obr. 3.4) a zobrazena.



Obr.3.4 Adresy a pozice bitů v displeji ŠMS - VÚVT

3.2.2 Vstupní/výstupní obvody

Školní mikropočítač má instalovány 3 obvody 8255. Brány 8255 se adresují přímo bity A1 a A0 adresy. Pro označení čísla V/V obvodu, jeho brány a jednotlivých bitů je použita následující symbolika:

X Y Z

Z — číslo bitu (0 - 7)

Y — jméno brány (A, B, C, CWR)

X — číslo obvodu 8255 (0, 1, 2)

Příklad: 2C2 ... bit 2 brány C druhého obvodu 8255

Adresy a přidělení bran V/V obvodu jsou uvedeny v tab. 3.1 a tab. 3.2, programování obvodů 8255 je uvedeno v kap. 2, odst. 2.8.4.

Při stisknutí klávesy "RESET" se všechny brány obvodů 8255 nastaví jako vstupní v režimu 0. Monitor inicializuje poté obvod 8255 č.0, (0A, 0B - vstup, 0C - výstup, režim 0), a tím umožní vstup z klávesnice, výstup na displej atd. Druhé dva

obvody 8255 monitor neprogramuje a pokud nepotřebujeme změnit jejich režim , nemusíme je programovat.

Pro správnou činnost přerušovacího systému se však musí brána 28 naprogramovat pro vstup a brána 2C pro výstup (v režimu 0). Brána 18, vyhrazená pro analogové vstupy/výstupy, se může použít v režimu 0 a 1, pouze u automatického režimu (viz dále) A/D se použije režim 0. Druh provozu (režim) 1 a 2 jsou vhodné pro bránu 1A. Přerušení, generované v těchto režimech, je přímo použité v přerušovacím systému ŠMS-VÚVT. Konfigurace pro druh provozu 0 jsou uvedeny v tab. 3.2. Tyto konfigurace je možno kombinovat s obvodem 8255 č. 1 v režimech 1 a 2.

Příklad programování obvodů 8255:

```
3E80   MVI A, 80H        ; řídicí slabika 8255
                         ; A, B, C = výstup
D307   OUT CNT1          ; vyslání řídicí slabiky na CNT1
3E92   MVI A, 92H        ; řídicí slabika:
                         ; A, B = vstup, C = výstup
D30F   OUT CNT2          ; vyslání řídicí slabiky na CNT2
```

Adresa	Jméno	Funkce	Bity bran 0B, 0C, 1C
00	0A	Vstup klávesnice	0B0 - vstup k modemu magnetofonu
01	0B	Nepřidělené s výjimkou 0B0	0C0 - výstup na modem magnetofonu
02	0C	Monitor - přerušení - viz sloupec napravo	0C1 - povolení monitor. přerušení
03	CNT0	Řídicí brána 8255 0	0C2 - výstup indikace CY
04	1A	Budič a indikace LED	0C3 - výstup indikace [illegible]
05	1B	A/D a D/A převodník	0C4 - povolení kláves 0 - 7
06	1C	Klávesy - viz sloupec napravo	0C5 - povolení kláves 8 - F
07	CNT1	Řídicí brána 8255 1	0C6 - povolení příkazových kláves
0C	2A	Nepřiděleno	0C7 - povolení zobrazení
0D	2B	Stavová slabika přerušení	1C0 - řízení A/D ("1" - aut. A/D)
0E	2C	Povolovací slabika přerušení	1C1 - povolení buzení motoru
0F	CNT2	Řídicí brána 8255 2	1C2 - nepřiděleno
14	TIM0	Časovač 0	1C3 - přerušení (povolované 2C6)
15	TIM1	Časovač 1	1C4 - 7 - nepřiděleno
16	TIM2	Časovač 2	
17	TIMCT	Řídicí registr časovače	

Tab. 3.1 Adresy a přidělení jednotlivých bran obvodů V/V

Poznámka: V tab. 3.1 je ponecháno označení řídicího registru obvodů 8255 8253 shodné s dokumentací ŠMS-VÚVT. Porovnej s odst. 2.8.4!

Po naprogramování obvodů 8255 je možno z jednotlivých bran číst nebo do nich zapisovat instrukcemi IN a OUT.

Příklad:

```
0B0C   IN 2A        ; čtení z brány 2A
3F80   MVI A,80H    ; zápis konst 80H do brány 1A
0304   OUT 1A       ;
3E01   MVI A,01H    ; nastavení 1C0 = 1
0307   OUT CNT1     ;
```

Řídicí slabika	Brána A	Brána B	Brána C0-C3	Brána C4-C7	8255		
					0	1	2
80	OUT	OUT	OUT	OUT		D/A	
81	OUT	OUT	IN	OUT		+	
82	OUT	IN	OUT	OUT		A/D	*
83	OUT	IN	IN	OUT		A/D	
88	OUT	OUT	OUT	IN		D/A	
89	OUT	OUT	IN	IN		+	
8A	OUT	IN	OUT	IN		A/D	
8B	OUT	IN	IN	IN		A/D	
90	IN	OUT	OUT	OUT		D/A	
91	IN	OUT	IN	OUT		+	
92	IN	IN	OUT	OUT	*	A/D	*
93	IN	IN	IN	OUT		A/D	
98	IN	OUT	OUT	IN		D/A	
99	IN	OUT	IN	IN		+	
9A	IN	IN	OUT	IN		A/D	
9B	IN	IN	IN	IN		A/D	

Poznámka : + ... zakázaná konfigurace

* ... standardní operace (použijte pouze tuto operaci)

Tab. 3.2 Řídicí slabiky pro 8255 v ŠMS-VÚVT (druh provozu 0)

3.2.3 Časovač

Školní mikropočítač obsahuje programovatelný časovač 8253. Programování obvodu 8253 bylo uvedeno v kap. 2, odst. 2.8.5. V ŠMS-VÚVT je použita adresace dle tab. 3.3, přehled řízení v tab. 3.4.

čítač	adresa
T0	14
T1	15
T2	16
TINCT (řídicí registr CWR)	17

Tab. 3.3 Adresy 8253 v ŠMS-VÚVT

Typickou činnost časovače ukazuje následující příklad programu:

```
MVI  A, 34H   ; programuj čítač  0  pro
OUT  TIMCT    ; režim 2, s dvouslabikovým
MVI  A, 20H   ; čtením/nastavením registru
OUT  T0       ; počáteční hodnoty a binárním
MVI  A, 0FH   ; počítáním čítače T0
OUT  T0       ; Nastav počítací interval = 0F20H
```

Poznámka : 1) Řídicí slabiky v tabulkách platí pro binární počítání
2) Řídicí slabiky z tab. 3.4 zapisujeme do řídicího registru TIMCT na adrese 17 (viz tab. 3.3).

Čítač	Činnost	Režim					
		0	1	2	3	4	5
0	Vzorkuj	00	00	00	00	00	00
	Čti/nastav pouze LSB	10	12	14	16	18	1A
	Čti/nastav pouze MSB	20	22	24	26	28	2B
	Čti/nastav obě slabiky (LSB jako první)	30	32	34	36	38	3A
1	Vzorkuj	40	40	40	40	40	40
	Čti/nastav pouze LSB	50	52	54	56	58	5A
	Čti/nastav puze MSB	60	62	64	66	68	6A
	Čti/nastav obě slabiky	70	72	74	76	78	7A
2	Vzorkuj	80	80	80	80	80	80
	Čti/nastav pouze LSB	90	92	94	96	98	9A
	Čti/nastav pouze MSB	A0	A2	A4	A6	A8	AA
	Čti/nastav obě slabiky	B0	B2	B4	B6	B8	BA

Tab. 3.4. Řídicí slabiky 8253 V ŠMS-VÚVT

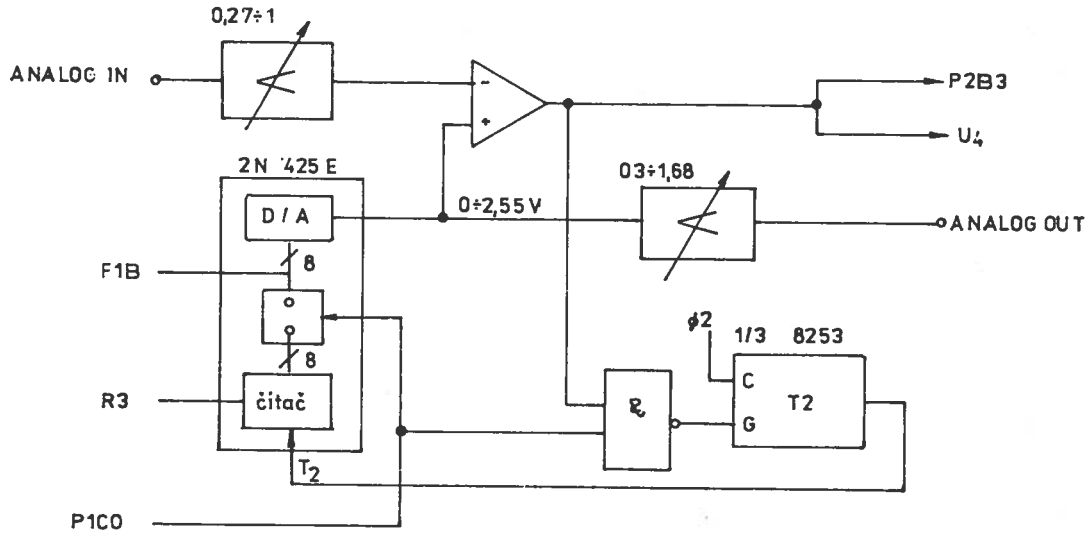
Čítače v ŠMS-VÚVT můžeme použít úplně univerzálně, neboť nemají v systému pevně vymezenou funkci. Jejich hodinové vstupy jsou napojeny na hodiny procesoru (2 MHz) Ø2 TTL, ale je možno je pomocí propojek přepojit jinam, případně i na externí taktovací pulsy (max. 2 MHz). Hradlovací vstupy čítačů 0 a 1 jsou připojeny na úroveň log. 1 (povolení), lze je však též řídit externě, neboť jsou vyvedeny na kontaktové pole. Čítač 2 má hradlovací vstup zapojený v obvodech převodníku A/D. Chceme-li jej použít pro jiné účely, musíme dosáhnout povolení nastavením hradlovacího vstupu na log. 1. Toho dosáhneme nulováním brány 1C0. Vstupy čítačů jsou jinak vyvedené na kontaktové pole, zároveň však vstupují i do přerušovacího systému, kde jsou zdrojem žádostí o přerušení. Čítače 0 a 1 mají v přerušovacím systému záchytné registry žádostí o přerušení schopné uchovat i velmi krátce trvající žádost o přerušení (0,5 μs). Čítač 2 nemá záchytný registr, jeho výstup je přímo hradlovaný vlastním povolovacím signálem 283, a dále sečten do společné přerušovací žádosti INT. Zde se žádost o přerušení od čítače 2 může uplatnit jen po dobu jejího trvání.

3.2.4 Převodníky A/D a D/A

Převodník ve školním mikropočítači používá osmibitový integrovaný převodník D/A typu ZN425E ve spojení s bránou 1B. Výstupní napětí převodníku D/A nabývá (v

souladu s hodnotou čísla na bráně 1B) příslušné hodnoty z 256 diskrétních hodnot napětí v rozsahu (0 mV - 2550 mV). Kalibraci umožňuje potenciometr ANALOG OUT (obr. 3.2), zapojený ve zpětné vazbě výstupního operačního zesilovače typu LM 324. Celkový výstupní rozsah analogového napětí je v rozmezí 0,731 - 3,778 V (cca 3 - 15 mV/bit).

