KNIŽNICA ČSVTS
MIKROPROCESOROVÁ TECHNIKA

CVIČNÉ PROGRAMOVANIE NA ŠKOLSKOM MIKROPOČÍTAČOVOM SYSTÉME ŠMS VÚVT

ZVÄZOK 1.

DIEL 5.

1983

Ing. Ivan B u r g e r ml.,CSc., Ing. Pavel Č i č á k

Ing. Andrej K a d l i c

CVIČNÉ PROGRAMOVANIE NA ŠKOLSKOM MIKROPOČÍTAČOVOM SYSTÉME ŠMS VÚVT

Bratislava 1982

ČESKOSLOVENSKÁ VEDECKOTECHNICKÁ SPOLOČNOSŤ - DOM TECHNIKY
BRATISLAVA

Pre účastníkov podujatia a pre služobnú potrebu podnikov.
Publikácia je predajná iba socialistickým organizáciám.
Cena je stanovená na základe smerníc ÚR ČSVTS č. 19/1978,
výmer č. 41/Z/83.

Autori	Ing. Ivan BURGER ml., CSc., Ing. Pavel ČIČÁK, Ing. Andrej KADLIC
Názov	CVIČNÉ PROGRAMOVANIE NA ŠKOLSKOM MIKROPOČÍTAČOVOM SYSTÉME ŠMS VÚVT
Lektor	Ing. Ivan Majer
Zodpovedná pracovníčka	Terézia Klajbanová
Vydavateľ	Slovenská rada ČSVTS Bratislava
Vydanie	Prvé, február 1983
Náklad	2000
Rozsah	192 strán, 9,681 AH, 9,929 VH
Č. výr.	DT/9/83
Č. pov.	SÚKK-747/I-68
Tlač	Dom techniky ČSVTS Bratislava

Úvod

Tento učebný text je určený záujemcom o programovanie na ŠMS VÚVT, ktorí sa oboznamujú s mikropočítačovou stavebnicou typu 8080. Hodí sa preto ako učebný text pri praktickom programovaní a laborovaní v rámci organizovanej výuky pri cvičení v kolektíve i v samostatnom štúdiu, ale aj samoukom, ktotí majú možnosť s ŠMS VÚVT pracovať.

Predpokladom efektívneho použitia učebného textu sú určité znalosti z číslicovej techniky, princípov počítačov, ich základných častí a základov programovania číslicových počítačov. Konkrétne sa žiada poznať množinu inštrukcií mikroprocesora 8080, účinok jednotlivých inštrukcií, jazyk symbolických adries 8080, základné programové konštrukcie, charakteristiku integrovaných obvodov - členov stavebnice 8080 a prirodzene štruktúru a niektoré konkrétne vlastnosti technických a programových prostriedkov Školského mikropočítača VÚVT. Učebný text nemá za cieľ a preto ani neobsahuje výuku programovania, ale uvádza príklady programov a programovanie na ŠMS VÚVT. To znamená, že študujúci čitateľ má poznať základy programovania mikropočítača 8080 vo všeobecnosti alebo aspoň charakter programovania v niektorom strojovo závislom jazyku (asembleri) menších procesorov-minipočítačov.

Autori pri tvorení učebného textu vychádzali zo skúseností pro výuke nových odborníkov v oblasti mikropočítačovej techniky, aj z kurzov na preškoľovanie odborníkov z praxe. Preto je učebný text zameraný najmä na otázky programovania vstupov a výstupov, obsluhy prerušení a aritmetické funkcie s cieľom ukázať možnosti a typické funkcie mikropočítača 8080.

Učebný text neobsahuje podrobný opis jednotlivých stykových obvodov použitých v ŠMS, ani opis inštrukčného súboru. V tomto odkazujeme čitateľa na inú literatúru [1], [3]. Zámerom je objasniť prístup pri tvorení aplikačných programov a to jednak na úrovni systémovej a tiež pri zapísaní konkrétneho programu s danou funkciou, s cieľom dosiahnuť základnú programátorskú rutinu a prehľad čitateľa.

Z didaktického hľadiska sú cvičné programy zostavené tak, aby ich požadovaná funkcia bola pre čitateľa pracujúceho v technickej oblasti zaujímavá a príťažlivá. Aj z tohoto aspektu obsahuje učebný text návrhy na doplnenie a úpravu celkovej zostavy ŠMS VÚVT. Z metodického hľadiska sú cvičné programy zoradené podľa stúpajúcej zložitosti funkcie a náročnosti programovania. Zároveň sú zostavené s ohľadom na typické aplikačné možnosti pri riadení technologických procesov, spracovaní dát, stavbe mnohofunkčných prístrojov s mikropočítačom atď.

Popritom nie je dôraz položený na funkciu programu, ale na zostavenie programu s danou funkciou. Preto by iste bolo možné vytvoriť programy s atraktívnejšími vonkajšími prejavmi, čo sa však v praxi zrejme málokedy hodí. Z podobných dôvodov nie sú cvičné programy najoptimálnejšie z hľadiska rýchlosti aj spotreby pamäti.

Dôraz je položený (okrem cvičných programov objasňujúcich funkciu niektorej zložky vybavenie ŠMS) na modulovosť programov tak, aby pokiaľ možno každá programová jednotka zabezpečovala jednu konkrétnu i keď jednoduchú funkciu. Jedným dôvodom je tvoriť u čitateľa návyk štrukturálneho programovania. Druhým dôvodom je to, aby čitateľ dostal k dispozícii súbor programových modulov, ktorý (okrem svojej inšpiračnej funkcie) je použiteľný pri tvorení programových systémov so zadanou alebo zvolenou funkciou. Tvorivá funkcia čitateľa sa tým posúva do oblasti systémových úvah.

Učebný text zámerne neobsahuje vývojové diagramy programov. Zámerom je prinútiť študujúceho čitateľa "myslieť" v kóde 8080, pretože ako už bolo spomenuté vyššie, nie tvorenie algoritmu je podstatné (na to sú iné literárne zdroje), ale jeho zakódovanie mnemokódom 8080 podľa zásad jazyka assembler 80.

Študujúcemu čitateľovi odporúčame pri štúdiu a praktickej činnosti (laborovaní) pridržať sa týchto pokynov:

- každý uvedený cvičný program alebo jeho podprogramy premyslieť najmä z hľadiska odôvodnenosti použitých inštrukcií a programových konštrukcií a pokúsiť sa nájsť možnosti na zlepšenie programu podľa niektorého kritéria (rýchlosť, náročnosť na pamäťový priestor, prehľadnosť)
- pokúsiť sa modifikovať uvedené programy s cieľom prispôsobiť alebo rozšíriť žiadanú funkciu podľa zadaných alebo zvolených požiadaviek a obmedzení
- zostaviť vlastný program s rovnakou alebo zmenenou funkciou, ale s vlastnou koncepciou a prístupom pri tvorení programu (= nájdenie iných prostriedkov na dosiahnutie danej funkcie). Dôležité je pritom stanoviť systémové väzby medzi programami (a aj okolím - vstupy, výstupy), správna voľba postupnosti inštrukcií, bezchybné preloženie do strojového tvaru, uloženie a najmä overenie funkcie na príkladoch so zdôvodnenými vstupmi (s cieľom odhaliť chyby)
- zostaviť rozsiahlejší programový systém s použitím uvedených programových modulov. Tu je dôležité konštatovať možnosti a obmedzenia v oblasti systémového návrhu

- pripraviť si cvičné periférie (opísané v čl.14) na dosiahnutie vstupu a výstupu informácií v rozličných formách s pozorovateľnými efektami. Je to významné z metodického, ale aj praktického hľadiska
- pri kolektívnej výuke alebo v osvetovej práci je vhodné cvičné programy (aj iné, samostatne vytvorené) uložiť na magnetofónovú pásku kazetofónu ŠMS so slovným komentárom a návodom na použitie
- doplniť si prostriedky výuky odporúčanou literatúrou

Autori dúfajú, že tento učebný text bude dobrým podporným prostriedkom výuky na ŠMS VÚVT a že sa využijú bohaté možnosti tohto školského mikropočítača.

Prehľad uvedených cvičných programov, podprogramov a ich funkcií

CP1 : (v čl.3) ukážka funkcie monitorových podprogramov SCAN a DBYTE

CP2 : (v čl.3) overenie funkcie monitorového podprogramu DWORD

CP3 : (v čl.3) overenie funkcie monitorového podprogramu ENTWD

CP4 : (v čl.3) ukážka kódovania segmentov LED displeja v obraze zobrazovaného znaku

CP5 : (v čl.3) zobrazenie písmena "U" na displeji podľa kódovania jeho segmentov

CP6 : (v čl.3) ukážka funkcie a použitia monitorového podprogramu ERROR

DELYH: (v čl.3) podprogram na programové zdržanie 1 ms až 65 sekúnd

CP7 : (v čl.4) nastavenie riadiaceho registra obvodu 8255 podľa zadaného režimu práce

CP8: (v čl.4) nastavenie zadaného bitu portu do log.1

CP9: (v čl.4) nulovanie zadaného bitu portu do log. 0

CP10: (v čl.4) zmena zadaného bitu portu do opačnej hodnoty

CP11: (v čl.5) príklad obsluhy prerušenia s meraním času stlačenia klávesu

CP12: (v čl.6) nastavenie počítadla T0 na generovanie kmitov s periódou 2 ms

CP13: (v čl.9) prenos kódu z podprogramu GETKY na LED - diódy portu P1A

CP14 : (v čl.9) prenos kódu z podprogramu SCAN na LED - diódy portu P1A

CP15 : (v čl.9) rozsvietenie daného počtu LED - diód portu P1A podľa kódu z klávesnice

CP16 : (v čl.9) cyklický posuv svietiaceho bodu na LED diódach portu P1A

CP17 : (v čl.9) kmitavý posuv svietiaceho bodu na LED diódach portu P1A

CP18 : (v čl.9) zmena hodnoty bitu portu P1A zadaného z klávesnice

CP19 : (v čl.10) generovanie tónu 500 Hz programom cez port P1A

CP20 : (v čl.10) plynulá cyklická zmena tónu generovaného programovo cez port P1A

CP21 : (v čl.10) generovanie žiadaného tvaru signálu (obdĺžnik, píla, trojuholník, sínus) pomocou D/A prevodníka

CP22 : (v čl.10) generovanie tónu 1 kHz počítadlom TØ

CP23 : (v čl.10) klávesový hudobný nástroj

CP24 : (v čl.11) ovládanie chodu motora z klávesnice cez D/A prevodník

CP25 : (v čl.11) impulzné ovládanie chodu motora s programovým riadením

CP26 : (v čl.11) impulzné ovládanie chodu motora s využitím časovačov a prerušení

CP27 : (v čl.11) ovládanie chodu motora cez D/A prevodník a meranie otáčok zisťovaním frekvencie zo snímača otáčok významné podprogramy:

- DA : na riadenie rýchlosti motora

- MER : na programové meranie frekvencie

- PRER : ukončenie daného časového intervalu

- ZDRZ : programové zdržanie 256 ms

CP28 : (v čl. 11) regulácia otáčok motora s meraním otáčok a spätnovázbovou slučkou

významné podprogramy:

- MER : zistenie otáčok motora meraním doby jedného segmentu snímača otáčok

- REGU : podprogram na reguláciu otáčok porovnaním žiadanej a skutočnej rýchlosti motora reléovým riadením

- STRT : začiatok merania časového intervalu 32 ms

- PRER : ukončenie merania časového intervalu 32 ms

- ZOBR : pomalé zobrazovanie údaja so spomalením frekvencie meraní v pomere 1 : 256

CP29 : (v čl. 12) číslicovo-analógový prevod kódu zadaného z klávesnice

CP30 : (v čl. 12) číslicový voltmeter s hexadecimálnym zobrazením

významné podprogramy:

- AD1 : programový A/D prevod s inkrementáciou

- AD2 : programový A/D prevod s postupnou aproximáciou

CP31 : (v čl. 12) automatický číslicový voltmeter

významný podprogram:

- PRER : obsluhuje prerušenie od A/D prevodníka

CP32 : (v čl. 12) ukážka obsluhy prerušení z dvoch zdrojov prerušení

významný podprogram:

- PRER : rozoznáva aktívny zdroj prerušenia a obsluhuje ho

CP33 : (v čl. 12) rýchly analógový vstup s číslicovým uložením nameraných hodnôt

významný podprogram:

- PRER: obsluha prerušení vo vzorkovacích okamihoch

CP34 : (v čl. 12) analógový výstup číslicovo uložených dát

významný podprogram:

- DELYH : programové zdržanie 1 ms až 1 min

CP35 : (v čl. 12) programové generovanie lineárne vzrastajúcej analógovej funkcie

CP36 : (v čl. 13) generátor reálneho času

významný podprogram:

- PRER : aktualizácia časového údaja podľa prerušení z časového počítadla

CP37 : (v čl. 13) generátor reálneho času s prednastavením

CP38 : (v čl. 13) podprogram na vyhľadanie maxima v súbore čísel

CP39 : (v čl. 13) kontrolný program k podprogramu CP38

CP40 : (v čl. 13) cvičný program na overenie funkcie aritmetických podprogramov so 16 bitovými operandami

významné podprogramy:

- SU16 : 16 bitové binárne sčítanie so znamienkom a ošetrením preplnenia

- PO16 : porovnanie dvoch 16 bitových binárnych čísel

- ZNAM : zmena znamienka 16 bitového binárneho čísla

CP41 : (v čl. 13) cvičný program na násobenie dvojkových čísel so znamienkom, 16 bitov

významné podprogramy:

- NA16 : násobenie dvoch čísel so znamienkom s posúvaním medzisúčtov, 16 bitov

- POSB : posúvanie dvojice BC doprava

CP42 : (v čl. 13) cvičný kalkulačný program s porovnávaním, sčítavaním, odčítavaním a násobením desiatkových čísel bez znamienka, 16 bitov

významné podprogramy:

- SU10 : súčet dvoch desiatkových čísel bez znamienka, 16 bitov

- PR10 : nastavenie príznakov z desiatkového čísla bez znamienka, 16 bitov

- RO10 : rozdiel dvoch desiatkových čísel bez znamienka, 16 bitov

- DO10 : vytvorenie desiatkového doplnku k desiatkovému číslu bez znamienka, 16 bitov

- PO10 : porovnanie dvoch desiatkových čísel bez znamienka, 16 bitov

- NA10 : násobenie dvoch desiatkových čísel bez znamienka, 16 bitov

CP43 : (v čl. 13) prevod zadaného hexadecimálneho čís-

la na dekadické a zobrazenie, 16 bitov

významný podprogram:

- PREV : prevod hexadecimálneho 16 bitového čísla na dekadický tvar, 16 bitov

CP44 : (v čl. 13) prevod jedného ASCII kódu na hexadecimálnu číslicu a späť

významné podprogramy:

- TEST : testovanie, či kód znaku predstavuje hexadecimálnu číslicu
- CSL : prevod kódu znaku na hexadecimálnu číslicu
- ZNAK : prevod hexadecimálnej číslice na kódové slovo ASCII kódu

1. Funkcie riadiaceho programu (monitora) a ich používanie

Školský mikropočítačový systém ŠMS VÚVT je určený na výuku základov mikropočítačovej techniky a základných aplikácií. Riadiaci program (ďalej monitor) má za úlohu poskytnúť funkcie, ktoré sú typické a nevyhnutné pri laborovaní na ŠMS a pri zostavení a odladovaní používateľských programov. V [2] sú uvedené podrobne všetky jeho funkcie najmä z hľadiska jeho konkrétnej realizácie a významu jednotlivých riadiacich klávesov. Pre naše potreby sa sústredíme na opis z opačného hľadiska: uvedieme, ktoré funkcie môže programátor využívať, v ktorých situáciách a akou manipuláciou na klaviatúre. Zameriame sa na typické úkony pri tvorení programov.

Základná zostava mikropočítača má celkový pamäťový priestor (adresy 0000 ÷ FFFF) fyzicky pokrytý dvomi pamäťami. V pevnej pamäti je na adresách 0000 ÷ 03FF uložený monitor, ktorého celkový rozsah je 1 KB. Druhá pamäť je typu RWM (read-write memory) a je určená na uloženie používateľského programu a jeho dát. Keďže aj riadiaci program potrebuje používať svoje premenné a mať ich v pamäti (pevná pamäť neumožňuje zápis dát), časť RWM pamäti je vyhradená pre potreby monitora a sú tam uložení jeho vnútorné tabuľky. Rozsah RWM pamäti je tiež 1 KB a to na adresách 8000 ÷ 83FF. Z toho pre monitor je určená oblasť 83A0 ÷ 83FF a pre programy používateľa oblasť 8000 ÷ 839F. Preto cvičné programy môžu byť len v tejto oblasti. Z toho vyplýva maximálna veľkosť používateľských programov (včítane dát), ktorá vychádza 928 byteov. V prípade núdze možno v určitom rozsahu zasiahnuť aj do oblasti monitora a používať ju. Hrozí, pravdaže, premazanie tabuliek monitora a tým následné zlyhanie jeho činnosti.

Takýto pokus je však viazaný na podrobné preskúmanie dôležitosti premazávaných dát v uvedenej oblasti. Inak je potrebné pamäť RWM rozšíriť prídavnými pamäťovými modulmi (podrobnejšie pozri [1]). Treba však poznamenať, že základný rozsah používateľskej RWM pamäti je pre cvičné programovanie úplne dostačujúci.

Pri opise používaných funkcií monitora predpokladajme, že máme k dispozícii strojový tvar programu, t.j. obsahy jednotlivých buniek pamäti v hexadecimálnom tvare. Programátor zapisuje program v assembleri (mnemokódy inštrukcií) a z neho možno hexadecimálny tvar získať preložením prekladačom ASSEMBLER na počítači alebo ručným preložením, čo nie je pri krátkom cvičnom programe problém (pozor na mechanické chyby). V každom prípade máme k dispozícii symbolický tvar programu (mnemokódy) a strojový tvar (obsah jednotlivých buniek).

V prvej etape po preložení treba program uložiť do pamäti na určené adresy. Využijeme pritom funkciu ukladania dát do pamäti. Najprv treba oznámiť adresu začiatku ukladania (červený kláves "A", po ňom štvormiestna hexa-adresa) a potom povolíme zmenu obsahu pamäti (stlačiť červené "M"). Tento režim je indikovaný bodkou v šiestej pozícii displeja. Adresovaná bunka pamäti (jej adresu ukazuje ľavá časť displeja) sa naplní hodnotou, ktorá sa zadá z klávesnice. Zvýšenie ukazovateľa adresy (aby sme mohli naplniť ďalšiu bunku) dostaneme klávesom "N". Takto treba pokračovať pri zadaní celého programu, od najnižších adries k najvyšším. Je možné ukladať program aj od vyšších adries k nižším (hoci je to nezvyklé) a dosiahneme to stláčaním klávesu "M" namiesto "N".

Funkcia ukladania dát (programu) je uľahčená týmito funkciami monitora:

- pri napĺňaní určitej bunky možno stlačiť viac údajových klávesov, ale významné sú len posledné dva (chybu možno priebežne opraviť)
- po zadaní nového obsahu určitej bunky možno obnoviť pôvodný obsah stlačením klávesu "C", ak medzitým nebol stlačený iný riadiaci kláves (ak bola chybne zadaná adresa bunky)
- monitor kontroluje, či sa zadané údaje skutočne zapísali do pamäti a v prípade potreby rozsvieti text "Err" (ak používateľ zadal adresu neexistujúcej pamäti resp. pri nesprávnej funkcii existujúcej pamäti)

Po uložení programu do pamäti treba skontrolovať správnosť uložených dát (chybne zadané dáta). Využijeme prehliadanie obsahu pamäti tak, že nastavíme ukazovateľ adries (kláves "A" nasledovaný štvormiestnou hexa-adresou) a postupne zdola nahor kontrolujeme program použitím klávesu "N". Použitím klávesu "M" možno program kontrolovať zhora nadol.

V ďalšej fáze je potrebné overiť správnu funkciu programu, resp. dosiahnuť jeho správnu funkciu odstránením chýb t.j. ladením. Na to poskytuje monitor niekoľko svojich funkcií.

Pri ladení je možné postupovať programom po jednotlivých inštrukciách pri súčasnom pozorovaní obsahu pamäti alebo registrov. Funkciu dosiahneme zadaním štartovacej adresy (kláves "A" nasledovaný štvormiestnou hexa-adresou) a postupným stláčaním klávesu "S". Každým stlačením sa vykoná ďalšia inštrukcia programu. Pritom musí byť prepínač STEP-AUTO v polohe STEP. Na displeji sa zobrazuje adresa nasledujúcej inštrukcie a jej operačný kód. Po každej inštrukcii je možné prezrieť si obsah pamäti a registrov (A, B, C, D, E, F, H, L, SP, vrch zásobníka). Pri la-

dení programátor porovnáva skutočný obsah pamäti alebo registrov s predpokladaným obsahom. Okrem toho v tomto režime možno priebežne pozorovať hodnotu príznakov "C" a "Z" procesora, ktoré sú vysvietené LED-diódami pri klávesnici. Posledne vybraný register sa neustále po každej inštrukcii zobrazuje. Posledne vybranú bunku pamäti možno po každej inštrukcii zobraziť stlačením klávesu "M". Toto sú pomerne bohaté a dostačujúce prostriedky ladenia v režime "po inštrukciách". Režim STEP sa dosahuje využitím prerušenia procesora, ktoré využíva monitor pri zabezpečovaní tejto funkcie. Z toho vyplýva obmedzenie: režim STEP nie je možný, ak procesor je v stave DI (disable interrupt). Ten nastane po inštrukcii DI alebo po prerušení od niektorej periférie. Stav EI (enable interrupt) dosiahneme inštrukciou EI. Preto ju treba dať do obslužnej rutiny prerušenia, anonáhle je to možné (pozri časť 5), ak chceme aj v nej využívať režim STEP. V režime STEP možno aj meniť obsah jednotlivých registrov resp. pamäti tak, ako si to želá programátor pri ladení. Možno aj zmeniť sled vykonávania inštrukcií tak, že sa zadá v niektorom bode programu nová štartovacia adresa a odtiaľ sa pokračuje v ladení. Ak je to potrebné, možno v režime STEP prejsť do režimu riadneho vykonávania programu, a to od bodu, kde v programe práve sme.

Zobraziť obsah niektorého registra možno stlačením klávesu "R" a príslušného mena registra (A, B, C, D, E, F). Použitím klávesu "N" sa zobrazí ďalší register (po F ide H, potom L). Šestnásťbitové registre sa zobrazujú takto:

BC	"A", "B", "M"
DE	"A", "D", "M"
HL	"A", "8", "M"
SP	"A", "1", "M"

vrch zásobníka "A", "2", "M"

Zároveň sa zobrazuje bunka pamäti na tej adrese, na ktorú ukazuje príslušný šestnásťbitový register. Túto bunku možno tiež zmeniť ako v procese zadávania dát do pamäti.

Režim riadneho vykonávania programu možno dosiahnuť zadaním štartovacej adresy podobne ako v režime STEP. Rozdiel je v tom, že potom treba stlačiť kláves "G". Ak je prepínač STEP/AUTO v polohe AUTO, procesor neprerušovane vykonáva používateľský program, t.j. beží plnou rýchlosťou. Program používateľa sa ukončí pri skoku do monitora na adresu 0020_H (inštrukciou RST 4). Tento bod sa nazýva "teplý štart monitora". Pri ňom sa uchovajú obsahy registrov programu používateľa, takže ich možno po skončení programu prezrieť. Takisto sa uchovajú niektoré iné informácie (body prerušenia, ukazovateľ pamäti, ukazovateľ programu používateľa atď.). Pri skoku na adresu 0000_H sa vykoná rovnaká činnosť monitora, ako po zapnutí mikropočítača. Toto je tzv. "studený štart monitora" a neuchovávajú sa uvedené informácie. Na adresu 0000_H sa skočí aj pri stlačení klávesu "RESET", čo možno využiť na násilné ukončenie napr. chybne bežiaceho programu.

Ladenie programu spôsobom "po inštrukciách" má tú nevýhodu, že nedôležité časti programu treba pracne "preskákať" pomocou klávesu "S". Preto monitor umožňuje zadať body dočasného prerušenia programu (ďalej len zarážka) k niektorým inštrukciám. Ak v režime "G" príde program na takúto inštrukciu, ihneď sa prejde na teplý štart monitora. Potom môže programátor vykonať podobné činnosti ako v režime "S" (prezeranie, zmena obsahu registrov a pamäti). Ďalej je možné v programe pokračovať stlačením klávesu "S" alebo "G" podľa požadovaného režimu (pokračuje sa od toho bodu, kde bola zarážka). Pritom je možné zadať, pri koľkom prechode danou zarážkou sa má pro-

gram zastaviť. Používanie zarážok sa riadi týmito zásadami:

- zarážka sa zadáva touto postupnosťou: "A" , štvormiestna hexa-adresa inštrukcie na ktorej je zarážka, "B" , dvojmiestne hexa-číslo = počet povolených opakovaných prechodov danou zarážkou
- po každom prechode danou zarážkou sa počet povolených opakovaní zmenší o jeden (ak už nie je nula)
- ak je okamžitý povolený počet prechodov nula, prejde sa na teplý štart monitora (tým sa zarážka neruší)
- možno zadať maximálne osem zarážok, každá môže mať maximálne 255 povolených opakovaných prechodov
- zarážku možno zobraziť (adresu a aktuálny počet povolených prechodov) stlačením klávesu "B" a ďalšie zarážky cyklicky prezerať klávesom "N" (zobrazovať možno len keď nebeží používateľský program)
- zarážku možno zrušiť tak, že sa zobrazí a stlačí sa kláves "C"
- stlačením klávesu "RESET" resp. skokom na adresu 0000_H sa rušia všetky zarážky
- funkcia zarážky sa dosahuje pomocou prerušení podobne ako funkcia "S". To znamená, že procesor nesmie byť v stave DI (pozri režim "S")
- funkcia zarážky sa realizuje tak, že po každej používateľskej inštrukcii sa preruší program a kontroluje sa, či nasledujúca inštrukcia nie je označená zarážkou. Preto prepínač STEP/AUTO musí byť v polohe STEP. Vyplýva z toho, že v režime STEP sa pri funkcii "G" program realizuje podstatne pomalšie, ako keď je prepínač v polohe AUTO.

- z dôvodu vlastností konkrétneho technického riešenia obvodov mikropočítača sa môže stať, že hoci procesor je v stave EI, zarážka je neúčinná. Napríklad ak program má inštrukcie: DI ... EI, K, K, K, K, D ... , tak bude účinná len taká zarážka, ktorá je umiestnená až za prvou dlhou inštrukciou /D/ a neúčinná je taká zarážka, ktorá ukazuje na niektorú krátku inštrukciu /K/ ešte pred prvou dlhou inštrukciou (krátka inštrukcia je jednobyteová, ostatné sú dlhé).

Funkciou monitora je aj kontrolovať správnosť operovania na klávesnici. Pri nesprávnej postupnosti zásahov sa na displeji objaví text "Err".

Okrem uvedeného umožňuje monitor simulovať činnosť procesora po inštrukciách resp. prerušení RST 5 a RST 6. Problém je v tom, že reálny mikropočítač s procesorom 8080 po RST 5 a RST 6 prejde na inštrukciu na adrese 0028_H resp. 0030_H. V školskom mikropočítači však používateľ na tieto adresy program uložiť nemôže, pretože je to oblasť pevnej EPROM pamäti. Preto monitor zariadi, že sa prejde na iné adresy, nie na 0028_H a 0030_H. Pri RST 5 sa prejde na adresu, ktorá je uložená na adrese 83EC, 83ED (dva byty). Po studenom štarte tam monitor ukladá hodnotu 8228_H. Pre RST 6 sa použije adresa 83EA, 83EB a po studenom štarte sa tam dosadí hodnota 8230_H. Teda obslužné rutiny prerušenia RST 5 a RST 6 sa uložia na určené a dohodnuté adresy a nie na adresy 0028_H a 0030_H.

Ďalšou významnou funkciou monitora je možnosť uložiť obsah pamäti na kazetovú pásku cez magnetofón (archivácia programov). Určená oblasť pamäti sa postupne po bitoch prenesie do magnetofónu tak, že dáta sa modulujú na akustický kmitočet. Mimo prenosu údajov sa do magnetofónu trvale vedie nemodulovaný tón. Pred záznamom na pásku

treba pomocou mikrofónu nahrať slovný komentár (funkcia programu, dátum, adresa štartovacia, ukladacia, koncová atď.). Potom sa pripraví monitor na prenos touto postupnosťou klávesov: "A", adresa začiatku prenosu, "M", "A", adresa konca prenosu, "B", "A", "0371". V tomto prípade majú uvedené klávesy zrejme iný význam ako sme uviedli predtým. Potom prepojíme šnúrou mikropočítač s magnetofónom a zapneme nahrávanie na magnetofóne. Tým sa začne nahrávať nemodulovaný tón a asi po dvoch sekundách treba stlačiť "G". Po prenose, ktorý trvá určitý čas, treba vypnúť magnetofón (koniec prenosu sa indikuje rozsvietením displeja). Treba poznamenať, že posledný uvedený byte sa už neprenáša.

Pri spätnom čítaní dát z pásky sa najprv ručne nastaví páska na požadovaný záznam a potom sa nastaví monitor na prenos postupnosťou: "A", adresa začiatku prenosu, "M", "A", "03AE". Spustí sa prehrávanie magnetofónu s dostatočnou intenzitou a po slovnom komentáre pri zaznení nemenného tónu sa stlačí "G". Dáta sa postupne prenesú od zadanej adresy vyššie. Na páske je na konci záznamu nahratý kontrolný údaj, ktorý sa pri čítaní z pásky kontroluje. Ak po prenose z pásky nie je v registri A nulový obsah, prenos sa vykonal chybne (zistíme stlačením "R", "A"). Vzhľadom na nízku cenu tohto záznamového zariadenia a z toho vyplývajúcu spoľahlivosť odporúčame na pásku zapísať viac exemplárov záznamov tých istých dát.

Pri prenose dát ktorýmkoľvek smerom musí byť prepínač STEP/AUTO v polohe AUTO.

Niektoré ďalšie informácie o vlastnostiach monitora možno nájsť v [2].

2. Ladenie programov na školskom mikropočítači

Prax ukazuje, že proces ladenia programov je nevyhnutný a odpadá len pri celkom jednoduchých programoch. Preto treba venovať jeho metodike a prostriedkom náležitú pozornosť.

Princíp hľadania chýb v strojovom programe v mikropočítači spočíva v porovnávaní plánovanej fumkcie procesora a jeho skutočnej činnosti, ktorá môže byť v dôsledku chybného programu nesprávna. Činnosť procesora sa zisťuje po vykonaní inštrukcie prezeraním obsahu dôležitých buniek pamäti, registrov procesora a príznakov procesora, prípadne sledom vykonávania programu v inštrukciách vetvenia. Ladenie programu je tvorivý proces, náročný pri stanovení postupu ladenia a presahuje rámec tejto publikácie. Uveďme však všeobecné konštatovania, že pri ladení treba prejsť všetky vetvy programu, použiť pritom také vstupné dáta programu, ktoré ľahko ukážu jeho nesprávnu činnosť a pokiaľ možno indikujú miesto chyby a že pracnosť odladovania závisí aj na spôsobe zostavenia programu (má byť zostavený štruktúrovane). Je vhodné, ak jednotlivé časti programu možno testovať zvlášť (moduly).

Rekapitulujme prostriedky ladenia školského mikropočítača:

- vykonávanie programu po inštrukciách (je to účinný prostriedok, pri väčších programoch pracný a ťažko sa uplatňuje v programoch s prerušeniami)
- použitie zarážok v dôležitých bodoch programu (vhodné pri testovaní cyklov a väčších programov, má to isté obmedzenie ako režim STEP - prerušenia)
- použitie trvalej zarážky RST 4 (nevýhodná je nutnosť zapísať dočasne pri ladení namiesto niektorej inštruk-

cie inštrukciu RST 4, výhodná je nezávislosť tejto zarážky na stave EI alebo DI)

- všetky uvedené prostriedky umožňujú zisťovať a meniť obsah registrov a buniek pamäti, pri ladení po inštrukciách dokonca možno pozorovať dva vybrané príznaky "C" a "Z"

Pri ladení rozsiahlejších programov je vhodné pred spustením programu urobiť si jeho kópiu na magnetofón, pretože po spustení sa môže eventuálna chyba prejaviť premazaním programu a jeho opätovné ručné zadávanie je pracné. Občas sa stáva, že dodatočne zistená chyba vyžaduje doplniť inštrukcie do programu, čo vyžaduje jeho nový preklad (posunutie zvyšnej časti programu na vyššie adresy). Pri ručnom prekladaní je to pracné a odporúčame preto urobiť odskok z chybnej časti programu do oblasti nepoužitej pamäti. Tam zaradíme chýbajúce inštrukcie, tie inštrukcie ktoré sa prekryli odskokovou inštrukciou a návratovú inštrukciu. Zbytočné časti inštrukcií treba nahradiť inštrukciami NOP. Inštrukcie NOP je vhodné dať preventívne na začiatok programu, prípadne na iné kritické miesta a podľa potreby ich neskôr nahradiť potrebnými inštrukciami. Po celkovom odladení sa program upraví a preloží posledný krát (odskoky sa zaradia normálne do programu a NOP-y sa zrušia).