Při realizaci převodníku A/D se výstupní napětí převodníku D/A (ZN425E) porovnává s napětím vstupním, které je (po oddělení vstupním operačním zesilovačem) úměrné nastavení potenciometru ANALOG IN. Rozsah vstupního napětí je limitován napájecím napětím operačního zesilovače a je v rozsahu 0 -3,8 V. Blokové schéma je znázorněno na obr. 3.5.



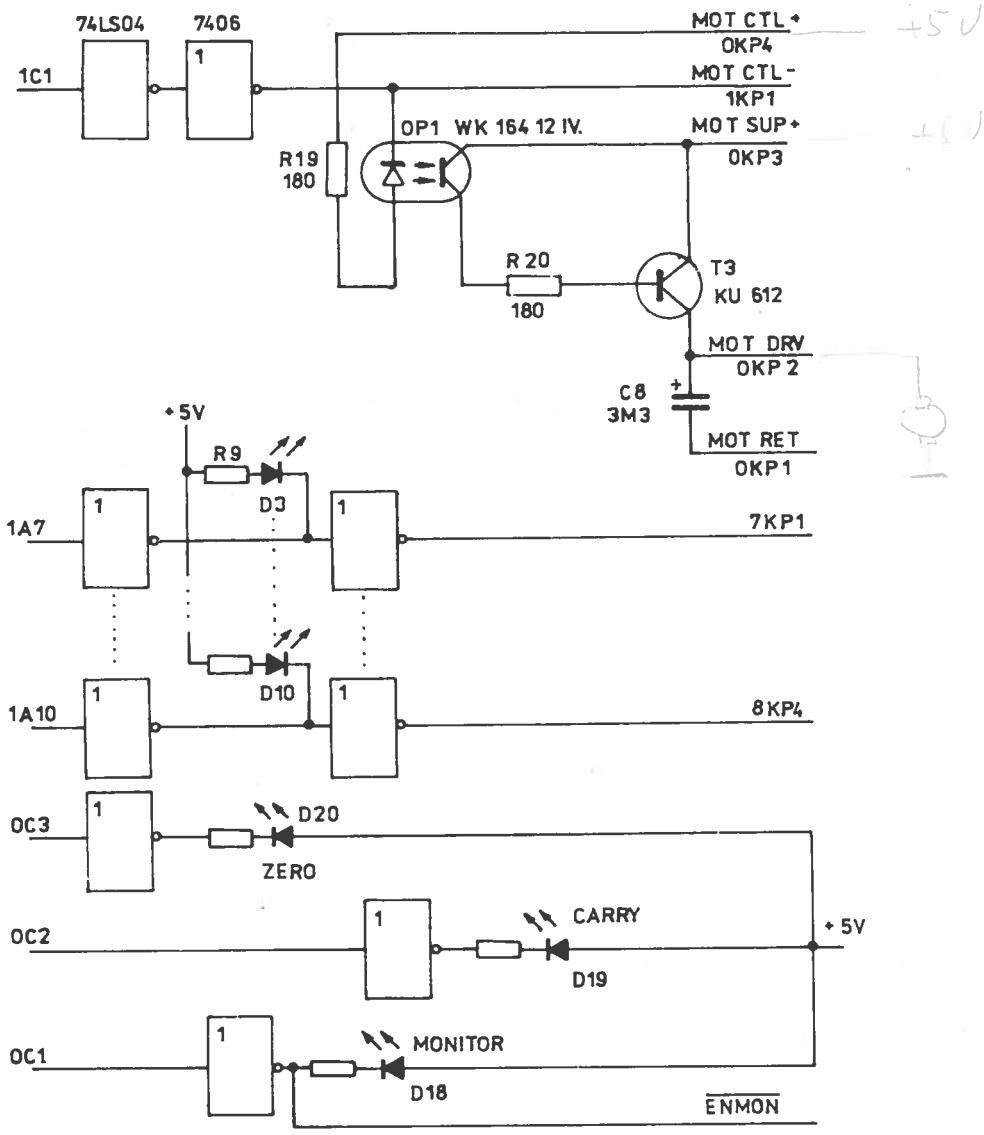
Obr. 3.5 Blokové schéma převodníku A/D a D/A

3.2.5 Výkonové výstupy

Výstupy připojené přes bránu 1C1 (viz obr. 3.6) jsou určené pro buzení výkonových zátěží typu stejnosměrný motorek, relé atd., které vyžadují proudy řádu max. jednotek A. V ŠMS-VÚVT je výkonový výstup oddělen optočlenem a pro regulaci jsou určeny signály MOT CTL+, MOT CTL-, MOT SUP+. Signály určují buzení optočlenu ovládajícího vlastní výkonový prvek (tranzistor KU 612). Signál MOT CTL- lze ovládat buď programově (1C1), nebo externě z kontaktového pole (viz obr. 3.2 a obr. 3.5). Pro signál MOT CTL+ lze použít napětí v rozsahu 0 - 5V. Pro napájení tranzistoru (MOT SUP+) je možné použít vnitřního napájecího napětí +5 V nebo + 12 V nebo externí zdroj.

Ostatní výkonové výstupy (budič 7406 brány 1A) se budí signály přes tuto bránu. Stavy jsou indikovány diodami LED (D3 - D10). Budiče 7406 nemají zapojeny kolektorové odpory, a proto chceme-li je použít pro buzení nějaké zátěže, musíme odpory připojit a připojit je na napětí. Maximální napětí je 30 V, zatížitelnost 40 mA.

Poznámka: V použité konfiguraci ŠMS-VÚVT je vstup 1C1 propojen na interfejs dálnopisu T 100, a proto před aplikací je nutno zrušit příslušnou spojku.



Obr. 3.6 Zapojení výkonových výstupů ŠMS-VÚVT

3.2.6 Přerušovací systém

a) Přerušení monitoru

Úkolem monitorových přerušení je umožnit uživateli prohlížení paměti a registrů po vykonání každé instrukce. Povolení monitorových přerušení lze ovládat ručně (přepinač AUTO/STEP), nebo programově (V/V signálem OC1). Zákaz monitorových přerušení je indikován zhasnutou diodou LED ozn. v obr. 3.2 MON. Vlastností **použitého monitoru** je, že po povolení přerušení (EI) se monitorové přerušení nemůže indikovat dříve, než se vykoná víceslabiková instrukce (viz odst. 3.1.2).

b) Uživatelské přerušení

Tato přerušení lze rozdělit na systémová a externí. Zdrojem systémových přerušení jsou signály generované ŠMS-VÚVT (časovač atd.). Zdroje externích přerušení jsou obvykle mimo ŠMS-VÚVT a je nutno je do příslušného vstupního konektoru přivést.

O přerušení generovaných časovačem již pojednává odst. 3.2.3. S výjimkou 1C3 se všechny zdroje přerušení vedou na bránu 2B jako vstupy, a proto je možno je považovat za stavovou slabiku přerušení. Jestliže dojde k přerušení, může program tuto bránu číst a určit zdroj přerušení. Obvykle je nežádoucí, aby se naráz generovalo více přerušení, a proto se všechny přerušení hradlují povolovacími signály, které je možno programově ovládat na výstupní bráně 2C (povolovací slabika přerušení). Procesor bude akceptovat přerušení, jestliže dojde k přerušovací události a zároveň bude její povolovací bit na bráně 2C nastaven ("1") - Viz tab. 3.5:

Zdroje přerušení	Povolovací bit	Stavový bit
Čítač 0 (kl. obvod)	2C0	2B0
Čítač 1 (kl. obvod)	2C1	2B1
Čítač 2 (bez kl. obvodu)	2C2	2B2
A/D komparátor	2C3	2B3
EXT 4 (kl. obvod)	2C4	2B4
EXT 5 (kl, obvod)	2C5	2B5
Brána 1C3	2C6	1C3
Všeobecné nepovolení	2C7	
EXT 4 (přímé - nepřímé)		2B6
EXT 5 (přímé - nepřímé)		2B7

Tab. 3.5. Přerušení ŠMS-VÚVT

Na bráně 2C7 je signál všeobecného nepovolení pro všechny uživatelská přerušení. Bude-li nastaven ("1"), může být přerušení vyvoláno pouze monitorem. Při systémovém nulování (RESET) se nastaví všechny obvody (viz odst. 3.2.2) jako vstupní v režimu 0. Signálové vodiče však budou ve třetím stavu a logika přerušení vyhodnotí 2C7 jako "1", a proto všechna uživatelská přerušení zablokuje. Jestliže se brána 2C naprogramuje (v režimu 0) jako výstupní, automaticky se nulují všechny bity. Všeobecný zákaz přerušení přestane platit, ale jednotlivá přerušení jsou nyní blokována individuálně na 2C0 a 2C6. Pro jejich povolení je třeba setovat všechny odpovídající bity v bráně 2C. Žádné uživatelské přerušení tedy nemůže vzniknout, dokud není 8255 č.2 naprogramován a nastaveny bity brány 2C.

Po přijetí přerušení procesorem se uskuteční obsluha přerušení, která zahrnuje i příkaz na nulování záchytného klopného obvodu zdroje přerušení. Pro tento účel se používají instrukce nastavení/nulování individuálního bitu brány 2C. Při zápisu slabiky 2C se stav klopného obvodu nemění. Přehled podává tab. 3.6.

Zdroj přerušení	Stavová slabika přerušení na bráně 2B (viz poznámka 2)	Příkazová slabika pro instrukci OUT CNT2 (viz poznámka 1)		
	Binárně	Hex	Zákaz	Povolení
Čítač 0	0 X X X X X X 1	D1	00	01
Čítač 1	0 X X X X X 1 X	02	02	03
čítač 2	0 X X X X 1 X X	04	04	05
A/D kompar.	0 X X X 1 X X X	08	06	07
EXT 4	0 X X 1 X X X X	10	08	09
EXT 5	0 X 1 X X X X X	20	0A	0B
Brána 1C3	. (viz pozn. 3)		0C	0D

Tab. 3.6 Stavové slabiky a příkazové slabiky pro povolení/zákaz přerušení

Pozn. 1: Příkazová slabika pro povolení/zákaz přerušení se musí zapsat do CNT2, aby se jí vynulovaly klopné obvody čítačů 0, 1, EXT 4 nebo EXT 5. Příkaz pro povolení/zákaz přerušení pro komparátor převodníku A/D nuluje přerušení pouze při automatickém převodu A/D.

Pozn. 2: V hexadecimálních hodnotách stavových slabik jsou ostatní bity (kromě určujícího) doplněny nulami, což ve skutečnosti nemusí být pravda. Používáme je však na testování individuálního zdroje přerušení. Výsledek pro instrukci ANI je vždy nula, jestliže testované přerušení nebylo přijato.

Pozn. 3: Přerušení od 1C3 není zahrnuto do stavové slabiky. Čteme je jako

X X X X 1 X X X

pomocí instrukce IN PORT 1C. Nulujeme jej čtením brány 1A ve strobovaném výstupním režimu (režim 1 a 2). Bránu 1C můžeme též nastavit (nulovat) zapsáním konstanty 06H (07H) do CNT1. Povolení přerušení pro bránu 1C3 nulujeme (nastavujeme) zápisem 0CH (0DH) do CNT2, data na bráně 1C3 se přitom nemění.

3.2.7 Modem pro magnetofon

Modem pro magnetofon je určen pro snímání dat nebo záznam dat na běžný magnetofon (viz odst. 3.1). Pro archivaci je použita frekvenční modulace, při které je úroveň log. 1 zaznamenána několika periodami frekvence 2400 Hz a pro záznam log. 0 je použita frekvence 1200 Hz. Modem pracuje až do rychlosti 500 Bd, ale s ohledem na magnetofon je použito nastavení přenosové rychlosti na 300 Bd. Na kazetu C60 je možno zaznamenat cca 90 kB.