3. Používanie podprogramov monitora

Riadiaci program monitor potrebuje na zabezpečenie svojich plánovaných úloh vykonávať štandardné úkony, ako napr. vstup z klávesnice, výstup na displej atď. Monitor je zostavený štrukturálne a na uvedené štandardné funkcie používa vlastné rutiny. Rutiny sú zapísané formou podprogramov (končia sa inštrukciou RET) a preto sú prístupné aj používateľovi - môže ich vo svojom programe volať inštrukciou CALL. Sortiment a vlastnosti jednotlivých podprogramov sú prirodzene dané potrebami monitora, ale často ich využijeme aj v cvičnom programovaní.

Súbor podprogramov možno rozdeliť do štyroch skupín:

- podprogramy na ovládanie vstupu z klávesnice
- podprogramy na ovládanie výstupu na displej
- podprogramy na časové zdržanie (oneskorenia)
- pomocné podprogramy

Každý podprogram je charakterizovaný svojou štartovacou adresou, funkciou, vstupmi (do podprogramu), výstupmi (registre, pamäť, príznaky), obmedzeniami použitia, premazávaním obsahu pracovných registrov a volaním ďalších podprogramov. Úplný zoznam podprogramov monitora s podrobnou špecifikáciou funkcie je uvedený v 2 . Upozorňujeme pritom na drobné tlačové chyby v citovanej publikácii, ktoré môžu vyvolať ich nesprávne použitie pri cvičnom programovaní.

Ďalej uvedieme najdôležitejšie podprogramy monitora a sústredíme sa pritom najmä na ich používanie z hľadiska potrieb programátora. Výstavba podprogramov je charakterizovaná tým, že v každej skupine existuje základný univerzálny podprogram a od neho sú odvodené ďalšie. K tomuto princípu uvedieme príklady s cieľom inšpirovať čitateľa v jeho ďalšej práci.

Podprogramy na vstup z klávesnice sú štyri: SCAN, GETKY, ENTBY, ENTWD. SCAN prehľadá všetky klávesnice a zistí, ktorý bol stlačený. Ak bol niektorý kláves stlačený, jeho kód sa uloží do akumulátora. Klávesy 0 ÷ F sú kódované kódmi 00 ÷ 0F, červené riadiace klávesy sú kódované kódmi 10_H ÷ 17_H. Ak nebol stlačený žiaden kláves, nastaví sa príznak CY = 0. Podprogram je zrejme vhodný vtedy, ak hlavný program má pokračovať ďalej keď nie je

stlačený kláves. Vstup klávesu RESET prirodzene nie je možné zariadiť nijakým podprogramom, ani týmto (má účinok ako pri zapnutí mikropočítača = studený štart).

Druhý podprogram GETKY má podobnú funkciu ako SCAN s tým rozdielom, že sa čaká na stlačenie niektorého klávesu. Podprogram v cykle čaká na stlačenie niektorého klávesu a po stlačení a pustení klávesu sa jeho hodnota podobne ako v predošlom podprograme uloží do akumulátora. Príznak CY = 0 v tomto prípade značí, že bol stlačený riadiaci kláves a nie hexadecimálny. Podprogram GETKY je vhodný vtedy, ak má program pokračovať až po stlačení niektorého klávesu. Podprogram GETKY využíva služby podprogramu SCAN:

```
023D    GETKY:  CALL    SCAN    ;volaj SCAN
0240            JNC     GETKY   ;bol stlačený kláves?
                .
                .
        NAVRAT: RET             ;návrat z podprogramu
```

Podprogramy SCAN a GETKY sú určené na vstup niektorého klávesu a teda možno nimi zadať hodnoty 00 ÷ 17_H . . A-k treba zaistiť vstup binárnej hodnoty (a nie klávesu), treba použiť podprogramy ENTBY alebo ENTWD. Podprogram ENTBY je určený na vstup jedného byte z klávesnice. Je možné stlačiť viacero hexadecimálnych klávesov, za platné sa však považujú len posledné dve. Vstup sa ukončí stlačením niektorého riadiaceho klávesu. Zadaný byte je uložený v registri L. Navyše sa kód riadiaceho klávesu ukladá do akumulátora a v registri H je byte zodpovedajúci stlačeným dvom klávesom, ktoré predchádzali pred poslednými dvomi klávesami (ak stlačíme klávesy

1 2 3 4 , v H bude 12_H a v L bude 34_H). Stláčané klávesy sú priebežne zobrazované na pravých dvoch pozíciách displeja, teda okrem vstupu jedného byte podprogram zabezpečuje aj kontrolné zobrazovanie. Príznak Z = 1 značí, že nebol zadaný žiaden hexadecimálny kláves (bol zadaný len riadiaci kláves). Podprogram ENTBY možno vlastne použiť aj na vstup dvoch byteov, na displeji sa však zobrazuje len jeden. Podprogram použijeme vtedy, ak treba zadať hodnotu v rozsahu jedného byte.

Podprogram ENTWD je určený na vstup jedného slova (dva byte). Funkčne je takmer zhodný s podprogramom ENTBY a líši sa od neho spôsobom zobrazovania. Po stlačení štyroch klávesov sa v pravej časti displeja nezobrazí zadaná hodnota, ale obsah pamäťovej bunky, ktorá je adresovaná zadanými štyrmi hexadecimálnymi klávesami. Samotná zadaná hodnota dvoch byteov sa priebežne zobrazuje v ľavej časti displeja. Inak ENTWD pracuje rovnako ako predošlý podprogram (výstup v HL, v akumulátore kód riadiaceho klávesu). Aj v tomto prípade možno zadať viac klávesov, platné sú však len posledné štyri ukončené riadiacim klávesom. Obidva podprogramy ENTBY aj ENTWD nechávajú v registri D počet stlačených hexadecimálnych klávesov.

Najjednoduchším zobrazovacím podprogramom je DISPR, ktorý je určený na zobrazenie jednej hexadecimálnej číslice. Kód číslice má byť uložený v nižších štyroch bitoch akumulátora. V dvojici DE má byť uložená žiadaná pozícia displeja ($83FF_H$ kóduje pravý krajný displej, $83F8_H$ kóduje ľavý krajný displej). Obsah DE sa v podprograme DISPR dekrementuje o jednotku, takže opakované použitie DISPR zapíše na ďalšie ľavé pozície displeja.

Na zobrazenie jedného bytu je určený podprogram DBY2. Zobrazuje sa byte uložený v akumulátore na pozície

displeja, ktoré určuje dvojica DE podobne ako v DISPR (DE určuje umiestnenie nižšej hexadecimálnej číslice). Od DBY2 je odvodený podprogram DBYTE, ktorý zobrazuje obsah akumulátora na pravých dvoch pozíciách displeja a podprogram DMEM, ktorý na posledných dvoch pozíciách displeja zobrazuje obsah pamäťovej bunky, ktorá je adresovaný dvojicou HL. Funkcia DBYTE a DMEM je zrejmá z nasledujúcej ukážky časti programu:

```
0294   DMEM:   MOV    A,M       ;daj (HL) do A
0295   DBYTE:  LXI    D,83FFH   ;pravá pozícia displeja
0298   DBY2:   PUSH   H         ;začína telo DBY2
                .
                .
                RET              ;návrat
```

Obsah dvojice DE sa dekrementuje o 2, takže následné použitie podprogramu DBY2 zapisuje na ďalšie ľavé pozície displeja.

Na zobrazenie dvojbyteovej informácie možno použiť podprogram DWD2, ktorý zobrazuje obsah dvojice HL na špecifikovaných pozíciách displeja analogicky ako v DBY2. Od tohto základného podprogramu sú odvodené ďalšie:

```
02CB   DYPC:   LHLD   83E6H     ;naplň HL z pamäti 83E6
02CE   DMWD:   CALL   DMEM      ;volanie pomocného podpr.
02D1   DWORD:  LXI    D,83FBH   :nastav displej
02D4   DWD2:   MOV    A,L       ;začína program DWD2
02D5           CALL   DBY2      ;zobraz nižší byte
                .
                .
```



```
                RET              ;návrat
```

Je zrejmé, že DWORD zobrazuje obsah HL v ľavých štyroch pozíciách displeja, DMWD navyše zobrazí aj obsah pamäťovej bunky ktorá je adresovaná dvojicou HL a DYPC zobrazí dva byte z adresy $83E6_H$ a tiež obsah bunky týmito dvomi byte adresovanej. Obsah dvojice DE sa analogicky ako v predošlých prípadoch dekrementuje o 4.

Technické prostriedky zobrazovania na displej sú založené na využití DMA kanála z pamäti mikropočítača. Displej pracuje tak, že jeho obvody pravidelne vyberajú obsah buniek $83F8_H \div 83FF_H$ a podľa obsahu jednotlivých bitov sa rozsvecujú jednotlivé LED segmenty každej pozície displeja. Úlohou zobrazovacích podprogramov je podľa dát používateľa naplniť príslušné bunky pamäti správnou kombináciou bitov (urobiť prevod binárna hodnota - obraz hexadecimálnej číslice). Teda displej sa nespráva ako prídavné zariadenie, ale využíva osem buniek pamäti. Rozsvecovanie jednotlivých pozícií displeja musí byť také časté, aby to ľudské oko nespozorovalo. Tým vlastne žiadame, aby sa DMA kanál displeja neustále aktivoval. Pritom počas aktivácie DMA kanála nastáva dočasné zmrazenie činnosti procesora a tým sa spomalí vykonávanie programu. Ak žiadame, aby program bežal plnou rýchlosťou, treba jednak dať prepínač STEP/AUTO do polohy AUTO a tiež zakázať aktiváciu DMA kanála displeja. Privedením byte 00_H na bránu 0C sa zároveň zakážu prerušenia monitora a aktivácia DMA kanála displeja. Treba však poznamenať, že zníženie rýchlosti procesora v dôsledku činnosti displeja je len asi 1 % (frekvencia aktivácie kanála je 2000Hz). Na obnovenie uvedených zakázaných činností možno použiť podprogram MENAB, ktorý patrí medzi pomocné podprogramy.

Uveďme príklad programu na overenie funkcie podpro-

gramov SCAN a DBYTE. Nech sa zobrazuje kód práve stlačeného klávesu.

```
8200  CP1:   CALL   SCAN     ;cvičný program č.1
8203         CALL   DBYTE    ;zobrazenie kódu klávesu
8206         JMP    CP1      ;návrat na začiatok
```

Program neustále cyklí a kód stlačeného klávesu sa priebežne zobrazuje. Podprogramy si vymieňajú informácie cez akumulátor a ostatné registre nie sú dôležité. Ak v uvedenom programe nahradíme volanie SCAN-u volaním podprogramu GETKY, program postúpi do volania DBYTE až po stlačení a uvoľnení klávesu.

Vytvorme program na zobrazovanie kódu stlačeného klávesu a zároveň na zobrazenie príznakov po opustení vstupného podprogramu:

```
8200 CP2:   CALL SCAN    ;cvičný program č.2
8203        PUSH PSW     ;ulož akumulátor a príznaky
8204        POP  H       ;naplň HL odloženými dátami
8205        CALL DWORD   ;zobrazenie HL
8208        JMP  CP2     ;návrat na začiatok
```

V ľavej časti displeja sa zobrazuje obsah akumulátora a byte príznakov. Z nášho hľadiska sú zaujímavé príznaky CY a Z (nultý a šiesty bit príznakov).

Program na testovanie vlastností podprogramu ENTWD uvedieme s využitím návratu do monitora:

```
8200 CP3:   CALL  ENTWD  ;cvičný program č.3 - začiatok
8203        RST   4      ;návrat do monitora
```


V podprograme ENTWD zadáme napr. šesť číslic 0, 1, 2, 3, 4, 5 a zadávanie ukončíme klávesom S . Po návrate do monitora nájdeme v akumulátore kód 13_H/S/ , v registri D počet stlačených klávesov /6/ a v dvojici HL hodnotu 2345_H (posledné štyri stlačené).

Ak chceme zobraziť iné znaky ako hexadecimálne číslice, treba zaviesť do príslušnej bunky pamäti, ktorá patrí danému miestu displeja, zodpovedajúcu kombináciu bitov. Takto možno rozsvietiť ktorékoľvek segmenty osemsegmentového displeja, včítane bodky. Kódovanie segmentov je uvedené v [1] na str. 47. Zostavíme program na overenie tohto kódovania. Uvedieme pritom aj jeho strojový tvar vzhľadom na to, že nie je triviálny:

```
8200 21F883  CP4:   LXI  H,83F8H  ;nastav ľavú pozíciu
                                  ;displeja
8203 3E01           MVI  A,1      ;nastav len nutný bit
                                  ;reg.A
8205 77      ULOZ:  MOV  M,A      ;ulož do pamäti
8206 37             STC           ;priprav CY=1
8207 17             RAL           ;posuň doľava a použi CY
8208 F5             PUSH PSW      ;schovaj pripravený akum.
8209 CD3D02         CALL GETKY    ;počkaj na stlačenie
                                  ;klávesu
820C F1             POP  PSW      ;obnovenie akumulátora
820D 23             INX  H        ;nová pozícia displeja
820E C30582         JMP  ULOZ     ;návrat na uloženie akum.
```

Program postupne vysvecuje ďalšie segmenty displeja tak, že obsah akumulátora sa postupne sprava napĺňa jedničkami. Každý nový obsah akumulátora sa zobrazí na ďalšiu pozíciu displeja (zabezpečí obsah HL). Podprogram GETKY slúži len

na čakanie na stlačenie klávesu a nevyužíva sa na vstup kódu. Obsah akumulátora treba uchovať, na čo slúžia inštrukcie PUSH PSW a POP PSW. Program možno ukončiť klávesom RESET a spustiť znova klávesom "G". Je zrejmé, že segmenty sa kódujú zhora v smere hodinových ručičiek (od nižších bitov). Tak napríklad písmeno "U" možno rozsvietiť kombináciou 00111110 t.j. $3E_H$. Dosiahneme to napríklad programom, ktorý využije pravú pozíciu displeja:

```
8200  3E3E    CP5:  MVI  A,3EH   ;naplň akumulátor 0011 1110
8202  32FF83          STA  83FFH  ;ulož na 83FF_H
8205  76              HLT         ;zastav procesor
```

Pozn. prepínač STEP/AUTO musí byť v polohe AUTO!

Na mazanie obsahu displeja sú určené nasledujúce podprogramy. Ich úlohou je vynulovať obsah pamäťových buniek (ktorých príslušné pozície displeja sa mali zmazať). Sú zostavené takto:

```
0282  CLRGT:  MVI  B,4      ;nastav počet = 4
0284          JMP  0289H    ;obíď nasledujúcu inštrukciu
0287  CLEAR   MVI  B,8      ;nastav počet = 8
0289          LXI  H,83FFH  ;pravá pozícia displeja
028C  CLRLP   MVI  M,0      ;vynuluj bunku pamäti
028E          DCX  H        ;zníž ukazovateľ pamäti
028F          DCR  B        ;parameter cyklu dekrementuj
0290          JNZ  CLRLP    ;opakuj cyklus podľa reg.B
0293          RET           ;koniec cyklu, návrat
```

Podprogram CLRLP nuluje bunky pamäti od adresy H,L nižšie, pričom v registri B je počet nulovaných buniek (max. 256). CLRLP je všeobecne použiteľný podprogram na nulovanie špecifikovanej oblasti RWM pamäti. CLEAR nuluje celý displej a CLRGT nuluje len pravé štyri pozície (pozor na chyby textu v [2]). Treba poznamenať, že podprogramy nerušia činnosť DMA kanála, len mažú zobrazované údaje.

Monitor obsahuje aj podprogramy na programové zdržanie. Ich jadrom je prázdny cyklus:

```
0236  DELAY:  MVI  A,83H   ;naplň konštantu pre  1 ms
0238  DELYA:  DCR  A       ;dekrementuj parameter cyklu
0239          JNZ  DELYA    ;skok cyklu
023C          RET           ;návrat
```

Podprogram DELYA vykonáva zdržanie podľa obsahu akumulátora. Jeho veľkosť môžeme vypočítať $t_{RET} + n.(t_{DCR}+t_{JNZ})$, pričom uvedené časy t značia čas vykonávania inštrukcií a n je začiatočný obsah akumulátora (pri zadaní A=0 beží cyklus 256 krát). Čas trvania inštrukcie zistíme z počtu jej taktov (napr. [1] str. 25), pričom jeden takt trvá 0,48828 us . Uvedený výpočet platí presne len ak nie sú monitorové prerušenia (prepínač STEP/AUTO v polohe AUTO) a nie je aktivovaný DMA kanál displeja (spôsobuje obyčajne len malú chybu). Pre oneskorenie 1 ms je vhodná konštanta 83_H , čo zabezpečuje podprogram DELAY. Najdlhšie možné oneskorenie získame zrejme vtedy, ak začiatočný obsah akumulátora je nula. Vtedy bude oneskorenie asi 1,88 ms. Ak vyžadujeme väčšie zdržanie, treba volať napr. DELAY viackrát v cykle. Zostavíme taký podprogram:

```
8000 CD3602   DELYH: CALL DELAY ; oneskorenie  1 ms
8003 2B       DCX  H            ;dekrementuj parameter
                                 cyklu
8004 7C       MOV  A,H          ;testovanie HL na nulu
8005 B5       ORA  L            ;testovanie HL na nulu
8006 C20080   JNZ  DELYH        ;opakovanie cyklu podľa HL
8009 C9       RET               ;návrat
```

Podprogram DELYH vykoná zdržanie podľa obsahu HL, pričom obsah HL značí čas v ms (len približne, lebo sme pritom neuvažovali čas vykonávania inštrukcií v DELYH-u). Pomocou tohto podprogramu možno dosiahnuť zdržanie aj 1 min. Poznamenajme ešte, že počas uvedených podprogramov by sa nemalo vyskytovať prerušenie (monitora alebo používateľa).