3.2.8 Modem pro připojení dálnopisu T 100

V použité konfiguraci ŠMS-VÚVT je vestavěna přídavná proudová smyčka - modem pro T 100. Oddělení T 100 od ŠMS-VÚVT je zajištěno optočlenem. Pro vysílání dat je použit 1C1, data jsou snímána na 1A0. Dálnopis se připojuje do zásuvky dle obr. 3.2. Potřebujeme-li použít 1C1 a 1A0 pro jiné účely, musíme zrušit příslušné spojky na desce.

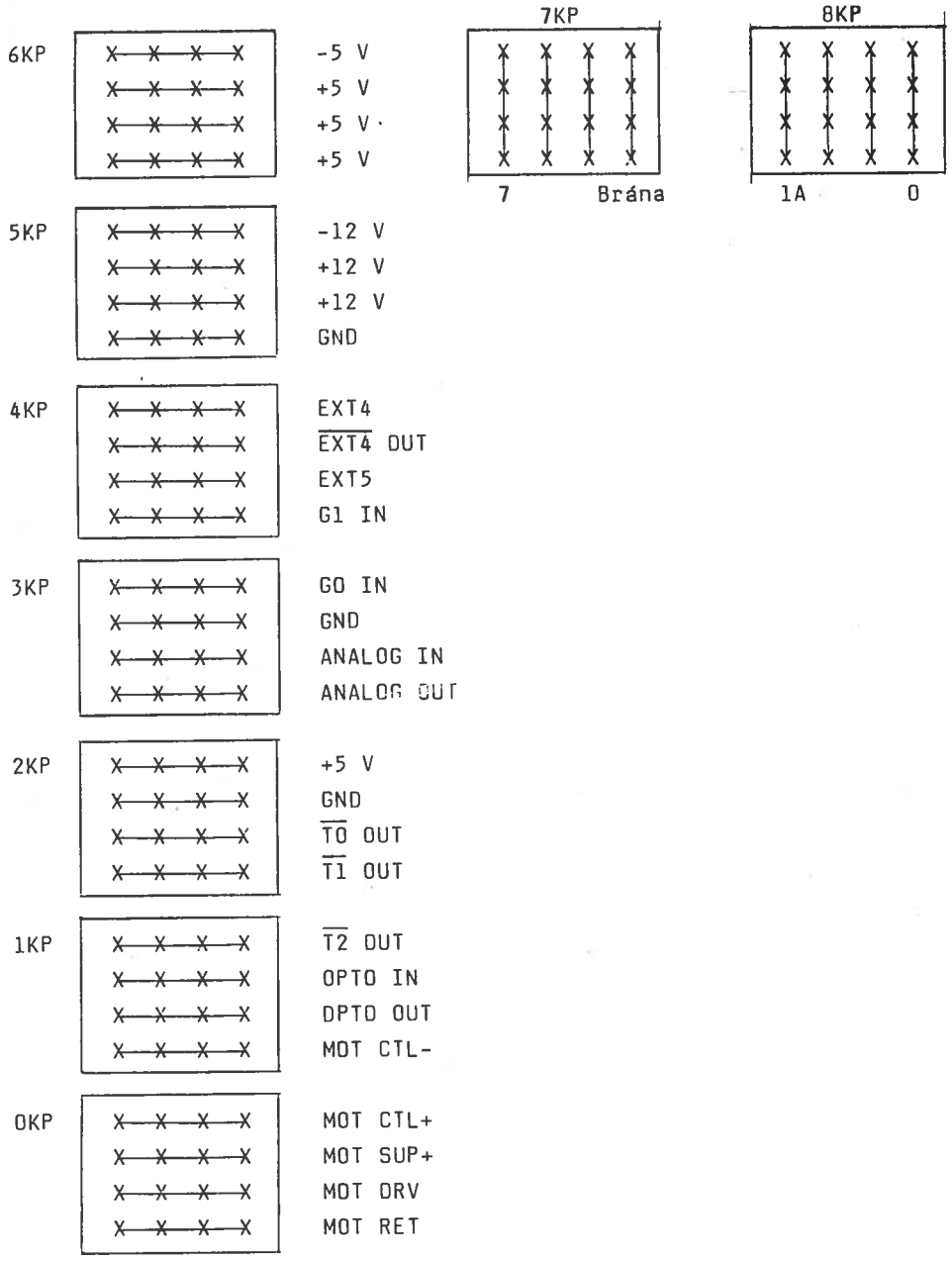
3.2.9 Interfejs RS 232C

Interfejs RS 232C je určen pro připojení terminálů typu SM 7202. Bližší viz lit. /22/.

3.2.10 Systémový konektor

Na systémový 62 pólový konektor jsou vyvedeny systémové sběrnice a další signály umožňující rozšiřování systému. V cvičeních bude systémový konektor zapojen na interfejs umožňující spolupráci ŠMS-VÚVT s řídicím počítačem laboratoře. Při jeho použití postupujeme dle pokynů vedoucího cvičení.

3.2.11 Přiřazení konektorů ŠMS-VÚVT



Obr. 3.7 Rozložení signálů v kontaktových polích 0KP-8KP na ŠMS-VÚVT

4 PŘÍKLADY PROGRAMŮ

Při návrhu programů postupujeme dle zásad uvedených v odst. 1.5, 1.6 a 2.9. Při zavádění jednotlivých programů do ŠMS-VÚVT využíváme poznatků z kap. 3. Základní programové moduly jsou uvedeny v odst. 4.7. Vyčerpávající přehled základních programů je uveden v lit. /27/.

V dalším se zaměříme na realizaci programů vycházejících přímo z možností školního mikropočítače ŠMS-VÚVT (lit. /23/) a používajících, co možná nejvíce, vhodných podprogramů monitoru školního mikropočítače a zejména jeho periférií.

4.1 Přehled podprogramů monitoru ŠMS-VÚVT

Využíváním vhodných podprogramů monitoru můžeme značně zjednodušit aplikační programy, které budeme na školním mikropočítači řešit. Soubor podprogramů monitoru můžeme rozdělit do čtyř skupin:

- podprogramy pro ovládání vstupu z klávesnice
- podprogramy pro ovládání výstupu na displej
- podprogramy pro realizaci časového zpoždění
- pomocné podprogramy

Přehled použitelných podprogramů podává tabulka 4.1 :

Jméno	Funkce	Volací adresa
CLEAR	Mazání celého displeje	0287H
CLRGT	Mazání poloviny displeje	0282H
CLRLP	Mazání paměti	028CH
DELAY	Milisekundové zpoždění	0236H
DELYA	Nastavitelné zpoždění	0238H
DBYTE	Zobrazení slabiky	0295H
DBY2	Zobrazení slabiky	0298H
DISPR	Zobrazení číslice	02A6H
DMEN	Zobrazení slabiky	0294H
DMWD	Zobrazení adresy a dat	02CEH
DWDZ	Zobrazení obsahu HL	02D4H
DWORD	Zobrazení obsahu HL	02D1H
DYPC	Zobrazení PC a instrukce	02CBH
ENTBY	Vstup a zobrazení slabiky	0336H
ENTWD	Vstup a zobrazení dvou slabik	0346H
ERROR	Výpis hlášení Err na displeji	00BCH
GETKY	Vstup klávesy	023DH
MENAB	Povolení monitoru a zobrazení	0222H
SCAN	Jednorázový test klávesnice	0257H
SHLRC	Posuv HL a CY	022FH
SHLRT	Posuv HL	022DH
SHLRZ	Posuv HL včetně Z	022EH

Tab. 4.1 Přehled podprogramů monitoru ŠMS-VÚVT

4.1.1 Podprogramy pro klávesnici a displej

Podprogramy pro vstup z klávesnice jsou: SCAN, GETKY, ENTBY, ENTWD.

Podprogram Scan prohlédne všechny klávesy a zjistí, zda byla některá stisknuta. Pokud ano, uloží její kód do střadače. Klávesy 0 - F mají kód 0 - FH, řídicí (červené) klávesy kód 10H - 17H. Jestliže nebyla stisknuta žádná klávesa, nastaví se CY = 0.

Podprogram GETKY pracuje obdobně jako SCAN s tím rozdílem, že se čeká na stisknutí některé klávesy. Tehdy uloží její kód do střadače a příznak CY = 0 znamená, že byla stisknuta některá z řídicích kláves a ne hexadecimální. Oba podprogramy jsou vhodné pro kód hexa z klávesnice, pro vstup binární hodnoty jsou vhodné podprogramy ENTBY a ENTWD.

Podprogram ENTBY přejímá jednu slabiku z klávesnice (při stisku více hexa kláves jsou platné pouze poslední dvě!). Vstup se ukončí stisknutím některé řídicí klávesy. Zadaná slabika je uložena v registru L, kód řídicí klávesy ve střadači a v registru H je slabika odpovídající stisknutým dvěma klávesám, které předcházely před posledními dvěma klávesami. Stisknuté klávesy jsou průběžně zobrazeny ve dvou pravých pozicích displeje. Příznak Z=1 znamená, že nebyla stisknuta žádná hexa klávesa a tedy, že byla zadána pouze klávesa řídicí.

Podprogram ENTWD je určen pro vstup dvouslabikového slova. Je obdobou podprogramu ENTBY a liší se pouze způsobem zobrazení. Po stlačení čtyř kláves se v pravé části displeje nezobrazí zadaná hodnota, ale obsah paměťové buňky, která je zadanými klávesami adresována. Zadané slovo (dvě slabiky) se průběžně zobrazuje v pravé části displeje. Ostatní je shodné s ENTBY. Podprogramy ENTBY a ENTWD ukládají v registru D současně číslo odpovídající počtu stisknutých hexa kláves.

Podprogram DISPR je určen pro zobrazení jedné hexa číslice. Kód číslice má být uložen v nižších čtyřech bitech střadače, v registrech DE má být uložena žádaná pozice displeje (viz obr. 4.4). Obsah registrů DE se v podprogramu DISPR dekrementuje, a proto opakované použití podprogramu aktivuje další levé pozice displeje.

Podprogram DBY2 zobrazuje slabiku uloženou ve střadači na pozici displeje, kterou určuje obsah registrů DE. Od podprogramu DBY2 jsou odvozeny DBYTE (obsah střadače se zobrazí na dvou pravých pozicích displeje) a DMEM (na posledních dvou pozicích displeje se zobrazí obsah paměťové buňky adresované registry HL).

Podprogram DWD2 zobrazuje obsah registrů HL na pozicích displeje shodně s DBY2.

Na tyto základní podprogramy navazují další:

```
02CB   2AE6B3   DYPC:   LHLD 83E6H      ;naplň HL z paměti 83E6H
02CE   CD9402   DMWD:   CALL DMEM       ;volej DMEM
02D1   11FBB3   DWORD:  LXI  D,83FBH    ;nastav displej
02D4   7D       DWD2:   MOV  A,L        ;začíná DWD2
02D5   CD9B02           CALL DBY2       ;zobraz nižší slabiku
                                        .
                                        .
                        RET             ;návrat
```

Z výše uvedeného je zřejmé, že DWORD zobrazuje obsah HL v levých čtyřech pozicích displeje, DMWD navíc zobrazí obsah paměťové buňky adresované HL a DYPC zobrazí dvě slabiky z adresy 83E6H a též obsah buňky adresované těmito dvěma slabikami. Stejně jako v předchozím se obsah DE dekrementuje o 4.