Medzi pomocné podprogramy monitora zaradíme SHLRT, SHLRZ, SHLRC, MENAB, COPY, ERROR. Prvé tri posúvajú obsah dvojice HL o jeden bit doprava (posuv doľava dosiahneme inštrukciou DAD H):

```
022D SHLRT: MOV  A,H  ;test HL na nulu
022E SHLRZ: ORA  L    ;test HL na nulu
022F SHLRC: MOV  A,H  ;daj H do pracovného registra
0230        RAR       ;posuv A doprava s použitím CY
0231        MOV  H,A  ;vráť do H posunutý obsah
0232        MOV  A,L  ;daj L do pracovného registra
0233        RAR       ;posuv A doprava s použitím CY
0234        MOV  L,A  ;vráť do L posunutý obsah
0235        RET       ;návrat
```


Podprogram SHLRC naplní uvoľnený (najvýznamnejší) bit H hodnotou CY, SHLRZ tento bit naplní nulou. SHLRT pracuje ako SHLRZ a navyše podľa obsahu HL pred posuvom nastaví príznak Z (Z=1 ak HL = 0). Vždy však v CY ostane najmenej významný bit L registra (ktorý posuvom doprava vypadne z formátu HL). Činnosť podprogramu MENAB sme vysvetlili pri zobrazovacích podprogramoch. Program COPY premiestni obsah jednej oblasti pamäti do inej oblasti (podrobnejšie pozri [2] str. 25). Podprogram ERROR vypíše na displeji text "Err" (predtým zmaže celý displej). Jeho funkciu môžeme overiť programom:

```
8200   CDBC00   CP6: CALL ERROR  ;volanie ERROR
8203   76       HLT              ;zastav procesor
```

Po zapísaní textu Err treba zastaviť procesor, aby sa displej neprekryl hlásením monitora. Opäť prepínač STEP/AUTO musí byť v polohe AUTO (po prerušení monitora by sa pokračovalo za inštrukciou HLT, čo si neželáme).

Používanie podprogramov monitora je výhodné, lebo tým možno dosiahnuť úsporu pamäťového priestoru a programátorských prác. Pri ich používaní treba uvažovať tieto aspekty:

- či súbor podprogramov poskytuje žiadanú funkciu a ktorý to konkrétne je (či nemá nežiadúce vedľajšie účinky)
- akým spôsobom sa vyvoláva a ako sa robí návrat (CALL, JMP)
- aké má vstupy a aké výstupy (príznaky, registre, pamäť)
- ktoré registre premazáva
- či umožňuje prerušenie

- ktoré (koľko) ďalších podprogramov volá (kvôli veľkosti zásobníka).

Záverom uveďme, že pri používaní podprogramov (všetkých) treba mať k dispozícii zásobník potrebnej hĺbky (na ukladanie návratových adries a prípadnú úschovu registrov) a to najmä vtedy, ak sú v podprogramoch volané ďalšie (vnorené podprogramy). Monitor zanecháva pre používateľa ukazovateľ zásobníka v RWM pamäti (vyhradenej pre monitor), takže používateľ sa nemusí starať o naplnenie ukazovateľa zásobníka, pokiaľ nemá na rozsah zásobníka veľké nároky (zásobník má aspoň 32 byteov, pokiaľ sa dodrží maximálny počet 8 programových zarážok).

4. Programovanie univerzálnych portov obvodu 8255

Na komunikáciu s okolím obsahuje školský mikropočítačový systém tri paralelné vstupno/výstupné obvody typu 8255. Funkcia obvodu 8255 je opísaná v lit. [1], preto sa obmedzíme len na zhrnutie základných princípov a niektorých aplikačných možností.

Pripojenie k zbernici ŠMS

Všetky tri vstupno/výstupné (ďalej I/O) obvody 8255 sú pripojené k zbernici mikropočítačového systému a čítanie a zápis do týchto obvodov je riadený signálmi I/OR a I/OW , preto je možné komunikovať s týmito obvodmi resp. s ich portami pomocou inštrukcií IN a OUT. Pod pojmom "port" tu (aj ďalej) budeme rozumieť jednoduchý kanál (vstupný alebo výstupný) s vyrovnávacím registrom. Slovenským ekvivalentom je výraz "brána".

Komunikácia s okolím

Na komunikáciu s okolím mikropočítačového systému má každý obvod 8255 k dispozícii 24 vstupno/výstupných signálov rozdelených na porty A, B a C, vyrovnávacie registre a riadiacu logiku. Charakter a funkciu jednotlivých I/O signálov obvodu 8255 je možné riadiť programovo nastavením jedného z troch možných režimov. Jednotlivé základné režimy možno charakterizovať nasledovne:

Režim 0 : základné vstupy/výstupy (nesynchronizované)

- dva 8 bitové porty (A, B)
- dva 4 bitové porty s možnosťou zápisu jednotlivých bitov (C)
- každý port môže byť vstupný alebo výstupný
- výstupné porty majú záchytný register (buffer)
- vstupné porty nemajú záchytný register (nesynchronizovaný prenos)

Režim 1: strobované vstupy/výstupy

- jeden alebo dva 8 bitové strobované porty (A,B) so záchytnými registrami
- každý port pracujúci v režime 1 má k dispozícii tri riadiace signály portu C a prerušovaciu logiku obvodu 8255 (na generovanie prerušovacieho signálu pre mikroprocesor)
- každý port môže byť vstupný alebo výstupný
- ak len jeden port pracuje v režime 1, zostávajúcich 13 bitov obvodu 8255 môže pracovať v režime 0
- ak pracujú v režime 1 dva porty (A, B), zostávajúce 2 bity portu C môžu pracovať ako v režime 0

Režim 2: strobovaný obojsmerný port (len A)

- jeden 8 bitový obojsmerný port A
- 5 riadiacich signálov
- prerušovacia logika obvodu 8255
- port A má záchytný register pre vstup aj pre výstup
- 11 zostávajúcich bitov (signálov) obvodu 8255 môže pracovať v režime 0 alebo 1.

Jednotlivé režimy sú znázornené na obr. 4.1. Časové priebehy jednotlivých strobovacích signálov a význam jednotlivých bitov riadiaceho registra je uvedený v lit.[1] na str. 60. V ďalšom sa obmedzíme na vysvetlenie režimov činnosti z programátorského hľadiska s možnosťou využitia v ŠMS VÚVT. Pretože v ŠMS VÚVT je možné využiť vo všetkých možných režimoch len port P1A v spolupráci s portom P1C, budeme sa v opise obmedzovať len na obvod 8255 č.1.

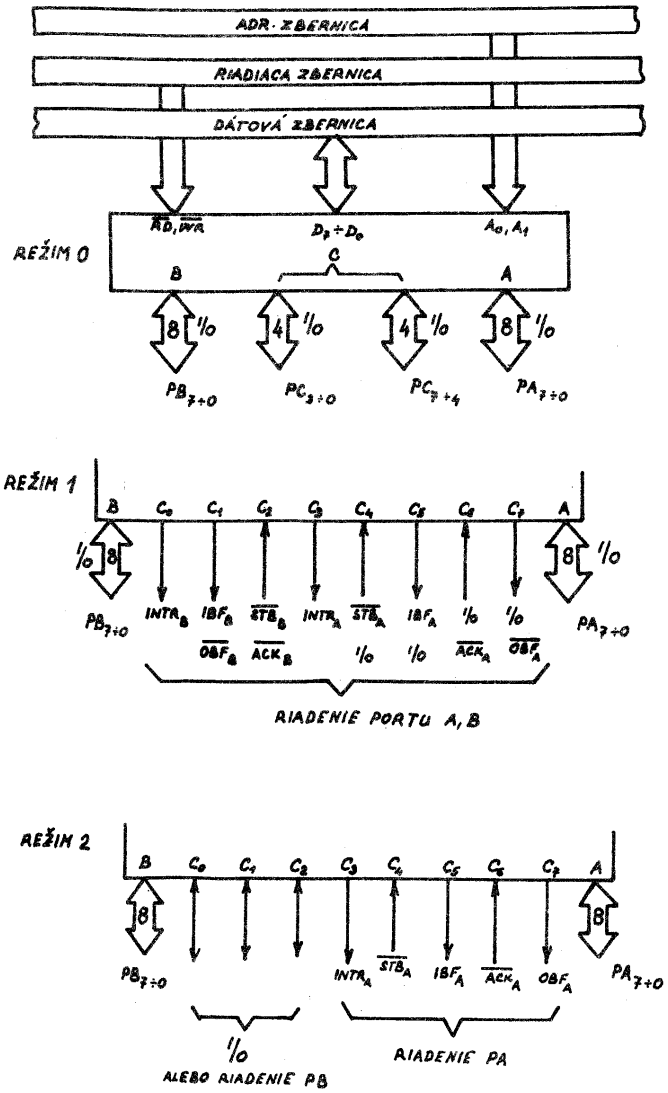
Nastavenie režimu

Nastavenie režimu obvodu 8255 sa vykonáva zápisom vhodného riadiaceho slova do riadiaceho registra. Nutnou podmienkou je, aby v najvyššom (siedmom) bite riadiaceho registra bola jednotka. V prípade, že v siedmom bite je zapísaná nula, jednotlivé bity riadiaceho registra majú úplne iný význam a slúžia na nastavenie jednotlivých bitov portu C (ak je výstupný).

Príklad 4.1

Nastavme obvod 8255 č.1 nasledovne:

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



Riadiaci register musíme nastaviť na hodnotu:

1000 0111 B = 87 $_H$ (pozri [1] str. 56)

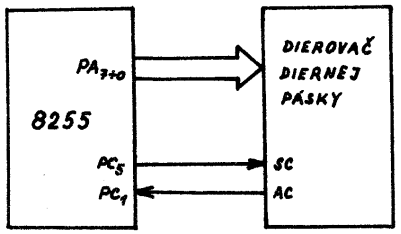
Zostavíme jednoduchý cvičný program na nastavenie obvodu 8255 č.1 podľa príkladu 4.1:

```
                ;nastavenie režimu 8255 č.1 podľa pr. 4.1
                ORG 8200H  ;začiatočná adresa
8200 3E87 CP7:  MVI  A,87H ;riadiace slovo pre 8255
8202 D307       OUT  7     ;výstup riadenia do 8255 č.1
8204 E7         RST  4     ;návrat do monitora
```

Program umiestnime na adresu 8200H. Po nastavení obvodu 8255 sa odovzdá riadenie monitoru. Po vykonaní programu zhasnú diódy LED portu PlA. Stlačením klávesu RESET sa diódy opätovne rozsvietia. Takéto správanie je zapríčinené tým, že nastavením hociktorého portu do výstupného režimu sa vynuluje záchytný register výstupu daného portu. Po signále RESET (ktorý sa vedie do všetkých obvodov 8255) sa automaticky všetky porty nastavia do vstupného režimu, pričom v tomto stave majú vývody obvodu 8255 vysokú impedanciu. To zapríčiní, že obvody budenia LED diód vyhodnotia tento stav ako log.1 (je to vlastnosť TTL obvodov, nie obvodu 8255) a preto diódy svietia. Zároveň to znamená, že po signále RESET možno porty používať ako vstupné bez zvláštneho nastavenia riadiacim slovom.

Režim 0

V ŠMS sa najčastejšie používa režim 0 pre všetky obvody 8255. Tento režim je vhodný na pripojenie zariadení, ktoré nepožadujú strobovacie signály. Aj v tomto režime je možné k ŠMS pripojiť prídavné zariadenia, ako napríklad snímač diernej pásky, dierovač diernej pásky, mozaikovú tlačiareň a podobne. Takéto riešenie je pomerne jednoduché. K portu A pripojíme údajový kanál vybraného zariadenia a pomocou portu C programovo nastavujeme a testujeme jednotlivé riadiace signály. Príklad takéhoto zapojenia je na obr. 4.2. Uvedená realizácia má však zvýšené nároky na čas a zložitosť programového ovládača daného zariadenia, čo možno pomerne značne zjednodušiť použitím režimu 1. Režim 0 nájde najširšie uplatnenie v riadení technologických procesov, kde v danom čase treba zmeniť úroveň napr. jediného signálu: zapnúť motor, uzavrieť ventil, rozsvietiť žiarovku a pod. Pritom hodnoty ostatných riadiacich signálov netreba meniť. Na takúto činnosť s výhodou využijeme možnosť nastavovania jednotlivých bitov portu C. Ak v riadiacom slove je najvyšší (siedmy) bit nulový, potom sa po zápise do riadiaceho registra interpretuje ako príkaz na nastavenie (nulovanie) jedného z bitov portu C (za predpokladu, že tento bit je nastavený ako výstupný). Na str. 55 v [1] je uvedený význam bitov riadiacich slov tohto typu.



Obr.4.2

Z uvedeného vyplýva, že programy na nastavenie a nulovanie bitu napr. PlCl budú pozostávať z dvoch inštrukcií:

```
                        ;postupnosť na nastavenie bitu P1C1
                        ;(výstupný)
MVI   A,00000011B       ;riadiace slovo na nastavenie P1C1
OUT   7                 ;prenos do riadiaceho registra CNT 1

                        ;postupnosť na nulovanie bitu P1C1
                        (výstupný)
MVI   A,00000010B       ;riadiace slovo na nulovanie P1C1
OUT   7                 ;prenos do riadiaceho registra CNT 1
```

Podčiarknuté bity pre orientáciu označujú číslo nastavovaného (nulovaného) bitu.

Niekedy je výhodné (šetrí pamäť) ovládať len jednotlivé bity výstupných portov A alebo B bez pamätania aktuálneho stavu výstupných portov v pamäti. Ako už bolo uvedené, každý port nastavený ako výstupný obsahuje záchytný register. Obsah tohto registra možno čítať inštrukciou IN (hoci port je vo výstupnom režime). Znamená to, že aktuálna hodnota portu je programovo prístupná v záchytnom registri obvodu 8255.

Zostavíme krátke programy na ukážku a/ nastavenia bitu (napr. P1A6), b/ nulovania bitu (P1A6), c/ zmeny stavu bitu na opačný stav (napr. P1A4). Program zostavíme tak, aby nadväzoval na cvičný program CP7, ktorý nastaví P1A do výstupného režimu. Jednotlivé zmeny bitov budeme pozorovať na LED diódach v ŠMS. Preto po každej zmene odovzdáme riadenie monitoru, aby sme mohli činnosť programu vizuálne pozorovať. Pokračovanie v programe dosiahneme stlačením klávesu "G". V programe najprv nastavíme P1A do výstupného režimu (programom CP7), potom nastavíme P1A6=1, ďalej nastavíme P1A6=0 a nakoniec zmeníme P1A4 na opačnú hodnotu:


```
                        ;nastavenie P1A6=1
                        ORG   8205H     ;hneď za CP7
8205 DB04   CP8:        IN    4         ;čítanie okamžitého sta-
                                        ;vu P1A
8207 F640               ORI   01000000B ;nastavenie 6.bitu
                                        ;akum.
8209 D304               OUT   4         ;výstup na port P1A
820B E7                 RST   4         ;volanie monitora

                        ;nulovanie P1A6=0
                        ORG   820CH     ;hneď za CP8
820C DB04   CP9:        IN    4         ;čítanie stavu P1A
820E E6BF               ANI   10111111B ;nulovanie 6.bitu
                                        ;akum.
8210 D304               OUT   4         ;výstup na port P1A
8212 E7                 RST   4         ;volanie monitora

                        ;zmena stavu P1A4
                        ORG   8213H     ;hneď za CP9
8213 DB04   CP10:       IN    4         ;čítanie pôvodného stavu
                                        ;P1A
8215 EE10               XRI   01000000B ;zmena 4.bitu na opač-
                                        ný
8217 D304               OUT   4         ;výstup na port P1A
8219 E7                 RST   4         ;volanie monitora
821A C30582             JMP   CP8       ;opakuj cyklicky od CP8
```

Uvedené programy odskúšame tak, že ich uložíme všetky do pamäti (aj CP7) a spustíme od adresy 8200_H. Stav výstupu portu P1A pozorujeme na LED diódach, pričom po každom kroku treba stlačiť kláves "G". Inštrukcia JMP na adrese $821A_H$ zaistí cyklické opakovanie činnosti programu. Program CP10 využíva vlastnosť operácie nonekvivalencia (XRI),

keď nulové bity priameho operandu inštrukcie XRI nemenia príslušné bity akumulátora a jednotkové bity priameho operandu spôsobia zmenu bitov akumulátora na príslušných miestach.