Příklad 4.1: Využití podprogramů SCAN a DBYTE k zobrazení kódu právě stisknuté klávesy.

```
                                ORG 8200H
8200    CD5702   CP1:   CALL SCAN      ;začátek
8203    CD9502           CALL DBYTE    ;zobrazení kódu klávesy
8206    C30082           JMP  CP1      ;návrat
```

Příklad 4.2: Využití podprogramů SCAN a DWORD pro zobrazení kódu stisknuté klávesy včetně zobrazení příznaků. V levé části displeje se zobrazuje obsah střadače a příznaku.

```
                                ORG 8200H
8200    CD5702   CP2:   CALL SCAN      ;začátek
8203    F5               PUSH PSW      ;úschova střadače a příznaků
8204    E1               POP  H        ;naplnění HL uchovanými daty
8205    CD0102           CALL DWORD    ;zobrazení HL
8208    C30082           JMP  CP2      ;návrat
```

Příklad 4.3: Sestavte program pro postupné rozsvěcení segmentů displeje směrem zleva doprava.

```
                                ORG 8200H
8200    21F883   CP4:   LXI  H,83F8H   ;nastavení levé pozice displeje
8203    3E01             MVI  A,1H     ;nastavení výchozího · tu
8205    77       ULOZ:   MOV  M,A      ;uložení dat
8206    37               STC           ;nastavení CY=1
8207    17               RAL           ;posuv včetně CY
8208    F5               PUSH PSW      ;uchování dat
8209    CD3D02           CALL GETKY    ;čekání na stlačení klávesy
820C    F1               POP  PSW      ;obnovení dat
820D    23               INX  H        ;další pozice displeje
820E    C30582           JMP  ULOZ     ;návrat na úschovu dat střadače
```

Příklad 4.4: Zobrazení písmena "U" na displeji

```
                                ORG 8200H
8200    3E3E     CP5:   MVI  A,3EH     ;naplnění střadače konstantou
                                       ;kódu U
8202    32FF83          STA  83FFH     ;vyslání dat na displej
8205    76              HLT            ;zastavení procesoru
```

Pro mazání obsahu displeje lze použít podprogramy CLRGT, CLEAR a CLRLP. Podprogram CLRLP nuluje paměťové buňky od adresy uložené v HL směrem k nižším adresám, přitom v registru B je počet nulovaných buněk (max. 256). Podprogram CLRLP lze použít všeobecně pro mazání obsahu paměti. podprogram CLEAR nuluje celý displej a podprogram CLRGT pouze čtyři pravé pozice. Bližší je zřejmé z komentovaného výpisu monitoru:

```
0282    0604     CLRGT: MVI  B,4H      ;nastavení N=4
0284    C38902          JMP  0289H     ;přeskok další instrukce
0287    0608     CLEAR: MVI  D,8       ;nastavení N=8
0289    21FF83          LXI  H,83FFH   ;pravá pozice displeje
028C    3600     CLRLP: MVI  M,0H      ;nulování buňky paměti
028E    28              DCX  H         ;snížení ukazatele paměti
```

```
028F    05                DCR  B        ;parametr cyklu dekrementace
0209    C28C02            JNZ  CLRLP    ;opakování cyklu dle reg. B
0293    C9                RET           ;návrat
```

4.1.2 Podprogramy pro realizaci časového zpoždění

Základní podprogramy pro realizaci zpoždění jsou dva DELAY a DELYA. Podprogram DELAY plní obsah střadače konstantou pro zpoždění 1 ms (83H). Na něj navazuje podprogram DELYA tvořící cyklus:

```
0236    3EB3    DELAY:    MVI  A,B3H    ;konstanta 1 ms do střadače
0238    3D      DELYA:    DCR  A        ;dekrementace střadače
0239    C23802            JNZ  DELYA    ;je obsah střadače nulový ?
023C    C9                RET           ;návrat
```

Pokud potřebujeme delší časová zpoždění, můžeme volat podprogram DELYA vícekrát v cyklu:

Příklad 4.5: Příklad programu pro realizaci časového zpoždění v závislosti na obsahu registrů H,L.

```
                                ORG 8000H
8000    CD3602  DELYH:    CALL DELAY    ;základní zpoždění 1 ms
8003    2B                DCX  H        ;dekrementace parametru cyklu
8004    7C                MOV  A,H      ;je obsah HL nulový ?
8005    B5                ORA  L        ;
8006    C20080            JNZ  DELYH    ;jestliže ne, opakování
8009    C9                RET           ;návrat
```

4.1.3 Pomocné podprogramy

K pomocným podprogramům patří SHLRT, SHLRZ a SHLRC, které posouvají obsah registrů HL o jeden bit doprava, přitom SHLRC naplní uvolněný nejvýznamnější bit registru H hodnotou CY, podprogram SHLRZ jej naplní nulou. Podprogram SHLRT navíc nastaví příznak Z=1 je-li HL=0. V registru CY vždy zůstává nejméně významný bit registru L.

K pomocným podprogramům řadíme dále podprogram COPY, který přemístí obsah specifikované části paměti do jiné oblasti a dále podprogram ERROR sloužící pro chybová hlášení (Err) na displeji.

Podprogram ERROR současně nuluje celý displej. Po jeho volání je nutno zastavit procesor, aby se opět nezobrazilo další hlášení monitoru:

```
                                ORG 8200H
8200    CDBC00  CP6:      CALL ERROR    ;chybové hlášení
8203    76                HLT           ;zastavení procesoru
```

4.2 Programování obvodů 8255 v ŠMS-VÚVT

Jak již bylo uvedeno v odst.3.2.2 má školní mikropočítač ve své základní sestavě 3 programovatelné obvody 8255. Ve výše zmíněném odstavci jsou shrnuty i základní údaje o volacích adresách a režimech použitých obvodů 8255.

Ve školním mikropočítači lze bez omezení ve všech režimech použít pouze obvod 8255 č.1, a proto je většina následujících programů na tento obvod zaměřena.

Příklad 4.6: Nastavte obvod 8255 č.1 následovně:

brána P1A - režim 0, výstup
brána P1B - režim 1, vstup
brána P1C - bity P1C4 až P1C7 výstup
P1C3 vstup

Poznámka: 1A0 je současně použit pro proudovou smyčku dálnopisu T100. (Viz odst.3.2.8).

Dle zásad uvedených v odst.2.8.4 je nutno vyslat do řídícího registru 8255 slovo 10000111B = 87H. V souladu s tab.3.1 a tab.3.2 můžeme napsat následující program:

```
                                DRG 8200H
8200    3E87    CP7:    MVI  A,87H     ;řídící slovo 8255
8202    D307            OUT  7         ;řídící slovo do CWR 8255
8204    E7              RST  4         ;návrat do monitoru
```

Po spuštění programu zhasnou všechny výstupní diody LED brány P1A na desce školního mikropočítače.

Příklad 4.7: Navrhněte program pro cyklický posuv rozsvícené diody na bráně P1A.

```
                                DRG 8200H
8200    3E80    CP16:   MVI  A,80H     ;řídicí slovo pro 8255 č.1
                                       ;brána A = výstup
8202    D307            OUT  7         ;odeslání řídicího slova na CWR
8204    D304    N:      OUT  4         ;rozsvícení jednoho bitu na P1A
8206    CD9502          CALL DBYTE     ;zobrazení výstupu P1A na displeji
8209    21E803          LXI  H,1000    ;nastavení parametru zpoždění
820C    F5              PUSH PSW       ;úschova výstupu P1A
820D    CD00B0          CALL DELYH     ;podprogram zpoždění
8210    F1              POP  PSW       ;obnovení stavu P1A
8211    0F              RRC            ;posuv jedničky ve střadači
                                       ;doprava
8212    C304B2          JMP  N         ;opakování cyklu
```

Poznámka: Před spuštěním programu CP16 musíme uložit do počítače i program DELYH z příkladu 4.5.

Příklad 4.8: S použitím obvodu 8255 generujte na reproduktoru tón o frekvenci 500 Hz se střídou 1 : 1 . Výstup na bráně P1A - druhý bit (b1).

```
                                DRG 8200H
8200    3E80    CP19:   MVI  A,80H     ;řídicí slovo 8255-brána A=výstup
```

```
8202    D307          OUT  7        ;řídicí slovo do CWR
8204    DB04    N:    IN   4        ;čtení stavu brány P1A
8206    EE02          XRI  02       ;změna bitu b1 na opačnou hodnotu
8208    D304          OUT  4        ;výstup negovaného bitu
820A    CD3602        CALL DELAY    ;čekání po dobu 1 ms
820D    C30482        JMP  N        ;opakování cyklu
```

Reproduktor připojíme jedním výstupem na kontaktové pole P1A, bit b1, druhým kontaktem na +5 V nebo na GND (nižší intenzita tónu).

Příklad 4.9 : Navrhněte program pro generování tónu jako v příkladu 4.8, ale s měnící se frekvencí.

```
                              ORG 8200H
8200    3E80    CP20:  MVI  A,80H     ;řídicí slovo pro 8255 č.1
8202    D307           OUT  7         ;řídicí slovo do CWR
8204    06FF           MVI  B,0FFH    ;nastavení parametru zpoždění
8206    DB04    N:     IN   4         ;čtení stavu brány P1A
8208    EE02           XRI  02H       ;negace prvního bitu brány P1A1
820A    D304           OUT  4         ;výstup negovaného bitu
820C    78             MOV  A,B       ;příprava parametru zpoždění
820D    CD3802         CALL DELYA     ;čekání na stav střadače
8210    05             DCR  B         ;snížení parametru zpoždění
8211    C30682         JMP  N         ;opakování cyklu
```

4.3 Programy využívající přerušení

Školní mikropočítač nemá, jak je obvyklé u mikroprocesoru 8080A, systém přerušení založen na použití instrukce RST. Vznik přerušení (signál INT procesoru) lze vyjádřit funkcí :

$$INT = P1 \cdot U1 + P2 \cdot U2 + P3 \cdot U3 + P4 \cdot U4 + P5 \cdot U5 + P6 \cdot U6 + P7 \cdot U7$$

kde Pi.....povolení i-tého přerušení
Ui.....výskyt i-té přerušovací události

Princip vytvoření přerušovacích signálů již byl popsán v odst.3.2.2 - 3.2.4 spolu s popisem obvodů 8255 a 8253 a obvody A/D a D/A. Zde pouze zrekapitulujeme přehled přerušovacích událostí - viz tab.4.2 :

Číslo události	Přerušovací zdroj
1	čítač z 8253 č.0
2	čítač z 8253 č.1
3	čítač z 8253 č.2
4	obvody převodníku D/A
5	externí vstup na konektoru ExT4
6	externí vstup na konektoru ExT5
7	bit č.3 brány 1C

Tabulka 4.2 Přehled přerušení ŠMS-VÚVT

Bližší podrobnosti použití zdrojů přerušení nalezneme v lit. /23/. Zde pouze připomeneme, že signál U1 způsobí přerušení typu RST 5, ostatní signály RST 6. Monitor používá pro svoje účely (režim STEP) přerušení RST 7. Jejich činnost je popsána v odst. 3.1.3.

4.4 Programování obvodů 8253 v ŠMS-VÚVT

Základní údaje jsou uvedeny v odst.3.2.3 v tab.3:3 a 3.4.

Příklad 4.10 : Pomocí obvodu 8253 generujte signál 500 Hz (t = 2 ms). Vstup signálu indikujte reproduktorem připojeným na svorky TO a GND.

```
                                ORG 8200H
8200    3E36    CP12:   MVI  A,36H     ;nastavení 8253 do režimu 3
8202    D317            OUT  17H       ;vyslání řídícího slova
8204    3E00            MVI  A,0H      ;nastavení LSB
8206    D314            OUT  14H       ;výstup LSB na TO
8208    3E10            MVI  A,10H     ;nastavení MSB
820A    D314            OUT  14H       ;výstup MSB na TO
820C    E7              RST4           ;návrat do monitoru
```

Poznámka: 1. Při generování využíváme signálu Ø 2 TTL s frekvencí 2048 kHz. Tuto frekvenci musíme vydělit 4096, tj. 1000H pro t = 2 ms. Pro číslo předvolby tedy platí : předvolba = 2048 . $10^3/f_x$ /Hz/.

2. Tón se generuje trvale až do zavedení dalšího řídicího slova (klávesa RESET je neúčinná, neboť 8253 nemá nulovací vstup !)

Příklad 4.11 : Navrhněte program "hudebního nástroje" ovládaného v rozsahu 2 oktáv klávesami 0 - F.

```
                                ORG 8200H
8200    3E36    CP23:   MVI  A,36      ; režim 3,2 slabiky, TO, binární
                                       ; čítání
8202    D317            OUT  17H       ; výstup na 8253
8204    CD5702  N1      CALL SCAN      ; je stisknuta klávesa ?
8207    D20482          JNC  N1        ; jestliže ne, opakování testu
                                       ; ano, vynásobení kódu klávesy 4
820A    87              ADD  A         ; neboť krok položek je 4
820B    87              ADD  A         ;
820C    210080          LXI  H,TAB     ; začátek tabulky
820F    0600            MVI  B,0       ; nulový MSB relativní adresy
8211    4F              MOV  C,A       ; relativní adresa tabulky
8212    09              DAD  B         ; skutečná adresa položky
8213    7E              MOV  A,M       ; vybrání LSB předvolby
8214    D314            OUT  14H       ; výstup LSB na 8253, TO
8216    23              INX  H         ; další část položky
8217    7E              MOV  A,M       ; výběr MSB předvolby
8218    D314            OUT  14H       ; výstup MSB do TO
821A    23              INX  H         ; další část položky
821B    7E              MOV  A,M       ; výběr názvu tónu
821C    32F883          STA  83F8H     ; jeho zobrazení
```

```
821F    23              INX  H          ; další část položky
8220    7E              MOV  A,M        ; výběr druhé části názvu tónu
8221    32F983          STA  83F9H      ; jeho zobrazení
8224    CD5702    N2:   CALL SCAN       ; je ještě stisknutá klávesa ?
8227    DA2482          JC   N2         ; jestliže ano, opakování testu
822A    3E00            MVI  A,0        ; není, smazání displeje
822C    32F883          STA  83F8H      ;
822F    32F983          STA  83F9H      ;
8232    C30082          JMP  CP23       ; opakování cyklu
                      ; následuje tabulka dat

                        ORG 8000H
8000    E8167100  TAB:  DB   0E8H,16H,71H,0
8004    68147D00        DB   68H,14H,7DH,0
8008    2F127700        DB   2FH,12H,77H,0
800C    2911767C        DB   29H,11H,76H,7CH
8010    4A0F3900        DB   4AH,0FH,39H,0
8014    9F0D5E00        DB   9FH,0DH,5EH,0
8018    230C7900        DB   23H,0CH,79H,0
801C    740B7100        DB   74H,0BH,71H,0
8020    340A7D00        DB   34H,0AH,7DH,0
8024    17097700        DB   17H,9,77H,0
8028    950B767C        DB   95H,B,76H,7CH
802C    A5073900        DB   0A5H,7,39H,0
8030    CF065E00        DB   0CFH,6,5EH,0
8034    11067900        DB   11H,6,79H,0
8038    BAD57100        DB   0BAH,5,71H,0
803C    1A057D00        DB   1AH,5,7DH,0
8040    02000000        DB   2,0,0,0
8044    02000000        DB   2,0,0,0
8048    02000000        DB   2,0,0,0
804C    02000000        DB   2,0,0,0
8050    02000000        DB   2,0,0,0
8054    02000000        DB   2,0,0,0
8058    02000000        DB   2,0,0,0
805C    02000000        DB   2,0,0,0
```

Program napřed nastaví obvod 8253 a čeká na předvolbu. Testuje klávesnici pomocí podprogramu SCAN. Jestliže byla stisknuta klávesa, vytvoří čtyřnásobek jejího kódu - vzdálenost položek v tabulce je právě 4 slabiky. Data vyzvedne z tabulky, první dvě slabiky jsou předvolba a druhé kód názvu tónu. Další volání SCAN zjišťuje, zda je klávesa stisknuta. Jestliže ano, vysílání tónu pokračuje, v opačném případě se nuluje displej, nastaví znovu řídicí slovo 8253 a čeká se na stisknutí další klávesy. Změnu tónu můžeme dosáhnout změnou dat v tabulce.

4.5 Programování převodníků A/D a D/A v ŠMS-VÚVT

Popis převodníků je uveden v odst. 3.2.4, blokové schéma na obr. 3.5. Připomeneme, že vstup (8 bitů) převodníku D/A je připojen na bránu P1B a osmibitový spinač je řízen nultým bitem brány P1C. Analogový výstup aktivujeme rozepnutím spínače (P1C0 = 0) a na bránu P1B programem vyšleme potřebný údaj.

Rozsah výstupního napětí nastavíme ručně potenciometrem. Při aplikaci převodníku A/D můžeme použít buď řízení převodu procesorem, nebo automatickým režimem. V obou případech zůstává princip shodný - porovnání vstupního napětí s napětím generovaným převodníkem D/A.

Programové řízení A/D převodu:

Spínač nastavíme do rozepnutého stavu (P1C0 = 0) a program, dle určitého algoritmu, generujeme číslo pro převodník D/A. Výstup komparátoru (třetí bit brány P2B) nám potom udá jejich shodu.

Lze použít dvě metody převodu - **metodu postupných přírůstků** a metodu postupné aproximace.

V prvém případu generujeme postupně čísla (8 bitů) pro D/A od nižší hodnoty (0 0 0 0 0 0 0 0 B) inkrementací po nejvyšší hodnotu (1 1 1 1 1 1 1 1 B). Převod je ukončen po hlášení komparátoru o shodě nebo nejbližší vyšší hodnotě. Tento způsob je sice jednoduchý, ale zdlouhavý, neboť v nejhorším případu musíme generovat až 255 čísel.

Vstupní analogový rozsah závisí na zesílení vstupního zesilovače (trimr ANALOG IN). Zesílení lze měnit v rozsahu 0,277 až 1 a tím dosáhnout vstupních napětí 0V až 2,55 V/0,277.

Druhý způsob, **metoda postupných aproximací**, je založen na principu půlení intervalu, kdy rozhodujeme v které polovině rozsahu bude hledaná hodnota. Nejprve necháme rozhodnout, v které polovině rozsahu je vstupní napětí, poté v které čtvrtině atd. Přitom se binární číslo tvoří směrem od nejvyšších bitů k nejnižším (po každém kroku je znám jeden bit).

Automatické řízení převodu A/D:

Zde je použita metoda přírůstková. V tomto případě nastavíme spinač v převodníku D/A do sepnuté polohy a bránu P1B do vstupního režimu. Na začátku převodu se čítač v převodníku D/A nuluje signálem R3. Čítač T2 z obvodu 8253 generuje pravidelné impulsy a čítač postupně inkrementuje. Děj se opakuje až do vyhodnocení komparátorem, že vstupní napětí je nižší než napětí generované převodníkem D/A. Tím se nastaví P1C0 = 1 a převod je ukončen. Kód napětí, uložený v D/A přečteme pomocí brány P1B.

Automatický převod je výhodný zejména tehdy, jestliže necháme procesor zpracovávat jiný program a skončení převodu oznámíme pomocí přerušení nebo, v průběhu hlavního programu, periodicky testujeme bit P2B3, kde hodnota log. 1 indikuje ukončení A/D převodu (v případě přerušení čteme stav bran P1B). Pro tento účel je třeba naprogramovat T2 v obvodu 8253 na vhodnou frekvenci. Nulovací signál R3 lze generovat programově vysláním 0 0 0 0 0 1 1 X B na adresu 0FH. Tím zároveň povolíme, nebo zakážeme, přerušení od ukončení převodu A/D.

Při programování nelze připustit, aby byl P1B ve výstupním režimu a současně P1C0 =1. Proto jsou nepřístupná i všechna řídicí slova obvodu 8255 č.1, která nastavují P1B do výstupníhho režimu a současně P1C (0 - 3) do režimu vstupního (tj. 81H, 89H, 91H, 99H).

Příklad 4.12: Navrhněte program převádějící číslo, zadané z klávesnice, na analogové napětí.

```
                                ORG 8200H
8200    3E90    CP29:   MVI  A,90H     ;řídicí slabika, brána B = vý-
                                       ;stup, brána C = výstup
8202    D307            OUT  7         ;vyslání řídicí slabiky na 8255
                                       ;č.1
8204    CD3603  CYKL:   CALL ENTBY     ;vstup slabiky z klávesnice
8207    7D              MOV  A,L       ;přesun zadané slabiky
8208    D305            OUT  5         ;výstup slabiky do převodníku D/A
820A    C3048 2         JMP  CYKL      ;opakování cyklu
```

Vstupní napětí kontrolujeme měřícím přístrojem zapojeným mezi svorky GND a ANALOG OUT.

Příklad 4.13: Sestavte program pro převodník A/D, řízený programem metodou přírůstků.

```
                        ORG 8000H
8000    C5      AD1:    PUSH B         ;uchování registrů B,C
8001    97              SUB  A         ;nulování střadače
8002    D305    CYKL:   OUT  5         ;výstup na převodníku D/A
8004    47              MOV  B,A       ;uschování vyslaného kódu
8005    3E20            MVI  A,20H     ;kód zdržení 240 µs pro ustálení
                                       ;dat
8007    CD3802          CALL DELYA     ;začátek programového zpoždění
                                       ;podle obsahu střadače
800A    DB0D            IN   0DH       ;vstup z P2V (pro P2B3)
800C    E608            ANI  00001000B ;maska pro P2B3
800E    C21780          JNZ  HVT       ;je-li P2B3 = 1 skok na ukončení
                                       ;převodu
8011    78              MOV  A,B       ;obnova kódu čísla
8012    C601            ADI  1         ;inkrementace testovaného kódu
8014    D20280          JNC  CYKL      ;a opakování jestliže nedošlo
                                       ;k přetečení
8017    78      HTV:    MOV  A,B       ;převzetí uchovaného výsledného
                                       ;kódu
8018    C1              POP  B         ;obnovení stavu registrů B,C
8019    C9              RET            ;návrat z podprogramu
```

Podprogram převodu A/D používá registr B, a proto jej vždy chráníme uložením do zásobníku. Činnost uvedeného programu AD1 můžeme sledovat v režimu STEP (zde je vhodné vypustit podprogram DELYA).

Příklad 4.14: Sestavte program, který zobrazí naměřenou hodnotu analogového napětí z příkladu 4.13 v pravé části displeje.

```
                                ORG 8200H
8200    3E90    CP30:   MVI  A,90H      ;řídicí slovo pro 8255 č.1, brána
                                        ;B = výstup, brána C = výstup
8202    D307            OUT  7          ;výstup řídicího slova do 8255
8204    3E92            MVI  A,92H      ;řídicí slovo pro nastavení 8255
                                        ;č.1 brána B = vstup
8206    D30F            OUT  0FH        ;výstup řídicího slova
8208    CD0080  AD:     CALL AD1        ;volání převodu A/D
820B    CD9502          CALL DBYTE      ;zobrazení naměřené hodnoty
820E    C30882          JMP  AD         ;opakování měření
```

Příklad 4.15: Sestavte program převodu A/D využívající metody postupné aproximace. Pro jeho ověření použijte program CP30 z příkladu 4.14. Záporná napětí budou zobrazena jako 00H, přepětí kódem FFH.

```
                                ORG 8000H
8000    C5      AD2:    PUSH B          ;uchování registrů B,C
8001    97              SUB  A          ;nulování střadače
8002    0680            MVI  B,10000000B ;nastavení nejvyššího bitu kódu
8004    B0      CYKL:   ORA  B          ;přepsání nastaveného bitu
                                        ;do střadače
8005    D305            OUT  5          ;vyslání kódu do D/A
8007    4F              MOV  C,A        ;úschova střadače do registru C
8008    3E30            MVI  A,30H      ;zpoždění pomocí podprogramu
800A    CD3802          CALL DELYA      ;DELYA asi 300 µs
800D    DB0D            IN   0DH        ;vstup brány P28 (kvůli P283)
800F    E608            ANI  08H        ;určení stavu P283
8011    79              MOV  A,C        ;obnovení kódu ve střadači
8012    CA1780          JZ   OBSK       ;přeskočení změny bitu
8015    A8              XRA  B          ;změna naposled nastaveného bitu
8016    4F              MOV  C,A        ;úschova nového kódu
                                        ;do registru C
8017    78      OBSK:   MOV  A,B        ;připravení obsahu B na posun
8018    0F              RRC             ;posun o bit doprava
8019    DA2180          JC   HTV        ;je-li převod ukončen skok
801C    47              MOV  B,A        ;jinak návrat posunu do reg. B
801D    79              MOV  A,C        ;obnovení měřené hodnoty
801E    C30480          JMP  CYKL       ;opakování cyklu
8021    79      HTV:    MOV  A,C        ;přesun výsledku do střadače
8022    C1              POP  B          ;obnova registrů B,C
8023    C9              RET             ;návrat z podprogramu
```