Režim 1 a 2

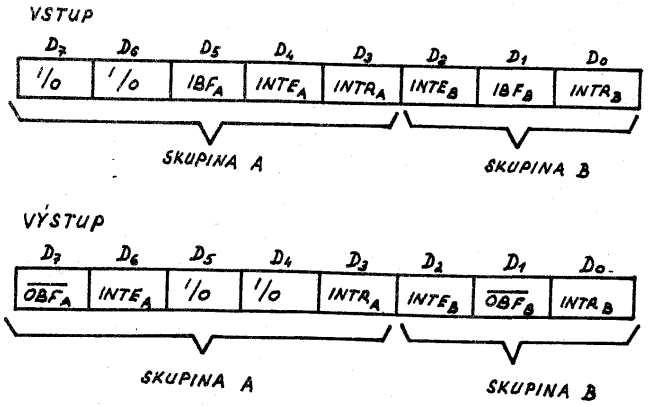
Režim 1 obvodu 8255 sa s výhodou využíva pri spojení mikropočítača s asynchrónnymi zariadeniami požadujúcimi pre komunikáciu strobovacie signály. V tomto režime nie je potrebné programovo ovládať riadiace signály na komunikáciu s prídavným zariadením. Tieto signály sú ovládané riadiacou logikou obvodu 8255 a navyše riadiaca logika je schopná generovať signál, ktorý možno zaviesť ako prerušovací do mikroprocesora (po skončení každého prenosu jedného byte medzi portom A alebo B a prídavným zariadením).

V režime 2 má obvod 8255 rovnaké vlastnosti ako v režime 1, ale port A možno pripojiť k zariadeniam s obojsmernou údajovou zbernicou (napríklad zariadenie alebo iný počítač). Pritom je k dispozícii 5 riadiacich signálov portu C.

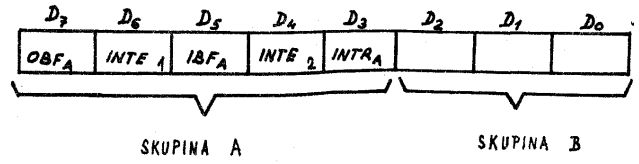
V ŠMS je možné využiť v režimoch 1 a 2 v spolupráci s portom P1C iba port P1A. Signál $INTR_A$ z portu P1C je vedený do prerušovacej logiky v ŠMS VÚVT. Ak sa má uplatniť prerušenie, musíme vopred nastaviť bit P2C6=1 (je to vlastnosť prerušovacieho systému, pozri časť 5 tejto príručky).

V režime 1 a 2 sú vývody z portu C nastavované ako riadiace signály vnútornou logikou obvodu 8255 a preto ich nemožno programovo ovládať. Výnimku tvoria len tie bity, ktoré nie sú v danej kombinácii režimov využité. Pri čítaní z portu C majú však jednotlivé bity priradený určitý význam. Port C sa v tomto prípade z programátorského hľadiska stáva stavovým registrom (prídavného zariadenia).

Čítaním tohto registra možno kedykoľvek zistiť, v akom stave sa nachádza vstupný alebo výstupný register (naplnený alebo nie), alebo či je generovaná žiadosť o prerušenie. Význam jednotlivých bitov v režime 1 pre vstup a význam je na obr. 4.3 a pre režim 2 portu A na obr. 4.4.



Obr. 4.3



Obr. 4.4

Bity IBF, OBF a INTR je možné programovo len čítať. Bity označené I/O a INTE je možné čítať a zapisovať v režime nastavovania jednotlivých bitov.

Žiadosť o prerušenie (t.j. INTR=1) sa generuje za predpokladu, že príslušný bit INTE=1.

V ŠMS VÚVT so základným sortimentom periférií nemožno jednoducho odskúšať režim 1 a 2 portu P1A. Nie je totiž k dispozícii zdroj jednotlivých impulzov. Pre používateľa, ktorý má k dispozícii prídavné zariadenia s medzistykom IRPR alebo podobným, je vyvedený port P1A a P1C na päticu PX (lit. [1] str. 12) a port P1A je vyvedený tiež na kontaktové pole 7KP a 8KP.

Pretože každý používateľ bude mať k dispozícii zrejme iný typ prídavného zariadenia, nechávame túto úlohu na samostatné riešenie. Pred akýmkoľvek laborovaním s portami prostredníctvom pätíc PX, PY, PZ je potrebné si uvedomiť, že sú pripojené priamo k obvodom 8255 bez elektronického oddelenia. Z toho vyplýva obmedzenie, podľa ktorého možno na jeden výstup pripojiť len jeden vstup TTL. V každom prípade pre výstupný prúd obvodu 8255 pri úrovni L musí platiť $I_{OL} \leqslant 1{,}7$ mA. Pri nedodržaní tejto podmienky môže dôjsť k znehodnoteniu obvodu 8255. Problém môže nastať pri pripojovaní štandardných prídavných zariadení, ktoré sú väčšinou na vstupoch opatrené prispôsobovacími odpormi. Odporúčame preto pri takomto prepojení výstupy obvodu 8255 oddeliť osobitnými budičmi napr. TTL.

Okrem toho pri manipulácii so vstupmi 8255 na uvedených konektoroch sa treba vyhýbať náhodnému výskytu statického náboja alebo iných zdrojov vyššieho napätia, pretože môže dôjsť k poškodeniu vstupov MOS obvodu (majú vysokú impedanciu a malú kapacitu).

5. Obsluha prerušení

Schopnosť procesora reagovať na vonkajšie prerušovacie signály podstatne zvyšuje výkonnosť celého výpočtového systému. Princíp prerušenia spočíva v tom, že vonkajším prerušovacím signálom sa preruší práve bežiaci program v procesore a začne sa vykonávať iný, ktorý je práve v tom čase dôležitejší. Po pominutí takýchto okolností sa obnoví vykonávanie prerušeného programu. Príkladom môže byť riadiaci počítač, ktorý spracováva daný program a v určitom čase treba prejsť na iný program, ktorý obsluhuje napr. havarijné stavy, lebo nastala taká situácia. Prerušenie sa však využíva aj v normálnej prevádzke najmä pri riadení periférnych zariadení. Ako príklad uvedieme riadenie tlačiarne v prípade, že program má pripravených niekoľko riadkov výsledkov a tlačiareň má vyrovnávací register na jeden riadok. Prerušenie sa využíva tak, že tlačiareň po vytlačení riadku vyšle prerušovací signál, procesor prejde na tzv. obslužný program prerušenia tlačiarne a po jeho vykonaní sa vráti späť do prerušeného programu. Program prerušenia tlačiarne odovzdá tlačiarni na vytlačenie ďalší riadok textu. Je zrejmé, že takto netreba čakať v hlavnom programe na ukončenie činnosti tlačiarne (prerušenie vzniklo pri ukončení činnosti prídavného zariadenia). Pomocou prerušení môžeme dosiahnuť paralelnú činnosť V/V podsystému a procesora. Ak by sa nevyužívalo prerušenie, bolo by treba stav napr. tlačiarne zisťovať inými prostriedkami, napríklad neustálym testovaním, či sa už ukončila činnosť (pri takomto testovaní beží testovací program a nie výkonný program). Prerušenia je vhodné využívať v prípade, že význačné stavy okolia procesora vznikajú asynchrónne k bežiacemu programu. V prípade mikroprocesorov, ktoré sú použité na riadenie technologických procesov, sa prerušenia využijú na hlásenie dôleži-

tých udalostí (uplynutie určeného časového úseku, dosiahnutie určitej hodnoty určitej veličiny v riadenom procese atď.).

Niekedy, pravdaže, je situácia, keď z nejakej príčiny nie je dovolené prerušenie (napr. beží časovo náročný program alebo sa obsluhuje rovnako dôležitá iná udalosť). V takom prípade programátor zaradí inštrukciu DI (disable interrupt), čím dosiahne, že procesor nebude reagovať na prerušovacie signály. Inštrukciou EI (enable interrupt) sa obnoví pôvodný stav. Inokedy naopak, netreba počítať žiaden program a treba čakať len na prerušenie. Na to je určená inštrukcia HLT (halt), po ktorej prejde procesor do vyčkávacieho stavu. Po výskyte prerušenia sa pokračuje podľa prerušenia.

Procesor 8080 má prerušenie fyzicky riešené ako jednoduchý vstup procesora. Po jeho výskyte (ak je stav EI) sa z procesora potvrdí akceptovanie prerušenia (signálom INTA), čo je znamením pre prerušujúce zariadenie, že má vyslať niektorú z inštrukcií RSTO ÷ RST 7. Tá sa okamžite vykoná a procesor samočinne ihneď prejde do stavu DI. Takto podľa inštrukcie RST možno rozoznať osem prerušujúcich zdrojov.

ŠMS VÚVT má sedem prerušovacích zdrojov, ale prerušovací systém nemá založený na používaní inštrukcií RST, ale inak. Vznik prerušovacieho signálu procesora INT možno vyjadriť logickou funkciou:

$$INT = P_1.U_1 + P_2.U_2 + P_3.U_3 + P_4.U_4 + P_5.U_5 + P_6.U_6 + P_7.U_7$$

kde P_i značí povolenie i-teho prerušenia a U_i značí výskyt i-tej prerušovacej udalosti. Programátor má teda možnosť pomocou signálov P_i dovoliť alebo zakázať prerušenie pri

výskyte i-tej prerušovacej udalosti. Prerušovacie udalosti sú tieto:

číslo udalosti	prerušovací zdroj
1	počítadlo z IO 8253 č. 0
2	počítadlo z IO 8253 č. 1
3	počítadlo z IO 8253 č. 2
4	obvody D/A prevodníka
5	externý vstup cez konektor EXT 4
6	externý vstup cez konektor EXT 5
7	bit č. 3 portu 1C

Výskyt signálov U_i možno definovať takto:

U_1 : signál vzniká a trvá ďalej po výskyte nábehovej hrany na výstupe počítadla č. 0. Signál sa ruší výstupom byte 0XXX 000X na adresu 0F .

U_2 : signál vzniká a trvá ďalej po výskyte nábehovej hrany na výstupe počítadla č.1. Signál sa ruší výstupom byte 0XXX 001X na adresu 0F .

U_3 : signál je totožný s výstupom počítadla č.2.

U_4 : signál trvá práve vtedy, keď napätie D/A prevodníka je väčšie ako vstupné analógové napätie (výstup komparátora).

U_5 : signál vzniká a trvá ďalej po výskyte nábehovej hrany vonkajšieho signálu EXT4. Signál sa ruší výstupom byte 0XXX 100X na adresu 0F .

U_6 : signál vzniká a trvá ďalej po výskyte nábehovej hrany vonkajšieho signálu EXT 5. Signál sa ruší výstupom byte 0XXX 101X na adresu 0F .

U_7 : signál je totožný s výstupom brány 1C, bit č.3.

Z uvedeného je zrejmé, že obvody vyhodnocovania (generovania) signálov U_1, U_2, U_5, U_6 obsahujú záchytné preklápacie obvody, takže prerušenie sa generuje aj po pominutí uvedených podmienok, ktorých čas trvania takto môže byť krátky. Okrem toho niektoré signály U_i sú privedené na vstupnú bránu 2B, takže programátor má možnosť programovo (bez prerušenia) zistiť stav týchto signálov. Jednotlivé bity portu 2B sú pripojené takto:

bit	pripojenie
0	na signál U_1
1	na signál U_2
2	na signál U_3
3	na signál U_4
4	na signál U_5
5	na signál U_6
6	na vonkajší signál EXT 4 (ešte pred záchytným preklápacím obvodom)
7	na vonkajší signál EXT 5 (ešte pred záchytným preklápacím obvodom)

Uvedené usporiadanie umožňuje programátorovi pri prerušení zistiť ktoré zariadenie prerušovalo (bity 0 ÷ 5), prípadne či podnet na vytvorenie prerušovacieho signálu ešte stále trvá (bity 6 ÷ 7). Toto zisťovanie vykoná čítaním z portu 2B inštrukciou IN z adresy 0D. Treba ešte poznamenať, že vstupy EXT 4 a EXT 5 (aj obvody generovania U_4) sú vystrojené Schmittovými obvodmi s hysteréziou, takže malé kolísanie týchto signálov okolo rozhodovacej úrovne nespôsobí mnohonásobný vznik prerušení.

Uplatnenie prerušovacích signálov U_i je závislé od hodnoty povoľujúcich signálov P_i (aj od stavu procesora EI resp. DI). Povoľujúce signály P_i sú výstupmi portu 2C

podľa tabuľky:

bit	význam log. 1 bitu
0	povolenie U_1 (= signál P_1)
1	povolenie U_2 (= signál P_2)
2	povolenie U_3 (= signál P_3)
3	povolenie U_4 (= signál P_4)
4	povolenie U_5 (= signál P_5)
5	povolenie U_6 (= signál P_6)
6	povolenie U_7 (= signál P_7)
7	nastavenie P_1 ÷ P_7 na log. 0 (zákaz všetkých prerušení)

Hardwareové riešenie obvodov portov 2B a 2C je zaujímavé tým, že adresa riadiaceho slova portu 2C (OF) je zhodná s adresou obvodov na nulovanie záchytných preklápadích obvodov (tiež je OF). Toto riešenie umožňuje jednou inštrukciou OUT zároveň povoliť prerušenie (napr. U_2) a zároveň vynulovať záchytný register (aby sa prejavili len nasledujúce udalosti). Uvedieme niekoľko príkladov:

Vynulovanie záchytného registra (pre U_2) a povolenie zodpovedajúceho prerušenia dosiahneme týmto výstupom:

```
MVI   A,00000011B
OUT   0FH
```

pričom sa zároveň 1. bit portu 2C nastavil do log. 1 a tiež sa vynuloval príslušný záchytný register. Zákaz prerušenia pre U_4 dosiahneme výstupom:

```
MVI   A,00000110B
OUT   OFH
```

Povolenie prerušenia bez nulovania záchytného registra (zaujíma nás výskyt prerušovacích udalostí v minulosti) dosiahneme výstupom:

```
MVI   A,XXX1XXXXB
OUT   OEH
```

čím sme povolili prerušenie od U_5 (namiesto znakov X treba dosadiť želanú hodnotu ostatných signálov P_i). V poslednom príklade sme zaviedli na výstup portu 2C celý byte (všetky P_i) a v predchádzajúcich príkladoch sme nastavovali len jednotlivé bity portu 2C (pozri programovanie obvodu 8255). Poznamenajme, že výstupom byte 00001010 na adresu OF dosiahneme síce zákaz prerušenia pre U_6, ale zároveň vynulujeme záchytný preklápací obvod, takže v budúcnosti budeme môcť rozhodnúť či od tohto okamihu sa vyskytla prerušovacia udalosť alebo nie (zachytí sa v preklápacom obvode). Takéto zistenie možno vykonať aj bez prerušenia čítaním z portu 2B.

Zrekapitulujme teda možnosti, ktoré poskytuje prerušovacia logika ŠMS VÚVT:

- existuje 7 prerušovacích zdrojov (3x počítadlá, D/A, 2x externý signál, programové prerušenie)
- jednotlivé prerušenia možno jednotlivo alebo všetky povoliť alebo zakázať bitmi portu 2C
- stav prerušovacích signálov U_i možno zisťovať programovo vstupom z portu 2B a ich výskyt možno obsluhovať v v poradí podľa dôležitosti, ktorá sa môže aj meniť (programové riešenie priority obsluhovania)

- niektoré prerušovacie signály U_i (i=1, 2, 5, 6) sú zachytávané preklápacími obvodmi a tak možno registrovať aj ich výskyt v minulosti (preklápacie obvody zachytávajú po nábehovej hrane)
- záchytné preklápacie obvody možno jednotlivo nulovať
- pri externých signáloch EXT 4 a EXT 5 možno programovo zisťovať zachytené prerušovacie stavy (bity 5,4 portu 2B) aj okamžité stavy (bity 6,7 portu 2B).
- niektoré signály U_i (i=4, 5, 6) sú odolné proti šumeniu z privedeného signálu (vykazujú hysteréziu)
- prerušovacie stavy možno zisťovať programovo aj bez uplatnenia prerušenia čítaním stavu z portu 2B (platí poznámka o zachytení stavov v preklápacích obvodoch)

Uvedený prerušovací systém je pomerne výkonný (poskytuje mnoho možností pre používateľa) a preto jeho technické prostriedky sú zložitejšie ako konvenčné riešenie pomocou RST-inštrukcií.

Uplatnenie prerušovacieho signálu U_1 spôsobí prerušenie typu RST 5, ostatné signály U_i (i= 2, 3, 4, 5, 6, 7) spôsobia prerušenie typu RST 6. Pri súčasnom výskyte RST 5 aj RST 6 sa uplatní RST 5 (má vyššiu hardwareovú prioritu tvorenia ako RST 6). Pre úplnosť dodajme, že monitor používa pre svoje potreby (napr. režim STEP) prerušenie typu RST 7, ktoré má ešte nižšiu prioritu ako RST 6. Ak sa teda naraz vyskytne používateľské aj monitorové prerušenie, uplatní sa používateľské.

Určitým problémom je skutočnosť, že vykonaním inštrukcie RST n (pri prerušení) sa prejde na adresu n.8, t.j. do oblasti ROM pamäti, kde nemôže používateľ umiestniť svoj obslužný program. Toto sa rieši využitím nepriamej adresy tak, že na dohodnutom mieste RWM pamäti je uložená

výkonná (efektívna) adresa obslužného programu prerušenia daného typu. Na adrese n.8 prirodzene musí byť program, ktorý si vyberie túto efektívnu adresu a prejde na ňu. Tento pomocný program je súčasťou monitora a napr. pre RST 5 je takýto:

```
0028 E5       RST5: PUSH H        ;ulož HL na vrchol zásob-
                                  ;níka
0029 2AEC83         LHLD 83ECH    ;daj do HL efektívnu
                                  ;adresu
002C E3             XTHL          ;vymeň HL a vrchol zá-
                                  ;sobníka
002D                RET           ;skoč na efektívnu adre-
                                  ;su
```

Je zrejmé, že efektívna adresa musí byť uložená na adrese $83EC_H$. Ľahko sa možno presvedčiť, že uvedený program nemení obsah žiadneho registra (ani SP) a ani obsah zásobníka. Preto obslužný program možno písať tak, ako by bol na adrese 0028_H. Obslužný program prerušenia takto možno napísať a umiestniť kdekoľvek v pamäti, treba však na adrese $83EC_H$ uložiť jeho adresu. Podobné programy sú uložené na adresách 0030_H a 0038_H teda pre RST 6 a RST 7. Rozdiel je v tom, že efektívna adresa pre RST 6 sa predpokladá na adrese $83EA_H$ a pre RST 7 na adrese $83E8_H$. Pravda, RST 7 je určené pre monitor a obvody ŠMS (používateľská prerušovacia logika) negenerujú RST 7. Monitor pri svojej inicializácii ukladá na adresu $83EC_H$ automaticky adresu 8228_H a na adresu $83EA_H$ ukladá 8230_H. Je vhodné zvoliť tieto adresy aj pri cvičnom programovaní, lebo sa čiastočne podobajú na pôvodné adresy pre RST 5 a RST 6 a pri stlačení klávesu RESET sa aj tak dosadí táto hodnota.

Na ukážku obsluhy prerušenia uvedieme jednoduchý cvičný program, v ktorom využijeme tzv. programové prerušenie signálom U_7. V praxi, pravdaže, nikdy netreba vytvoriť prerušenie programovými prostriedkami (stačí priamo zaradiť príslušnú RST inštrukciu). Hlavný program očakáva stlačenie niektorého klávesu, pričom po stlačení sa vytvorí prerušenie. Obslužná rutina prerušenia má za úlohu odmerať, ako dlho bol kláves stlačený. Hlavný program umiestnime na adresu 8200_H a obslužný podprogram bezprostredne za ním:


```
                    ;cvičný program na ukážku obsluhy
                     prerušenia
                    ORG  8200H    ;ukladacia adresa programu
8200 3E92    CP11:  MVI  A,92H    ;nastavenie 8255 č.1 na
8202 D307           OUT  7        ; A-in, B-in, C-out
8204 D30F           OUT  0FH      ;a rovnako aj 8255 č.2
8206 3E40           MVI  A,40H    ;nastavenie P7 na "1",
                                  ;t.j.
8208 D30E           OUT  0EH      ;povolenie prog.preruše-
                                  ;nia
820A 211E82         LXI  H,OBSL   ;adresa obsluž.podprogra-
                                  ;mu
820D 22EA83         SHLD 83EAH    ;jej uloženie na 83EAH
8210 FB             EI            ;nastavenie procesora do
                                  ;EI
8211 CD5702  CYKL:  CALL SCAN     ;bol stlačený kláves?
8214 D21182         JNC  CYKL     ;ak nie, znova testuj cez
                                  ;SCAN
8217 3E07           MVI  A,00000111B ;nastav U7=P1C3 na
                                  ;log. 1,
8219 D307           OUT  7        ;t.j. vytvor prog.preru-
                                  ;šenie
821B C31182         JMP  CYKL     ;opakuj cyklus
821E 210000  OBSL:  LXI  H,0      ;nastav počítadlo na nulu
8221 CDD102         CALL DWORD    ;zobraz nulu
```

```
8224 CD3602   NAV:  CALL DELAY   ;milisekundové zdržanie
8227 23             INX  H       ;inkrementuj počítadlo
                                 ;HL
8228 CD5702         CALL SCAN    ;kláves je ešte stlače-
                                 ;ný?
822B DA2482         JC   NAV     ;ak áno, opakuj cyklus
                                 ;od NAV
822E CDD102         CALL DWORS   ;zobraz obsah počítadla
                                 ;HL
8231 3E06           MVI  A,00000110B ; nastav U7 na log.
                                 ;0,
8233 D307           OUT  7       ;t.j. zruš signál prog.
                                 ;prerušenia
8235 FB             EI           ;obnov stav EI proceso-
                                 ;ra
8236 C9             RET          ;návrat z prerušenia
```

Na začiatku programu treba nastaviť V/V obvody 8255 do potrebného stavu. Potrebujeme P1C vo výstupnom režime (P1C3-programové prerušenie) a P2C tiež vo výstupnom režime (na povolenie prerušení). Ostatné porty uvedených obvodov nevyužijeme a preto nech sú vo vstupnom režime. Tomu zodpovedá pre obidva obvody riadiace slovo 92_H, ktoré zavedieme na príslušnú adresu 07 resp. $0F_H$. C-porty po nastavení na výstupný režim majú začiatočný stav na výstupe nula. Preto P_7=0 a U_7=0. Programové prerušenie povolíme nastavením P_7=1, t.j. P2C6=1. Dosiahneme to zavedením byte 0100 0000 na port P2C. Ešte treba uložiť na adresu $83EA_H$ adresu obslužného podprogramu prerušenia OBSL, a nastaviť procesor do stavu EI (fakticky to v ŠMS nie je treba, lebo monitor tento stav zanecháva vždy po vstupe do programu používateľa po príkaze G). Potom nasleduje cyklus, v ktorom sa pomocou podprogramu SCAN zisťuje, či je alebo nie je stlačený kláves. Ak je, nastaví

sa P1C3=1 riadiacim slovom 07 (adresa bitu je podčiarknutá). Po vzniku prerušenia sa vynuluje dvojica HL, ktorá bude slúžiť ako počítadlo milisekúnd a zobrazí sa nula na displeji. Potom nasleduje cyklus, v ktorom sa využíva milisekundové zdržanie, inkrementuje sa počítadlo a zisťuje sa, či je kláves ešte stále stlačený. Ak nie, hodnota počítadla sa zobrazí a zruší sa signál programového prerušenia, teda U_U=0. Až potom možno obnoviť stav EI (inak by sa prerušenie znova uplatnilo) a nakoniec sa vykoná návrat do prerušeného programu za miesto prerušenia (pri inštrukcii RST sa do zásobníka uložila návratová adresa).

Uvedený program je len kvalitatívny, zobrazená hodnota je šestnástková a okrem toho nepresná, pretože v milisekundovom cykle sa vykonáva podprogram SCAN, ktorý trvý istý čas (s ním sa v časovej bilancii cyklu nerátalo). Ďalšie príklady na obsluhu prerušení uvedieme pri riadení periférií ŠMS.

6. Programovanie univerzálnych počítadiel obvodu 8253

V ŠMS VÚVT je použitý jeden univerzálny časovač 8253 zo stavebnice 8080. Obvod 8253 obsahuje tri navzájom nezávislé 16 bitové počítadlá, počítajúce impulzy s frekvenciou 0-2 MHz. Každé počítadlo môže pracovať v hociktorom zo šiestich režimov, ktorý sa dá nastaviť programovo. Funkcia obvodu v jednotlivých režimoch je podrobne opísaná v [1] a preto sa obmedzíme na aplikácie obvodu.

Univerzálne počítadlá môžu v ŠMS vykonávať nasledovné funkcie:

- hodiny reálneho času, generovanie prerušenia v určitých, presne stanovených časových intervaloch (napr. režim 2)

- monostabilný preklápací obvod s možnosťou štartovania ešte počas trvania minulého impulzu (napr. režim 1)
- vytvorenie časového intervalu (napr. režimy 0, 4, 5)
- astabilný multivibrátor s nastaviteľnou frekvenciou (periódou) výstupného signálu (režimy 2, 3)
- počítadlo vonkajších udalostí, pracujúce v rôznych režimoch činnosti (režimy 0, 1, 4, 5)

Uvedený prehľad nie je úplný a predstavuje riešenie typických problémov pri návrhu mikropočítačového systému. Možno konštatovať, že obvod 8253 rieši problémy počítania a časovania. Preto sa v literatúre často stretneme s pojmom "univerzálne počítadlo a časovač".

Obvod 8253 zaberá 4 adresy adresného priestoru vstupno/výstupných zariadení. Tieto štyri adresy sa líšia v najnižších dvoch bitoch. Jednotlivým adresovateľným prvkom obvodu sú pridelené v ŠMS VÚVT nasledovné adresy:

počítadlo T0	- 14_H
počítadlo T1	- 15_H
počítadlo T2	- 16_H
riadiaci register	- 17_H

Režim každého počítadla sa nastavuje zápisom určitého byte do riadiaceho registra. Význam jednotlivých bitov riadiaceho slova je uvedený v [1] na str. 70.

Na školskom mikropočítačovom systéme budeme použitím obvodu 8253 riešiť najčastejšie tieto úlohy:

a) generovanie časového intervalu
b) generovanie tónu v reproduktore (t.j. generovanie frekvencie)

c) počítanie vonkajších udalostí

V úlohách skupiny b) budeme vždy používať režim 3. Takými úlohami sa budeme podrobne zaoberať v časti 10.

Pri riadení technologických procesov sa často stretávame s problémom vykonávania daných operácií po určitom čase napr. každú jednu milisekundu treba odmerať určitú hodnotu. Zostavme jednoduchý program, ktorý na výstupe T0 OUT generuje impulzy s periódou 2 ms.

Na počítačí vstup počítadla T0 je cez drôtovú prepojku vedený signál ϕ2 TTL s frekvenciou 2048 kHz (t.j. periódou asi 488 ns). Pre náš požadovaný časový interval T = 2 ms platí:

$$T = \frac{4096}{2048}\ ms$$

Z toho vyplýva, že hodinový signál ϕ2 treba vydeliť číslom 4096, čím dostaneme požadovanú frekvenciu 500 Hz. Preto predvoľba v režime 3 (žiadame opakované generovanie impulzov) má byť 4096 dekadicky, t.j. 1000_H. Zvolíme teda riadiaci byte 0011 0110 B = 36_H, t.j. použijeme počítadlo T0 v režime 3 ktoré bude počítať binárne, pričom zadáme dva byte predvoľby. Zapíšme časť programu na takéto nastavenie počítadla T0:

```
                    ;program na generovanie signálu s pe-
                    ;riódou
                    ;2 ms v režime 3
                    ORG  8200H  ;začiatok programu
8200 3E36  CP12:    MVI  A,36H  ;riadiaci byte pre 8253
8202 D317           OUT  17H    ;výstup riadenia do 8253
8204 3E00           MVI  A,0    ;nastavenie LSB
8206 D314           OUT  14H    ;výstup LSB do T0
8208 3E10           MVI  A,10H  ;nastavenie MSB
```

```
820A D314       OUT  14H     ;výstup MSB do T0
820C E7         RST  4       ;návrat do monitora
```

Po spustení a vykonaní programu sa na výstupe T0 generuje počuteľná frekvencia 500 Hz, o čom sa môžeme presvedčiť pripojením reproduktoru na svorky T0 a GND.
V programe sme mohli použiť skrátené zadanie predvoľby (len MSB), pretože LSB je nulový. Tú istú funkciu dosiahneme nastavením počítania v dekadickom kóde riadiacim slovom 37_H (namiesto 36_H) a zadaním predvoľby 4096 (MSB=40 a LSB=96).

Takýmto jednoduchým programom môžeme vytvoriť signál s periódou až do 32 ms, čomu zodpovedá plný rozsah počítadla 2^{16}. V rozsahu 5 až 32 ms však môžeme použiť iba binárne počítanie (štyrmi hexadecimálnymi číslicami FFFF vyjadríme väčšie číslo ako štyrmi dekadickými 9999).

Väčší časový interval môžeme vytvárať viacerými spôsobmi. Jeden spôsob je programový, keď výstupom počítadla generujeme prerušenia, ktoré počítame programovým počítadlom. Príklad je uvedený v čl. 13.

Ďalší spôsob spočíva v prepojení výstupu jedného počítadla s počítacím vstupom iného počítadla. Toto riešenie je jednoduché, treba však odmontovať plexikryt ŠMS.

V praxi sa často môžeme stretnúť s úlohami počítania vonkajších udalostí a na základe určitého počtu takýchto udalostí vykonať určitú operáciu. Pre názornosť si môžeme prestaviť baliacu linku riadenú mikropočítačom. Mikropočítač riadi hlavným programom linku na prísun výrobkov. Po prísune daného počtu výrobkov, snímaných napr. fotoelektricky, je potrebné prísun materiálu zastaviť a vykonať operáciu balenia. Takúto úlohu možno pomerne jednoducho riešiť použitím obvodu 8253, ktorý po napočítaní daného počtu udalostí preruší vykonávanie hlavného programu.

Ak treba napočítať napr. 5 výrobkov, inicializačný program môže vyzerať takto:

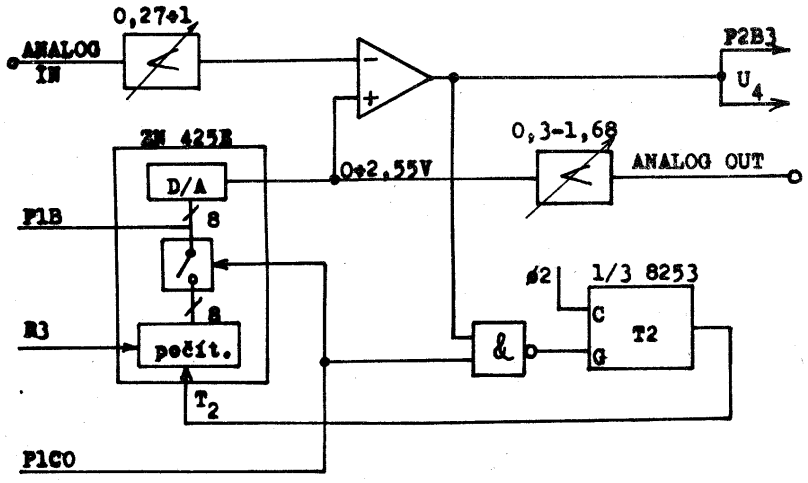
```
MVI   A,51H    ;nastavenie režimu nula počítadla T1 s de-
OUT   17H      ;kadickým počítaním a nastavením len LSB
MVI   A,5      ;zadanie počtu výrobkov (LSB)
OUT   15H      ;výstup LSB do T1
```

V ŠMS VÚVT možno počítať impulzy vonkajšieho signálu tak, že odstránime drôtovú prepojku pod plexikrytom na vstupe CLK počítadla T1 a na tento počítací vstup privedieme počítané impulzy napr. z výstupu EXT 4 OUT. Na vstup EXT 4 potom pripojíme zdroj impulzov, napr. fotoodpor podľa čl. 14. Upozorňujeme, že priame pripojenie mechanického kontaktu vytvára viacnásobné impulzy, čím sa naruší správne počítanie (je to vlastnosť kontaktov, nie obvodu 8253).

7. Programovanie analógového prevodníka a jeho vlastnosti

Školský mikropočítačový systém VÚVT obsahuje ako jeden z podsystémov aj zostavu analógového prevodníka, ktorý tvorí jeho periférium na vstup a výstup informácií v analógovej forme. Technicky je vyhotovený veľmi jednoducho, poskytuje však dostatok možností na cvičné programovanie jeho obsluhy.