Příklad 4.16: Navrhněte program pro generování sinusového, obdélníkového, pilového a trojúhelníkového průběhu pomocí převodníku D/A. Tvar signálu lze volit z klávesnice následovně:

1 ... sinusový průběh 3 ... pilový průběh
2 ... trojúhelníkový průběh 4 ... obdélníkový průběh

Stisknutí jakékoliv jiné klávesy způsobí chybové hlášení a zastavení programu. Kontrolu činnosti programu proveďte: a) osciloskopem
b) posouzením změny tónu reproduktorem

```
                                ORG 8200H
8200    3E90     CP21:  MVI  A,90H      ;řídicí slovo pro 8255 č.1,
                                        ;brány B,C = výstup
8202    D307            OUT  7         ;vyslání řídicího slova
8204    210080          LXI  H,TABL    ;začátek tabulek obdélníku
8207    220081          SHLD 8100H     ;úschova aktuální adresy tabulky
820A    0620     N1:    MVI  B,32      ;počet vzorků v tabulce
820C    2A0081          LHLD 8100H     ;převzetí adresy tabulky
820F    7E       NO:    MOV  A,M       ;výběr položky z tabulky
8210    0305            OUT  5         ;a její vyslání do D/A
8212    23              INX  H         ;další položka tabulky
8213    05              DCR  B         ;jsou všechny položky tabulky
                                        ;přečteny ?
8214    C20F82          JNZ  NO        ;jestliže ne, opakování výběru
                                        ;z tabulky
8217    CD5702          CALL SCAN      ;jinak testování klávesnice
821A    D20A82          JNC  N1        ;nebyla stisknuta žádná klávesa,
                                        ;a proto opakování
821D    0604            MVI  B,4       ;4 možnosti vstupu z klávesnice
821F    210080          LXI  H,TABL    ;adresa první tabulky-obdélníky
8222    112000          LXI  D,32      ;vzdálenost tabulek
8225    220081   N2:    SHLD 8100H     ;odložení adresy aktuální tabulky
8228    B8              CMP  B         ;je v registru B kód
                                        ;stisknuté klávesy ?
8229    CA0A82          JZ   N1        ;jestliže ano, opakování výběru
                                        ;z tabulky
822C     19             DAD  D         ;jestliže ne, adresa další tabul-
                                        ;ky
822D    05              DCR  B         ;snížení parametru cyklu
822E    C22582          JNZ  N2        ;a jeho opakování
8231    CDBC00          CALL ERROR     ;když program došel až sem,
                                        ;znamená to, že byla stisknuta
                                        ;chybná klávesa
8234    76              HLT            ;a proto zastavení programu
                 ;následují tabulky průběhu signálu
                                ORG 8000H
8000    FFFFFFFF TABL:  DB   0FFH,0FFH,0FFH,0FFH  ;tabulka obdélníků
8004    FFFFFFFF        DB   0FFH,0FFH,0FFH,0FFH
8008    FFFFFFFF        DB   0FFH,0FFH,0FFH,0FFH
800C    FFFFFFFF        DB   0FFH,0FFH,0FFH,0FFH
8010    00000000        DB   0,0,0,0
8014    00000000        DB   0,0,0,0
8018    00000000        DB   0,0,0,0
801C    00000000        DB   0,0,0,0
8020    00081018 TAB2:  DB   0,8,10H,18H          ;tabulka pily
8024    20283038        DB   20H,28H,30H,38H
8028    40485058        DB   40H,48H,50H,58H
802C    60687078        DB   60H,68H,70H,78H
8030    80889098        DB   80H,88H,90H,98H
8034    A0A8B0B8        DB   0A0H,0A8H,0B0H,0B8H
```

```
8038    C0C8DDD8        DB   0C0H,0C8H,0D0H,0D8H
803C    E0E8F0F8        DB   0E0H,0E8H,0F0H,0F8H
8040    00102030 TAB3:  DB   0,10H,20H,30H        ;trojúhelník
8044    40506070        DB   40H,50H,60H,70H
8048    8090A0B0        DB   80H,90H,0A0H,0B0H
804C    C0D0E0F0        DB   0C0H,0D0H,0E0H,0F0H
8050    FFF0E0D0        DB   0FFH,0F0H,0E0H,0D0H
8054    C0B0A090        DB   0C0H,0B0H,0A0H,90H
8058    80706050        DB   80H,70H,60H,50H
805C    40302010        DB   40H,30H,20H,10H
8060    809981C7 TAB1:  DB   80H,99H,0B1H,0C7H     ;sinus
8064    DBEBF6FD        DB   0DBH,0EBH,0F6H,0FDH
8068    FFFDF6EB        DB   0FFH,0FDH,0F6H,0EBH
806C    0BC7B199        DB   0DBH,0C7H,0B1H,99H
8070    80675439        DB   80H,67H,54H,39H
8074    25160A05        DB   25H,16H,0AH,5H
8078    00050A16        DB   0,5H,0AH,16H
807C    25395467        DB   25H,39H,54H,67H
```

4.6 Programy s elektromotorem

V základní sestavě ŠMS-VÚVT je k dispozici elektromotor na jehož hřídeli je optický kotouč pro přerušování světla diody LED a tím umožňující zpětné měření počtu otáček. Ovládání elektromotoru je možné z výkonového stupně ŠMS-VÚVT. Řízení rychlosti otáčení motorku je možné buď stejnosměrným napětím z D/A převodníku, nebo impulsně programem, případně s použitím časovače 8253.

Příklad 4.17: Navrhněte program pro řízení rychlosti otáček elektromotoru v závislosti na kódu stisknuté klávesy.

```
                                ORG 8200H
8200    3E92    CP25:   MVI  A,92H        ;inicialisace bran:
8202    D307            OUT  7            ;P1C = výstup
8204    3E02    CYKL:   MVI  A,00000010B  ;nastavení P1C1 = 0
8206    D307            OUT  7            ;
8208    97              SUB  A            ;nulování střadače
8209    FE00    CP:     CPI  0            ;test parametru cyklu
820B    F5              PUSH PSW          ;úschova parametru cyklu
820C    C21382          JNZ  DELA         ;přskok nastavení P1C1 = 1
820F    3E03            MVI  A,00000011B  ;nastavení P1C1 = 1
8211    D307            OUT  7            ;
8213    3E10    DELA:   MVI  A,10H        ;nastavení zpoždění cca 110 µs
8215    CD3802          CALL DELYA        ;podprogram zpoždění
8218    F1              POP  PSW          ;obnova parametru cyklu
8219    3C              INR  A            ;inkrementace parametru cyklu
821A    C20982          JNZ  CP           ;není-li konec cyklu, opakovámí
8210    CD5702          CALL SCA          ;vstup klávesy
8220    D20482          JNC  CYKL         ;nebyla-li stisknuta nová kláve-
                                          ;sa pokračování novým cyklem se
                                          ;starou hodnotou
```

```
8223    07          RLC          ;posuv bitů o 4
8224    07          RLC          ;
8225    07          RLC          ;
8226    07          RLC          ;
8227    320A82      STA  CP+1    ;uložení nové rozhodovací hodnoty
822A    C30482      JMP  CYKL    ;pokračování novým cyklem
```

Program nastaví P1C na výstupní režim pro řízení diody LED optočlenu bitem P1C1. (MOT CTL+ připojen na +5V). Řízení začíná nastavením plného napětí motoru (P1C1=0) a poté běží cyklus 256 krát a dle stavu střadače se buzení motoru odpojí v okamžiku, kdy se zjistí shoda parametru cyklu a hodnoty zadané z klávesnice (instrukce CPI na adrese 8209H). Trvání cyklu je upraveno programovým zdržením DELYA. Po ukončení cyklu se testuje klávesnice. Není-li stisknuta žádná klávesa, začíná nový cyklus se shodnou hodnotou rozhodování pro vypnutí buzení, v opačném případě se kód klávesy posune o 4 místa doleva (dosažení rozsahu hodnot 00-F0H s krokem 10H) a takto upravená hodnota se uloží přímo do instrukce rozhodování na adrese 8209H.

Podobným způsobem je možno sestavit řadu dalších programů pro řízení rychlosti otáčení včetně smyčky zpětné vazby přes snímání stavu otáček hřídele.

4.7 Standardní programy

Jak již bylo uvedeno, podává vyčerpávající přehled základních programů lit. /27/, případně snáze dostupná lit. /39/, /29/.
V dalším uvedeme některé z nich (lit./27/), upravené pro paměťovou lokaci ŠMS-VÚVT. V přehledu jsou uvedeny nejčastěji používané programy, bližší viz citovaná literatura.

Příklad 4.18: Nulování paměťového místa.
Vynulujte paměťové místo s adresou 8000H.

```
                                ORG 8200H
8200    97          START: SUB  A        ;nulování střadače
8201    320080             STA  8000H    ;nulování RWM
8204    C33000      STOP:  JMP  STOP     ;návrat do monitoru
```

Příklad 4.19: Jedničkový doplněk
Vytvořte doplněk k číslu na adrese 8000H a výsledek uložte na adresu 8001H.

výchozí data (8000) = 6A
výsledek (8001) = 95

```
                                ORG 8200H
8200    3A0080      START: LDA  8000H    ;čtení dat RWM
8203    2F                 CMA           ;jedničkový doplněk
8204    320180             STA  8001H    ;uložení dat
8207    C33000      STOP:  JMP  STOP     ;návrat do monitoru
```

Příklad 4.20: Osmibitové sčítání
Sečtěte čísla na adrese 8000H a 8001H a výsledek uložte na adresu 8002H.

výchozí data (8000) = 38
(8001) = 2B
výsledek (8002) = 63

```
                                ORG 8200H
8200    210080    START: LXI  H,800H     ;adresa prvního operandu do HL
8203    7E               MOV  A,M       ;první operand
8204    23               INX  H         ;adresa druhého operandu
8205    86               ADD  M         ;součet
8206    23               INX  H         ;adresa pro výsledek
8207    77               MOV  M,A       ;uložení součtu
8208    C33000    STOP:  JMP  STOP      ;návrat do monitoru
```

Příklad 4.21: Posuv o jeden bit vlevo

Posuňte obsah paměti s adresou 8000H o jeden bit vlevo a prázdnou pozici bitu b0 vynulujte. Výsledek uložte na adresu 8001H.

výchozí data (8000) = 6F
výsledek (8001) = DE

```
                                ORG 8200H
8200    3A0080    START: LDA  8000H     ;čti data
8203    87               ADD  A         ;posun dat vlevo
8204    320180           STA  8001      ;uložení dat do RWM
8207    C33000    STOP:  JMP  0030      ;návrat do monitoru
```

Příklad 4.22: Maskování vyšších 4 bitů slova

Přesuňte nižší čtyři bity slova z adresy 8000H na adresu 8001H. Vyšší čtyři bity slova na adrese 8001H vynulujte.