Architektúru celého podsystému prevodníka možno vyjadriť blokovou štruktúrou podľa obr. 7.1. Použitý je jeden D/A prevodník (monolitický, osembitový), ktorý však v spolupráci s ostatnými obvodmi môže tvoriť aj A/D prevodník. Obvodové zapojenie celej štruktúry, parametre a ďalšie informácie možno nájsť v [1]. Tu sa sústredíme na otázky, ktoré budú zaujímať čitateľa v časti 12.



Obr. 7.1

Jadrom celej štruktúry je integrovaný obvod ZN 425E, ktorý obsahuje osembitový D/A prevodník, osembitové počítadlo a osembitový spínač medzi nimi tak, ako to ukazuje obr. 7.1. Obvod umožňuje priviesť osembitovú kombináciu na vstup D/A prevodníka zvonka, ale tiež z vnútorného počítadla. Voľba sa uskutoční vhodným prepnutím osembitového spínača, ktorý je riadený jednoduchým signálom. Výstup D/A prevodníka generuje (podľa privedeného kódu) napätia od 0 V do 2,55 V. Výstupné napätie z prevodníka sa ďalej vedie cez zosilňovač s meniteľným zosilnením na výstupné kontaktové pole s označením ANALOG OUT. Zosilenenie možno meniť v rozsahu 0,304 ÷ 1,68 a to potenciometrom s označením ANALOG OUT. Vonkajší osembitový vstup D/A prevodníka je v ŠMS pripojený na port P1B a osembitový spínač je riadený nultým bitom portu P1C.

Funkciu analógového výstupu možno dosiahnuť tak, že

spínač nastavíme do rozpojenej polohy (P1C0=0) a na port P1B programovo privedieme potrebnú osembitovú kombináciu. Rozsah výstupného napätia nastavíme ručne potenciometrom. Ostatné súčasti zapojenia pri D/A prevode nie sú v činnosti.

Použitie štruktúry z obr. 7.1 vo funkcii A/D prevodníka je komplikovanejšie. Možno využiť procesorom riadený A/D prevod alebo automatický režim, vždy však na princípe spätnej väzby medzi D/A prevodníkom a analógovým komparátorom, ktorý porovnáva vstupné merané analógové napätie s generovaným napätím z D/A prevodníka. Podstata zakéhoto prevodu spočíva v tom, že sa snažíme nájsť určité číslo pre D/A prevodník, pomocou komparátora zistiť či je väčšie ako merané napätie, potom určiť iné-presnejšie číslo, znova testovať atď.

A/D prevod možno riadiť programovo a to tak, že spínač nastavíme do rozopnutého stavu (P1C0=0) podobne ako pri D/A prevode. Potom programom podľa určitého algoritmu generujeme pre D/A prevodník čísla s tým, že pozorujeme výstup komparátora (tretí bit portu P2B). Takto možno využiť dve metódy prevodu - metódu postupných prírastkov a metódu postupnej aproximácie.

Pri metóde postupných prírastkov generujeme osembitové kombinácie pre D/A prevodník postupne od najnižšej (00000000) inkrementáciou po najvyššiu (11111111). Na začiatku prevodu komparátor zrejme oznámi, že generované číslo je menšie ako vstupné analógové napätie. Prevod sa ukončí vtedy, keď komparátor oznámi vyrovnanie resp. prekročenie analógového vstupného signálu číslom. Algoritmus je zrejme jednoduchý, ale pomalý, lebo v najhoršom prípade treba generovať 255 čísel (D/A prevodov a komparácií). Vstupný analógový rozsah je závislý na nastavení zosilnenia vstupného zosilňovača, ktoré možno ručne na-

staviť (potenciometrom ANALOG IN) v rozsahu 0,277 až jedna. Takto možno dosiahnuť vstupný napäťový rozsah A/D prevodníka od nula do 2,55V/0,277. Záporné napätia sa vyhodnotia ako nulová kombinácia, napätia vyššie ako rozsah prevodníka je vhodné vyhodnotiť ako samé jedničky (závisí od algoritmu riadenia!). Podobné poznámky platia aj pre ostatné spôsoby prevodu.

Metóda postupných aproximácií je založená na myšlienke, že treba tak riadiť prevod, aby komparátor pri jednej komparácii dodal čo najväčšie množstvo informácií, t.j. 1 bit. Potom totiž možno prevod urobiť na osem krokov (a nie až 255). To sa dosiahne vtedy, ak komparátor rozhodne z dvoch rovnako pravdepodobných možností. Najprv teda necháme rozhodnúť v ktorej polovici rozsahu je vstupné napätie, potom v ktorej štvrtine, v ktorej osmine atď. Pritom sa binárne číslo tvorí od najvyšších bitov (po každom kroku je známy ďalší bit). Konkrétne programové riešenie obidvoch uvedených metód pozri časť 12.

A/D prevod sa môže vykonávať aj automaticky a to metódou postupných prírastkov. Pri tomto režime nastavíme spínač v D/A prevodníku do zopnutej polohy a port P1B nastavíme do vstupného režimu. Na začiatku prevodu sa počítadlo v D/A prevodníku vynuluje signálom R3. Počítadlo T2 z IO 8253 generuje pravidelne impulzy, takže počítadlo v D/A prevodníku postupne inkrementuje svoj obsah. Dej sa opakuje dovtedy, kým komparátor nevyhodnotí vyššie napätie D/A prevodníka ako je vstupné napätie. Tým vznikne na výstupe komparátora log. 1, takže súčinový obvod zastaví ďalšiu činnosť (P1CO=1) privedením log. 0 na vstup G počítadla 8253. Tým je A/D prevod ukončený a v počítadle D/A prevodníka je kód meraného napätia. Odtiaľto ho možno programom načítať cez port p1B. Automatický prevod je výhodný v tom, že procesor v čase prevodu môže vykonávať inú činnosť (paralelne). Je však potrebné oznámiť procesoru (hlavnému programu), že sa ukončil automatický A/D prevod. To možno vykonať dvomi spôsobmi:

- v časove nenáročnom úseku programu možno testovať bit P2B3, pričom hodnota log. 1 indikuje ukončený A/D prevod
- signál z komparátora môže vyvolať prerušenie (signálom U_4) a v prerušovacej rutine možno načítať kód napätia portom P1B

Treba poznamenať, že obvod T_2 8253 musí generovať vhodnú frekvenciu (treba ho naprogramovať) a že ak vstupné napätie presiahne napäťový rozsah A/D prevodníka, komparátor nikdy nevyhodnotí ukončenie prevodu (počítadlo v D/A prevodníku je cyklické) a nevznikne ani prerušenie. Nulovací signál R3 možno generovať programom tak, že sa vykoná výstup byte 0000011X na adresu OF_H. Zároveň (pozri čl. 5) sa týmto byte povolí alebo zakáže prerušenie od ukončenia automatického A/D prevodu (podľa hodnoty nultého bitu uvedeného byte).

Pri všetkých metódach A/D prevodu treba dbať na to, aby D/A prevodník a komparátor mali dodržané časové parametre, tzn. že výstup komparátora treba testovať až po jeho ustálení. Prakticky to znamená, že po zadaní nového kódu do D/A prevodníka treba počkať na ustálenie výstupu D/A prevodníka a potom na ustálenie výstupu komparátora. Obidva uvedené časy tvoria spolu 300 - 400 us.

Pri cvičnom programovaní treba použiť vhodné vstupné analógové zariadenie. Niektoré námety sú uvedené v čl.14. Pripojenie skutočného analógového procesu k ŠMS treba preskúmať z hľadiska elektronických parametrov resp. použiť prevodník signálov (ŠMS nespracováva zápor-

né napätia). Detailná elektronická schéma je v [1] na str. 86 (výpočty zosilnení obvodov nie sú v texte uvedené správne).

Pre úplnosť dodajme, že nastavenie portu P1B do výstupného režimu a zopnutie spínača na obr. 7.1 nie je z logického hľadiska korektné a z elektronickej stránky ne neprípustné. Prepojíme tým totiž nakrátko výstupy portu P1B a počítadla v D/A prevodníku. Pri programovaní nesmieme preto pripustiť stav, aby port P1B bol výstupný a súčasne aby P1C0 = 1 . To nastane napr. aj vtedy, ak nastavíme P1B-výstupný a P1C-vstupný, pretože P1C0 ako vstupný bit má vysokú impedanciu a v dôsledku okolitých obvodov TTL dosiahne stav log. 1. Preto všetky riadiace slová obvodu 8255 č.1, ktoré nastavujú P1B-out a P1C/0÷3/-in treba považovať za neprípustné (t.j. 81_H, 89_H, 91_H , 99_H).

8. Používanie zosilňovača s optočlenom

Na ovládanie výkonových záťaží (do 10 W) je v ŠMS VÚVT k dispozícii výkonový zosilňovač, ktorý je od mikropočítačového systému oddelený optočlenom WK 16412. Takáto koncepcia umožňuje mikropočítačom ovládať zariadenia ako elektromotor, relé, žiarovka a pod. Na napájanie týchto zariadení je možné použiť externý zdroj, ktoré nemusia mať so ŠMS spoločný vzťažný potenciál. Zapojenie optočlena so zosilňovačom a podrobný opis je v lit. [1] na str. 89.

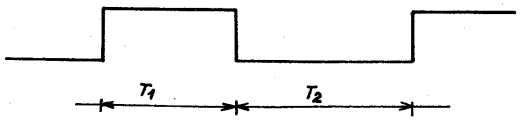
Medzi ovičnými perifériami, ktoré sú k dispozícii v ŠMS, je jednosmerný elektromotor. Tento motorček je možné napájať zo zdrojovej sústavy ŠMS. Rýchlosť otáčania je možné ovládať rôznym spôsobom pomocou optočlena a výkonového zosilňovača.

Riadenie bitom portu P1C1

Za predpokladu, že sú prepojené nasledovné kontakty:

MOT SUP$^+$, MOT CTL+ na +5V
MOT RET na GND
MOT RET, MOT DRV na svorky motora,

sa nulovaním bitu P1C1 roztočí motor na maximálne otáčky. Na určité príklady postačuje aj uvedená možnosť riadenia motora (vypnuté, zapnuté). Ak však požadujeme zmenu rýchlosti otáčania motora, musíme bit P1C1 ovládať impulzne. Na výstupe obvodu 7406 (obr. 2.9-1 v [1] na str. 89) dostaneme pulzujúce napätie podľa obr. 8.1, takže stredná hodnota budenia motora je daná pomerom času impulzu T_1 a času medzery T_2. Zmenou tohto pomeru dosiahneme zmenu otáčok motora. V praxi sa takéto riadenie rieši tak, že niektoré z uvedených časov sú konštantné



Obr.8.1

a riadi sa zmenou iného parametra (konštantné môže byť T_1 alebo T_2 alebo $T_1 + T_2$). Pri programovaní takýmto spôsobom možno výhodne použiť podprogramy monitora DELAY a DELYA, ale aj tvorenie časových intervalov univerzálnym počítadlom s prerušením.

Vzhľadom na to, že obvod 7406 je invertor s voľným kolektorom, možno pripojiť na kontakt MOT CTL- iný zdroj impulzov a nastaviť P1C1=1. Takýmto zdrojom môže byť ktorýkoľvek bit portu P1A (použiť len výkonové výstupy

z kontaktov 7KP a 8KP).

Riadenie motora D/A prevodníkom

Z programátorského hľadiska je jednoduché ovládať otáčky motora D/A prevodníkom. Výstup D/A prevodníka sa pripojí na kontakt MOT CTL+ , a na kontakt MOT CTL- sa pripojí GND alebo sa výstup P1C1 nastaví na nulu. Rýchlosť otáčania motora sa mení zmenou kódu D/A prevodníka (= port P1B). Pripojenie ostatných kontaktov výkonového zosilňovača ostáva nezmenené.

Ak použijeme na napájanie motora externý napájací zdroj, ktorý nemá s napájacou sústavou ŠMS spoločný potenciál, tak sa kladný pól externého napájacieho zdroja pripojí na MOT SUP^{+} a MOT RET sa pripojí na záporný pól externého zdroja. Takéto riešenie umožňuje galvanické oddelenie od zdrojovej sústavy ŠMS pomocou optoelektronického člena.

9. Cvičné programy k univerzálnemu portu P1A s LED diódami

Obvodom 8255 sme sa podrobnejšie zaoberali v článku 5., kde sme okrem vlastností a spôsobu používania tiež naznačili možné použitie týchto obvodov. Vieme teda, že vzhľadom na technické prostriedky ŠMS VÚVT má pre laborovanie praktický význam iba obvod 8255 č.1, z ktorého nás bude zaujímať práve port P1A, pretože tento je opatrený LED diódami na vizuálne pozorovanie. V ďalšom si zostavíme niekoľko programov, v ktorých ukážeme spôsoby práce s týmto obvodom. Keďže, ako už bolo spomenuté, je pre nás zaujímavé práve vizuálne pozorovanie na LED diódach, budeme v našich programoch využívať port P1A ako výstupný. Poznamenajme však, že vo vstupnom režime je

činnosť s portom obvodu z hľadiska programátora analogická. Rozdiel je len v riadiacom slove a v používaní inštrukcie IN, ktorou sa stav portu P1A (stav vstupu) prenesie do akumulátora.

Zostavme teraz jednoduchý program, ktorým zobrazíme binárnu hodnotu stlačeného klávesu na LED diódach portu P1A.

```
                    ;zobrazenie kódu klávesu na P1A
                    ORG   8200H  ;nastavenie ukladacej
                                  adresy
8200 3E80    CP13:  MVI   A,80H  ;riadiace slovo A-out,
                                  ;B-out, C-out
8202 D307           OUT   7      ;výstup do obvodu 8255
8204 CD3D02  N:     CALL  GETKY  ;vstup kódu klávesu
8207 D304           OUT   4      ;zobraz kód na LED
                                  t.j. P1A
8209 C30482         JMP   N      ;opakuj cyklus
```

V tomto programe využívame vlastnosti podprogramu monitora GETKY, ktorý čaká na stlačenie klávesu a jeho kód uloží do akumulátora. Inštrukciou OUT potom môžeme priamo tento kód vyslať na port P1A (ktorý bol nastavený do výstupného režimu) a zobraziť ho binárne LED diódami.

Zostavme ďalej podobný program, v ktorom si prakticky overíme vlastnosti niektorých podprogramov monitora. Na LED diódach portu P1A budeme zobrazovať kód stlačeného klávesu, ale iba počas stlačenia klávesu. Pravý polbyte kódu klávesu nech sa zobrazuje na displeji vľavo.

```
                    ;zobrazenie kódu na P1A použitím
                    ;programu SCAN
                    ORG  8200H  ;začiatok programu na
                    8200H
```

```
8200 3E80    CP14:  MVI   8200H  ;začiatok programu na
                                  ;8200H
8202 D307           OUT   7      ;výstup do obvodu 8255
                                  ;č.1
8204 11F883         LXI   D,83F8H ;adresa ľavého displeja
8207 CD5702  NO:    CALL  SCAN   ;test klávesnice
820A D20782         JNC   NO     ;ak nie je stlačený klá-
                                  ;ves, opakuj testovanie
820D D307           OUT   4      ;zobraz kód na PlA
820F CDA602         CALL  DISPR  ;zobraz časť kódu na
                                  ;displeji
8212 CD5702  N1:    CALL  SCAN   ;test klávesnice
8215 DA1282         JC    N1     ;opakuj ak je kláves
                                  ;stlačený
8218 3E00           MVI   A,00   ;nuluj akumulátor
821A D304           OUT   4      ;zhasni LED diódy
821C 1C             INR   E      ;obnov adresu displeja
821D CD8702         CALL  CLEAR  ;zmaž displej
8220 C30782         JMP   NO     ;opakuj cyklus
```

Aj v tomto programe najskôr nastavíme obvod 8255 do zvoleného režimu a potom do registrov DE pripravíme adresu bunky ľavého displeja. Použitie týchto registrov je dané vlastnosťami podprogramu DISPR, ktorý vyžaduje v DE dvojici adresu použitého displeja. Ďalej na testovanie klávesnice používame podprogram SCAN vzhľadom k tomu, že CY príznak signalizuje stlačenie niektorého klávesu. Chceme totiž vedieť, kedy kláves je a nie je stlačený. Ak je stlačený, zobrazíme jeho kód na porte PlA a na displeji. Ak kláves pistíme, displej aj port PlA zhasneme. Rozhodovanie sa vykoná inštrukciami JNC a JC na adresách $820A_H$ a 8215_H. Použitie inštrukcie INR E na adrese $821C_H$ vyplýva tiež z vlastností podprogramu DISPR, ktorý po zobrazení dekrementuje adresu displeja (pozri vlastnosti

podprogramov monitora v [2]).