výchozí data (8000) = B8
výsledek (8001) = 08

```
                                ORG 8200H
8200    3A0080    START: LDA  8000H     ;čti data
8203    E60F             ANI  00001111B ;maskování 4 vyšších bitů
8205    320180           STA  8001H     ;uložení dat do RWM
8208    C33000    STOP:  JMP  STOP      ;návrat do monitoru
```

Příklad 4.23: Určení většího ze dvou čísel

Porovnejte čísla na adresách 8000H a 8001H a větší z nich uložte na adresu 8002H. Uložená čísla jsou binární čísla bez znaménka.

výchozí data (8000H) = 6F
(8001H) = 1A
výsledek (8002H) = 6F

```
                                ORG 8200H
8200    210082           LXI  H,8200H   ;nastavení adresy
8203    7E               MOV  A,M       ;a přečtení prvního operandu
8204    23               INX  H         ;druhý operand
8205    BE               CMP  M         ;je druhý operand větší ?
8206    020A82           JNC  DALE      ;není
8209    7E               MOV  A,M       ;je, a proto druhý do střadače
820A    23        DALE:  INX  H         ;nastavení adresy paměti
820B    77               MOV  M,A       ;uložení výsledku
820C    C33000    STOP:  JMP  0030      ;návrat do monitoru
```

Poznámka: Instrukce CMP M porovnává obsah střadače a paměti a nastaví:

Z = 1 pro A = M
Z = 0 pro A ≠ M
CY = 1 pro A < M
CY = 0 pro A ≥ M

Příklad 4.24: Sestavte program pro násobení dvou osmibitových čísel. Výchozí data jsou uložena na adrese 8000H a 8001H, výsledek uložte na adresy 8002H a 8003H:

výchozí data (8000) = 02
(8001) = 07
výsledek (8002) = 0EH
(8003) = 00

```
                                DRG 8200H
8200    210080    ZAC:   LXI  H,8000H     ;adresa paměti
8203    5E               MOV  E,M        ;čtení násobence
8204    1600             MVI  0,0H       ;rozšíření na 16 bitů
8206    23               INX  H          ;nová adresa
8207    7E               MOV  A,M        ;násobitel
8208    210000           LXI  H,0H       ;součin 0
820B    0608             MVI  8,8H       ;nastavení počitadla cyklu
820D    29        MULT:  DAD  H          ;posun o 1 bit vlevo
820E    87               ADD  A          ;posun násobitele do CY
820F    D21382           JNC  TEST       ;je testovaný bit = 1 ?
8212    19               DAD  D          ;ano
8213    05        TEST:  DCR  B          ;
8214    C20D82           JNZ  MULT       ;konec násobení ?
8217    220280           SHLD 8002H      ;uložení výsledku
821A    C33000    STOP:  JMP  0030       ;návrat do monitoru
```

Příklad 4.25: Dělte šestnáctibitové číslo (bez znaménka) uložené na adrese 8000H (LSB) a 8001H (MSB) osmibitovým číslem uloženým na adrese 8002H. Čísla jsou upravena tak, že MSB dělence i dělitele jsou nulové a číslo na adrese 8002H je větší než číslo na adrese 8001H, podíl tedy může být vyjádřen osmibitovým číslem. Výsledek uložte na adresu 8003H, zbytek na adresu 8004H.

1) výchozí data (8000) = 40H
(8001) = 00
(8002) = 08
výsledek (8003) = 08
(8004) = 00
(64 : 8) = 8

2) výchozí data (8000) = 60H
(8001) = 32H
(8002) = 47H
výsledek (8003) = B5H
(8004) = 3AH
(12909 : 71 = 181 + 58/71)

```
                                ORG 8200H
8200    2A0080    ZAC:   LHLD 8000H       ;čtení dělence
8203    3A0280           LDA  8002H      ;čtení dělitele
8206    4F               MOV  C,A        ;
8207    0608             MVI  B,8        ;nastavení počítadla cyklu
8209    29        DEL:   DAD  H          ;posuv dělence a podílu o 1 bit
                                         ;vlevo (b = 0)
820A    7C               MOV  A,H        ;
```

```
820B    91                SUB  C          ;8 zbylých MSB dělence dělitelné
820C    DA1182            JC   DALE       ;není, pokračuj dalším cyklem
820F    67                MOV  H,A        ;ano, odečtení dělitele
8210    2C                INR  L          ;přičtení 1 k podílu
8211    05        DALE:   OCR  8          ;
8212    C20982            JNZ  DEL        ;konec dělení ? (počítadlo = 0)
8215    220380            SHOL 8003H      ;uložení výsledku
8218    C33000    STOP:   JMP  0030       ;návrat do monitoru
```

Příklad 4.26 : Převeďte obsah paměťové buňky s adresou 8000H do kódu ASCII. Výsledek uložte na adresu 8001H.

výchozí data (8000) = 0CH
výsledek (8001) = 43H = C´

```
                                ORG 8200H
8200    3A0080    ZAC:    LDA  8000H      ;přečtení dat
8203    FE0A              CPI  10         ;jsou data  10?
8205    0A0A82            JC   ASCZ       ;
8208    C607              ADI  'A'-'9'-1  ;ano, přičti posuv pro písmena
820A    C630      ASCZ:   ADI  '0'        ;ne, přičti '0'ASCII
820C    320180            STA  8001H      ;uložení výsledku
820F    C33000    STOP:   JMP  0030       ;návrat do monitoru
```

Poznámka : Připočtením kódu 0´ASCII převedeme správně pouze číslice 0 až 9. Zbývajících 7 znaků při převodu hexadecimálních čísel A až F musíme překlenout.

Příklad 4.27 : Převeďte kód ASCII znaku z adresy 8000H na dekadické číslo a uložte je na adresu 8001H.

výchozí data (8000) = 36H = 6´
výsledek (8001) = 06

```
                                ORG 8200H
8200    06FF      ZAC:    MVI  B,0FFH     ;uložení značky chyby (FFH)
8202    3A0080            LDA  8000H      ;data do střadače
8205    D630              SUI  '0'        ;odečtení '0'ASCII
8207    DA1082            JC   STORE      ;jsou-li data  než ASCII '0'
                                          ;pak CY = 1
820A    FE0A              CPI  10         ;je výsledek  10D?
820C    D21082            JNC  STORE      ;jestliže ano, CY = 0
820F    47                MOV  B,A        ;
8210    78        STORE:  MOV  A,B        ;
8211    320180            STA  8001H      ;uložení výsledku
8214    C33000    STOP:   JMP  0030       ;návrat do monitoru
```

Poznámka : První rozhodnutí o převodu se uskuteční odečtením kódu '0' ASCII (data ASCII 0´), druhé rozhodování instrukcí CPI.

Příklad 4.28 : Sestavte podprogram pro převod hexadecimálního kódu na kód ASCII. Výchozí data jsou uložena na adrese 8000H, hlavní program na adrese 8200H data vyzvedne, vyvolá podprogram z adresy 8100H a výsledek uloží na adresu 8001H. Princip převodu použijte shodný s příkladem 4.26.

1) výchozí data (8000) = 0EH
   výsledek (8001) = 45H = 'E'

2) výchozí data (8000) = 04
   výsledek (8001) = 34H = '4'

```
                        ORG 8200H
                                          ;zde je uložen hlavní program
8200    319082   ZAC:   LXI  SP,8290H     ;vytvoření zásobníku na adrese
                                          ;8290H
8203    3A0080           LDA  8000H       ;data do střadače
8206    CD0081           CALL HEXASC      ;zavolání podprogramu převodu
                                          ;kódu
8209    320180           STA  8001H       ;uložení výsledku
820C    C33000   STOP:   JMP  0030        ;návrat do monitoru

                 ;podprogram HEXASC slouží pro převod hexadecimálního
                 ;čísla,uloženého ,ve střadači, na znak ASCII, výsledek
                 ;ponechá ve střadači
                 ;
                 ;vstupy :  A - hexadecimální číslo
                 ;výstupy : A - znak ASCII
                 ;ovlivňuje : A,F
                 ;volá podprogramy : žádné
                 ;
                        ORG 8100H
8100    FE0A     HEXASC:CPI  10D          ;jsou data  10D?
8102    DA0781           JC   POSUN       ;
8105    C607             ADI  'A'-'9'-1   ;ano, připočtení posunu pro
                                          ;písmena
8107    C630     POSUN:  ADI  '0'         ;přičtení posunu pro ASCII
8109    C9               RET              ;návrat z podprogramu
```

P Ř Í L O H A

5. Instrukční soubor 8080A

V této kapitole je obsaženo přehledné dělení instrukčního souboru mikroprocesoru 8080A rozdělené, v návaznosti na tabulky odst. 2.7, dle provádění operací. Cílem je usnadnit zapamatování jednotlivých zkratek jazyka symbolických adres mikroprocesoru 8080A zdůrazněním způsobu operací s osmibitovými a šestnáctibitovými daty.

5.1 Skupina přesunových instrukcí

Neovlivňují žádné příznakové bity.

Osmibitové přesunové instrukce

MOV r1,r2	MOVE REGISTER
MOV r,M	MOVE FROM MEMORY
MOV M,R	MOVE TO MEMORY
MVI r,data	MOVE IMMEDIATE
MVI M,data	MOVE TO MEMORY IMMEDIATE
LDA,adr	LOAD ACCUMULATOR DIRECT
STA,adr	STORE ACCUMULATOR DIRECT
LDAX rp	LOAD ACCUMULATOR INDIRECT
STAX rp	STORE ACCUMULATOR INDIRECT

Šestnáctibitové přesunové instrukce

XCHG	EXCHANGE H AND L WITH D AND E
LHLD adr	LOAD H AND L DIRECT
SHLD adr	STORE H AND L DIRECT
LXI rp	LOAD IMMEDIATE REGISTER PAIR

5.2 Instrukce měnící obsah registru nebo paměti

Ovlivňují všechny příznakové bity s vyjímkou CY.

Operace s osmibitovými daty

INR r	INCREMENT REGISTER
INR M	INCREMENT MEMORY
DCR r	DECREMENT REGISTER
DCR M	DECREMENT MEMORY

Operace s šestnáctibitovými daty

INX rp	INCREMENT REGISTER PAIR
DCX rp	DECREMENT REGISTER PAIR

5.3 Skupina aritmetických operací

Přičtení osmibitového operandu k obsahu střadače

ADD r	ADD REGISTER TO A
ADD M	ADD MEMORY TO A
ADC r	ADD REGISTER TO A WITH CARRY
ADC M	ADD MEMORY TO A WITH CARRY
ADI data	ADD IMMEDIATE
ACI data	ADD IMMEDIATE TO A WITH CARRY

Odečtení osmibitového operandu od obsahu střadače

SUB r	SUBTRACT REGISTER FROM A
SUB M	SUBTRACT MEMORY FROM A
SBB r	SUBTRACT REGISTER FROM A WITH BORROW
SBB M	SUBTRACT MEMORY FROM A WITH BORROW
SUI data	SUBTRACT IMMEDIATE FROM A
SBI data	SUBTRACT IMMEDIATE FROM A WITH BORROW

Přičítání šestnáctibitových operandů
k registrovému páru HL
Ovlivňuje pouze bit CY.

DAD rp	DOUBLE ADD REGISTER PAIR TO HL

Porovnání obsahu střadače
s obsahem registrů a paměti
Ovlivňuje všechny příznakové bity.

CMP r	COMPARE REGISTER WITH A
CMP M	COMPARE MEMORY WITH A
CPI data	COMPARE IMMEDIATE WITH A

5.4 Skupina logických operací

Podskupina logických součinů
Ovlivňuje všechny příznakové bity.(CY=0, AC=0)

ANA r	LOGICAL AND WITH ACCUMULATOR
ANA M	LOGICAL AND MEMORY WITH ACCUMULATOR
ANI data	AND IMMEDIATE WITH ACCUMULATOR

Podskupina logických součtů
ovlivňuje všechny příznakové bity. (CY=0, AC=0)

ORA r	INCLUSIVE OR WITH ACCUMULATOR
DRA M	INCLUSIVE OR MEMORY WITH ACCUMULATOR
ORI data	INCLUSIVE OR IMMEDIATE WITH ACCUMULATOR

Podskupina exkluzivních logických součtů
Ovlivňuje všechny příznakové bity.
(CY=0, AC=0)

XRA r	EXCLUSIVE OR WITH ACCUMULATOR
XRA M	EXCLUSIVE OR MEMORY WITH ACCUMULATOR
XRI data	EXCLUSIVE OR IMMEDIATE WITH ACCUMULATOR

Podskupina speciálních logických instrukcí

DAA	DECIMAL ADJUST ACCUMULATOR
CMA	COMPLEMENT ACCUMULATOR
STC	SET CARRY
CMC	COMPLEMENT CARRY

5.5 Skupina rotací a posunů
Ovlivňuje pouze bit CY.

Rotace bez přenosového bitu

RLC	ROTATE ACCUMULATOR LEFT
RRC	ROTATE ACCUMULATOR RIGHT

Rotace včetně přenosového bitu

RAL	ROTATE LEFT THROUGH CARRY
RAR	ROTATE RIGHT THROUGH CARRY

5.6 Operace se zásobníkem a ukazatelem zásobníku
Neovlivňuje žádné příznakové bity.

Uložení do zásobníku

PUSH rp	PUSH REGISTER PAIR
PUSH PSW	PUSH PROCESSOR STATUS WORD

Vyzvednutí ze zásobníku

POP rp	POP REGISTER PAIR
POP PSW	POP PROCESSOR STATUS WORD

Nastaveni obsahu ukazatele zásobníku(SP)
a zásobníku

!!! Lze užít i instrukci LXI rp !!!!

SPHL	MOVE HL TO SP
XTHL	EXCHANGE TOP OF STACK WITH H AND L

5.7 Skupina instrukcí větvících program

Neovlivňují žádné příznakové bity.

Instrukce skoků

JMP adr	JUMP UNKONDITIONAL
J "ccc"adr	JUMP ON CONDITION
Z=0	JNZ
Z=1	JZ
CY=0	JNC
CY=1	JC
P=0	JPO
P=1	JPE
S=0	JP
S=1	JM

Instrukce volání podprogramu

CALL adr	CALL UNCONOITIONAL
C "ccc"adr	CALL ON CONDITION
Z=0	CNZ
Z=1	CZ
CY=0	CNC
CY=1	CC
P=0	CPO
P=1	CPE
S=0	CP
S=1	CM

Instrukce Přerušení

RST n	RESTART
n= 0 - 7	

Instrukce návratu z podprogramu

RET	RETURN UNCONDITIONAL FROM SUBROUTINE
R "ccc"adr	RETURN ON CONDITION
Z=0	RNZ
Z=1	RZ
CY=0	RNC
CY=1	RC
P=0	RPO
P=1	RPE
S=0	RP
S=1	RM

5.8 Instrukce vstupu a výstupu

Neovlivňují žádné příznakové bity.

IN port	INPUT FROM PORT
OUT port	OUTPUT TO PORT

5.9 Řídicí instrukce

Neovlivňují žádné příznakové bity.

NOP	NO OPERATION
HLT	HALT
OI	DISABLE INTERRUPT
EI	ENABLE INTERRUPT

Literatura :

[1] Lojík,V.-Sýkora,J.:Číslicová a impulsová technika I. Skriptum ČVUT-FEL, Praha 1981

[2] Lojík,V.: Číslicová a impulsová technika II. Skriptum ČVUT-FEL, Praha 1984

[3] Chod,J.: Číslicová a impulsová technika I. - cvičení Skriptum ČVUT-FEL, Praha 1983

[4] Integrované obvody, tranzistory, diody, triaky, optoelektronické součástky. Katalog fy. Tesla Rožnov, Rožnov 1986

[5] Bernard,M.-Hugon,J.: Od logických obvodů k mikroprocesorům I. SNTL, Praha 1983

[6] Bernard,M.-Hugon,J.: Od logických obvodů k mikroprocesorům II. SNTL, Praha 1983

[7] Bernard,M.-Hugon,J.: Od logických obvodů k mikroprocesorům III. SNTL, Praha 1983

[8] Bernard,M.-Hugon,J.: Od logických obvodů k mikroprocesorům IV. SNTL, Praha 1984

[9] Putz,J. a kolektiv : Úvod do číslicové techniky SNTL, Praha 1983

[10] Katalog elektronických součástek, konstrukčních dílů, bloků a přístrojů 1983 - 1984, 1.díl. Tesla Eltos, Praha 1983

[11] Mikropočítače a mikroprocesory. Příloha ARA 1/82 až ARA 10/82. Naše vojsko, Praha 1982

[12] Čech,M.-Jandoš,J.: Úvod do mikropočítačů. Výběr informací z organizační a výpočetní techniky, Kancelářské stroje, Praha 1983

[13] Stříž,V.: Číslicově - analogové převodníky. Sdělovací technika 6/1982. SNTL, Praha 1982

[14] Jiřička,K.: Analogově - číslicové převodníky pro systémy s mikroprocesory. Sdělovací technika 2/1983. SNTL, Praha 1983

[15] Mirtes,B.-Sýkora,Z.: Číslicové měření a zpracování analogových veličin. SNTL, Praha 1981

[16] Sobotka,Z.: Přehled číslicových systémů. SNTL, Praha 1981

[17] Budinský,J.: Polovodičové paměti a jejich použití. SNTL, Praha 1977

[18] Mikroprocesor 8080 a základní obvody. Skriptum ČVUT-ČSVTS, Praha 1982

[19] Základní instrukce mikroprocesoru 8080. Skriptum ČVUT-ČSVTS, Praha 1982

[20] Dlabola,F.-Starý,J.: Systémy s mikroprocesory a přenos dat. NADAS, Praha 1984

[21] Dolejší,A.: Návod na použitie ŠMS-VÚVT. Manuál mikropočítače ŠMS-VÚVT. Žilina 1983

[22] Dolejší,A.: Školský mikropočítačový systém VÚVT. Datasystém, Bratislava 1982

[23] Burger,J.: Cvičné programovanie na ŠMS-VÚVT. DT ČSVTS Bratislava. Bratislava 1983

[24] Zdeněk,J.: Číslicová a impulsová technika - cvičení.
Skriptum ČVUT-FEL, Praha 1983

[25] Intel : The 8080/8085 Mikroprocesor Book.
John Wiley, New York 1980

[26] Dědina,B.-Valášek,P.: Mikroprocesory a mikropočítače.
SNTL, Praha 1983

[26] Leventhal,L.A.: 8080/8085 Assembly Language Programming.
Osborne/McGraw-Hill, 1978

[28] Sobotka,Z.-Starý,J.: Mikropočítače.
ÚTEPS Tesla-VÚST, Praha 1980

[29] Jeníček,Č.-Nohel,J.-Santus,A.: Programování v asembleru.
ČSVTS, Praha 1982

[30] Chod,J.: Číslicová a impulsová technika II. - cvičení
Skriptum ČVUT-FEL, Praha 1985

[31] Kraus,V.: Instrukční soubor mikroprocesoru 8080.
ČSVTS, ČVUT-FEL, Praha 1983

[32] Mikropočítačové systémy s obvody řady Z-80 (U880D).
Sborník přednášek ČSVTS Brno, ČSVTS, Brno 1984

Obsah :

1. **Mikropočítače** 3
   1.1 Struktura mikropočítače 3
   1.2 Centrální jednotka 4
   1.3 Způsoby adresování 5
   1.4 Programové vybavení mikropočítače 6
   1.5 Jazyk symbolických adres 7
   1.6 Grafické značky pro zápis algoritmů 8
2. **Mikroprocesor 8080A** 10
   2.1 Struktura mikroprocesoru 8080A 11
      2.1.1 Registry 8080A 11
      2.1.2 Aritmeticko logická jednotka (ALU) 13
      2.1.3 Řídicí registr a řízení 14
      2.1.4 Datové oddělovací obvody pro sběrnici (Data Bus Buffer) 14
   2.2 Časování mikroprocesoru 8080A 14
   2.3 Průběhy přerušení 16
   2.4 Průběh HOLD 17
   2.5 Průběh při stavu HALT 17
   2.6 Uvedení mikroprocesoru 8080A do provozu 17
   2.7 Instrukční soubor mikroprocesoru 8080A 17
   2.8 Mikropočítačová stavebnice MCS-80 33
      2.8.1 Skupina mikroprocesoru 33
         2.8.1.1 Generátor hodinového signálu 8224 33
         2.8.1.2 Řídicí obvod sběrnice 8228 (System Controller) 33
      2.8.2 Paměti 36
      2.8.3 Obousměrný zesilovač datové sběrnice a logika řízení systému 36
         2.8.3.1 Obvody 8216 a 8226 36
      2.8.4 Obvody vstup/výstup 37
      2.8.5 Doplňující obvody stavebnice MCS 80 39
         2.8.5.1 Universální registr 3212 (8212) 39
         2.8.5.2 Programovatelný obvod 8255 39
         2.8.5.3 Programovatelný seriový interface 8251 50
         2.8.5.4 Programovatelný časovač 8253 52
   2.9 Zásady pro tvorbu programů a jejich ladění 56
      2.9.1 Testování 56
      2.9.2 Zásobníková paměť 58
      2.9.3 Tabulky 59
      2.9.4 Bitové operace 59
      2.9.5 Pole a použití ukazatelů 62
      2.9.6 Styk mikropočítače s vnějším prostředím 62
3. **Školní mikropočítač ŠMS-VÚVT** 64
   3.1 Popis činnosti ŠMS-VÚVT 68
      3.1.1 Monitor - základní funkce 68
      3.1.2 Ladění programu 68
      3.1.3 Simulace činnosti procesoru pro RST 5 a RST 6 69
      3.1.4 Archivace programu na magnetofonové pásce 69
      3.1.5 Monitor - rozšiřující funkce 70
      3.1.6 Příkazy pomocného monitoru 70
   3.2 Technické prostředky ŠMS-VÚVT 71

3.2.1 Displej....71
3.2.2 Vstupní a výstupní obvody....71
3.2.3 Časovač....73
3.2.4 Převodníky A/D a D/A....74
3.2.5 Výkonové výstupy....75
3.2.6 Přerušovací systém....76
3.2.7 Modem pro magnetofon....78
3.2.8 Modem pro připojení dálnopisu T100. 78
3.2.9 Interfejs RS 232C....78
3.2.10 Stykový konektor....78
3.2.11 Přiřazení konektorů ŠMS-VÚVT....79
4. **Příklady programů**....80
4.1 Přehled podprogramů monitoru ŠMS-VVÚT....80
4.1.1 Podprogramy pro klávesnici a displej....81
4.1.2 Podprogramy pro realizaci časového zpoždění....83
4.1.3 Pomocné podprogramy....83
4.2 Programování obvodů 8255 v ŠMS-VÚVT....84
4.3 Programy využívající přerušení....85
4.4 Programování obvodů 8253 v ŠMS-VÚVT....86
4.5 Programování převodníků A/D a D/A v ŠMS-VÚVT....86
4.6 Programy s elektromotorem....92
4.7 Standardní programy....93
Příloha....98
5. **Instrukční soubor 8080A**....98
5.1 Skupina přesunových instrukcí....98
5.2 Instrukce měnící obsah registru nebo paměti....98
5.3 Skupina aritmetických operací....98
5.4 Skupina logických operací....99
5.5 Skupina rotací a posunů....99
5.6 Operace se zásobníkem a ukazatelem zásobníku....99
5.7 Skupina instrukcí větvící program....100
5.8 Instrukce vstupu a výstupu....100
5.9 Řídicí instrukce....100
Literatura....101