V ďalšom podobnom programe si ukážeme použitie programovej štruktúry - tabuľky. Zostavme program, ktorý bude snímať kód klávesu a na porte PlA bude rozsvecovať určitý počet LED diód, zodpovedajúci hodnote klávesu. Nech sa LED diódy rozsvecujú zľava. Ak stlačíme kláves s kódom väčším ako 08_H, nech sa rozsvieti všetkých osem LED diód. Vieme, že po použití podprogramu na testovanie klávesnice máme po stlačení klávesu v akumulátore jeho binárny kód. Túto hodnotu použijeme v programe ako relatívnu adresu prístupu v tabuľke. Hodnoty položiek v tabuľke budú vyjadrovať počet rozsvietených LED diód. Tabuľku uložíme od adresy 8000_H.

```
                    ;rozsvietenie daného počtu LED diód
                    ORG   8200H  ;začiatok programu
8200 3E80    CP15:  MVI   A,80H  ;riadiace slovo 8255 A-out
8202 D307           OUT   7      ;výstup do 8255 č. 1
8204 CD3D02  N1:    CALL  GETKY  ;vstup z klávesnice
8207 FE08           CPI   8      ;je kód väčší ako 08H ?
8209 DA0E82         JC    N2     ;ak nie, hľadaj v tabuľke
820C 3E08           MVI   A,8    ;inak nastav 08H do akumul.
820E 210080         LXI   H,TAB  ;nastav začiatok tabuľky
8211 0600           MVI   B,0    ;nuluj MSB dvojice BC
8213 4F             MOV   C,A    ;relatívna adresa v tabuľ-
                                 ;ke
8214 09             DAD   B      ;skutočná adresa položky
                                 ;v TAB
8215 7E             MOV   A,M    ;vyber položku z TAB
8216 D304           OUT   4      ;a zobraz ju na PlA
8218 C30482         JMP   N1     ;opakuj cyklus
                    ;nasleduje tabuľka položiek
                    ORG   8000H  ;adresa tabuľky
8000 0080C0E0 TAB:  DB    00H, 80H, 0C0H, 0E0H
```

```
8004 F0F8FCFE      DB    0F0H, 0F8H, 0FCH, 0FEH
8008 FF            DB    0FFH
```

Uvedenú štruktúru dát - tabuľku môžeme s výhodou použiť, ak potrebujeme pracovať vždy iba s určitým množstvom údajov, pričom máme k dispozícii relatívnu adresu položky. Skutočná adresa sa v cvičnom programe získa sčítaním relatívne adresy (je osembitová v dvojici BC) s adresou začiatku tabuľky, ktorú máme v dvojici HL.

Doposiaľ uvedené programy ukazujú použitie obvodu pri činnosti s celým byte údajov (číslicová forma informácie). Niekedy však, napríklad pri logickom riadení potrebujeme pracovať iba s jediným bitom. Zostavme program, ktorý bude cyklicky s periódou 1 s aktivizovať určité technologické zariadenia, vždy však iba jedno z ôsmich. Výstup aktivačných signálov budeme simulovať LED diódami v ŠMS. Prakticky sa činnosť programu prejaví pohybom svietiaceho bodu zľava doprava s rotáciou. Súčasne nech sa na displeji zobrazí hexadecimálna hodnota stavu výstupu P1A.

```
                    ;cyklický posuv svietiaceho bodu
                    ORG   8200H   ;začiatočná adresa programu
8200 3E80   CP16:   MVI   A,80H   ;riadiace slovo pre 8255
                                  ;A-out
8202 D307           OUT   7       ;výstup do 8255 č.1
8204 D304   N:      OUT   4       ;vysvieť jeden bit na P1A
8206 CD9502         CALL  DBYTE   ;zobraz výstup P1A na dis-
                                  ;pleji
8209 21E803         LXI   H,1000  ;nastav parameter oneskore-
                                  ;nia
820C F5             PUSH  PSW     ;schovaj výstup P1A
820D CD0080         CALL  DELYH   ;podprogram oneskorenia
                                  ;podľa HL
```



```
8210 F1             POP   PSW     ;obnov výstup P1A
8211 0F             RRC           ;posuň jednotku v akum. doprava
8212 C30482         JMP   N       ;opakuj cyklus
```

Na začiatku programu nastavíme P1A na výstup a ľavý bit nastavíme do log. 1. Použijeme na to priamo riadiace slovo 80_H. Zároveň sa táto hodnota zobrazí na displeji podprogramom DBYTE. Po zobrazení sa vykoná programové zdržanie 1 s a po ňom sa v cykle obsah akumulátora posúva cyklicky doprava. Na oneskorenie použijeme podprogram DELYH, uvedený v čl. 3. Tento vyžaduje v HL dvojici parameter oneskorenia, ktorý sa interpretuje ako čas v ms. Podprogram DELYH kazí obsah akumulátora, preto ho treba pred volaním uložiť a po návrate obnoviť.

Upozorňujeme, že pred spustením celého programu treba uložiť do pamäti aj podprogram DELYH.

Zostavme ďalej podobný program s tým rozdielom, že svietiaci bod sa bude pohybovať z jednej strany na druhú a späť. V tomto programe si ukážeme, ako možno obísť uchovávanie akumulátora. Využijeme pritom vlastnosť obvodu 8255, keď každý port vo výstupnom režime má záchytný register, ktorého obsah možno čítať inštrukciou IN (pozri čl.4).

```
                    ;kmitajúci svietiaci bod
                    ORG   8200H   ;začiatočná adresa
8200 3E80   CP17:   MVI   A,80H   ;riadiace slovo 8255 A-out
8202 D307           OUT   7       ;výstup do 8255 č. 1
8204 3E01   NO:     MVI   A,1     ;nastav posledný bit
8206 D304   N1:     OUT   4       ;vysvieť jeden bit
8208 213803         LXI   H,1000  ;nastav parameter onesko-
                                  ;renia
820B CD0080         CALL  DELYH   ;programové zdržanie podľa
                                  ;HL
```

```
820E DB04          IN   4       ;čítaj stav PlA
8210 07            RLC         ;cyklický posuv jednotky cez
                               ;CY
8211 D20682        JNC  N1      ;opakuj kým je jednotka
                               ;v akum.
8214 0F            RRC         ;ak vypadla do CY, vráť ju
                               ;naspäť do akumulátora
8215 D304     N2:  OUT  4       ;vysvieť jeden bit PlA
8217 21E803        LXI H,1000  ;nastav parameter oneskore-
                               ;nia
821A CD0080        CALL DELYH  ;programové zdržanie podľa
                               ;HL
821D DB04          IN   4      ;čítaj stav PlA
821F 0F            RRC         ;cyklický posuv cez CY
8220 D21582        JNC  N2     ;opakuj, kým je jedn. v akum.
8223 C30482        JMP  N0     ;opakuj cyklus
```

V programe obnovujeme hodnotu akumulátora prečítaním stavu portu PlA. V cykle N1 sa jednotka posúva inštrukciou RLC doľava a v cykle N2 sa inštrukciou RRC posúva zase doprava.

V poslednom príklade tohto článku si ukážeme, ako možno obvod 8255 využiť v logickom riadení na nastavovanie, nulovanie a zmenu na opačnú hodnotu daného bitu. Použijeme pritom časti cvičných programov z čl. 4. Jedná sa o cvičné programy CP8 až CP10. Zostavme program, ktorý bude na porte PlA meniť hodnotu bitu na opačnú ako práve je, pričom číslo bitu zadáme ručne z klávesnice. Preto budeme používať klávesy 0 až 7. Ak použijeme iný kláves, nech sa na displeji rozsvieti nápis "Err" a zastaví sa vykonávanie programu. Na zmenu bitu použijeme operáciu neekvivalencie (inštrukciu XRA). Preto si musíme pripraviť napr. v B registri vhodnú masku, ktorá bude mať v príslušnom bite jed-

notku. V ostatných bitoch sa akumulátor nezmení. Masku získame vstupom z klávesnice a použitím tabuľky.

```
                   ;zmena hodnoty zadaného bitu
                   ORG  8200H   ;ukladacia adresa
8200 3E80    CP18: MVI  A,80H   ;riadiace slovo pre
                               ;8255,A-out
8202 D307          OUT  7       ;výstup do 8255 č.1
8204 CD3D02  N:    CALL GETKY   ;vstup klávesu
8207 FE08          CPI  8       ;je kód menší ako 8 ?
8209 DA1082        JC   N1      ;ak áno, hľadaj v tabuľ-
                               ;ke
820C CDBC00        CALL ERROR   ;ak nie, hlás chybu
820F 76            HLT          ;zastav vykonávanie pro-
                               ;gramu
8210 210080  N1:   LXI  H,8000H ;nastav adresu tabuľky
8213 0600          MVI  B,0     ;nulový MSB relatívnej
                               ;adresy
8215 4F            MOV  C,A     ;daj relatívnu adresu
8216 09            DAD  B       ;vytvor skutočnú adresu
8217 46            MOV  B,M     ;vezmi položku z tabuľky
8218 DB04          IN   4       ;čítaj stav PlA
821A A8            XRA  B       ;zmeň príslušný bit na
                               ;opačný
821D C30482        JMP  N       ;a cyklicky opakuj
                   ;nasleduje tabuľka dát
                   ORG  8000H   ;umiestnenie tabuľky
8000 01020408 TAB: DB   01H,02H,04H,08H
8004 10204080      DB   10H,20H,40H,80H
```

V programe zámenou inštrukcie XRA B na adrese 821A za inštrukciou ANA B (ORA B) získame program, ktorý bude bit portu PlA zvolený z klávesnice nastavovať na log.0 (na log.1). Pri použití inštrukcie ANA treba zmeniť aj

položky v tabuľke. Tak napríklad maskou 11111101 docielime logickým súčinom vynulovanie bitu č.1.

10. Cvičné programy s reproduktorom

Zostava ŠMS VÚVT obsahuje súpravu cvičných prídavných zariadení, medzi nimi reproduktor využiteľný na generovanie tónov. Možno ho pripojiť na kontaktové polia vľavo na ŠMS.

Technické prostriedky ŠMS VÚVT umožňujú generovať tóny nasledovnými spôsobmi:

1/ obvodom univerzálnych portov 8255,
2/ analógovým prevodníkom,
3/ obvodom univerzálnych počítadiel 8253.

V ďalšom uvedieme programy na všetky vyššie uvedené možnosti generovania tónov.

Zásady používania univerzálnych portov sú vysvetlené v čl. 4. Na generovanie tónu použijeme port P1A obvodu 8255 č.1 vo výstupnom režime, ktorý je v ŠMS vybavený výkonovými výstupmi aj na pripojenie reproduktoru. Tón budeme generovať periodickou zmenou jedného bitu, kde pripojíme reproduktor. Medzi jednotlivými zmenami logických hodnôt zavedieme oneskorenie podľa želanej frekvencie. Takto generovaný signál bude mať obdĺžnikový tvar. Zostavíme program na generovanie tónu s frekvenciou 500 Hz.

```
                    ;generovanie tónu 500 Hz programom
                    ORG  8200H  ;začiatok
8200 3E80   CP19:   MVI  A,80H  ;riadiace slovo 8255,
                                ;A-out
8202 D307           OUT  7      ;výstup na obvod 8255 č.1
8204 DB04   N:      IN   4      ;čítaj stav na porte P1A
8206 EE01           XRI  01H    ;zmeň nultý bit na opačný
8208 D304           OUT  4      ;výstup zmeneného bitu
820A CD3602         CALL DELAY  ;čakaj  1 ms
820D C30482         JMP  N      ;opakuj cyklus
```


Port P1A obvodu 8255 č. 1 je naprogramovaný ako výstupný. Reproduktov pripojíme jedným kontaktom na kontaktové pole P1A na bit č. 0 a druhým kontaktom na GND alebo na +5V (vyššia intenzita tónu). Frekvencia generovaného signálu - 500 Hz je daná použitím podprogramu oneskorenia DELAY, ktorý oneskoruje 1 ms. Trvanie nuly a jednotky na P1A0 je 1 ms a preto perióda kmitov je 2 ms. Táto hodnota nie je presná, lebo sme zanedbali čas vykonávania inštrukcií v cykle programu. Iným oneskorením možno dosiahnuť iný tón (použitím podprogramu DELYA), prípadne inú striedu kmitov.

Zostavme program na generovanie signálu opäť so striedou asi 1:1, ale s meniacou sa frekvenciou. Zmenu frekvencie dosiahneme zmenou parametra oneskorenia pre podprogram DELYA.

```
                    ;programovo generovaný tón so zmenou
                    ;frekvencie
                    ORG  8200H  ;adresa začiatku
8200 3E80   CP20:   MVI  A,80H  ;riadiace slovo pre 8255
                                ;č.1
8202 D307           OUT  7      ;výstup do obvodu 8255
                                ;č.1
8204 06FF           MVI  B,0FFH ;nastav parameter oneskore-
                                ;nia
8206 DB04   N:      IN   4      ;čítaj stav portu P1A
8208 EE01           XRI  1      ;zmeň nultý bit P1A0 na
                                ;opačný
820A D304           OUT  4      ;výstup zmeneného bitu
```

```
820C 78        MOV  A,B      ;priprav parameter oneskore-
                             ;nia
820D CD3802    CALL DELYA    ;čakaj podľa hodnoty akumul.
8210 05        DCR  B        ;zníž parameter oneskorenia
8211 C30682    JMP  N        ;opakuj cyklus
```

Aj v tomto prípade sa tón generuje programovo cez port P1A. Procesor je pritom stále vyťažený a preto nemôže robiť nič iné. Perióda signálu sa dosahuje tiež programovým oneskorením, pritom však sa neustále zmenšuje. Okamžitá hodnota parametra oneskorenia sa udržuje v registri B, ktorý sa postupne dekrementuje. Dekrementovanie je cyklické postupne od hodnoty FF_H po 00_H.

Ďalším technickým prostriedkom umožňujúcim generovať tón, je analógový prevodník opísaný v čl. 7. Pri generovaní tónov ho využijeme v jeho najjednoduchšom režime - ako D/A prevodník. Jednotlivé byty do D/A prevodníka je možné generovať programom (algoritmicky vypočítavať), alebo je možné tieto hodnoty vyberať z tabuľky uloženej v pamäti ako konštanty. Pretože úlohou je ukázať princíp generovania tónov, obmedzíme sa na jednoduchší tabuľkový spôsob tvorenia údajov pre D/A prevodník. Ako vyplýva z čl. 7, analógový prevodník je možné využiť na generovanie analógových priebehov rôznych tvarov. Napríklad obdĺžnikový, pílový, trojuholníkový atď. Ukážeme si, ako tvar priebehu analógového signálu ovplyvňuje farbu generovaného tónu. Zostavíme program, ktorý bude generovať tóny rovnakej frekvencie, avšak s iným priebehom, pričom tvar možno zvoliť z klávesnice takto:

1 - sínusový priebeh
2 - trojuholníkový priebeh
3 - pílový priebeh
4 - obdĺžnikový priebeh.

Každý iný kláves spôsobí zobrazenie textu "Err" a zastaví program. Podľa obr. 7.1 treba nastaviť nultý bit P1CO=0 a na port P1B privádzať údaje z tabuľky. Stav P1CO=0 dosiahneme nastavením portu P1C do výstupného režimu (automaticky sa výstup portu vynuluje - pozri čl.4). Na generovanie každého tvaru signálu použijeme 32 vzoriek, ktoré sú uložené v tabuľke. Príslušné hodnoty položiek určíme dopredu "ručným" výpočtom. Presnosť tvaru generovaného priebehu je daná počtom vzoriek na periódu. Použitý počet 32 je pomerne málo, na cvičné programovanie však stačí. Reproduktor v tomto prípade zapojíme na ANALOG OUT a GND (alebo +5V).

```
                    ;generovanie tónov D/A prevodníkom
                    ORG  8200H    ;ukladacia adresa programu
8200 3E90   CP21:   MVI  A,90H    ;riadiace slovo 8255,
                                  ;B-out, C-out,
8202 D307           OUT  7        ;výstup na obvod 8255 č.1
8204 210080         LXI  H,TABL   ;začiatok tab.obdĺžnika
8207 220081         SHLD 8100H    ;schovaj aktuálnu adr.ta-
                                  ;buľky
820A 0620   N1:     MVI  B,32     ;počet vzoriek v tabuľke
820C 2A0081         LHLD 8100H    ;vezmi adresu tabuľky
820F 7E     NO:     MOV  A,M      ;vyber položku z tabuľky
8210 D305           OUT  5        ;a pošli do D/A
8212 23             INX  H        ;ďalšia položka tabuľky
8213 05             DCR  B        ;prečítali sa všetky po-
                                  ;ložky ?
8214 C20F82         JNZ  NO       ;ak nie, opakuj výber
                                  ;z tabuľky
8217 CD5702         CALL SCAN     ;inak testuj klávesnicu
821A D20A82         JNC  N1       ;nestlačený kláves-opakuj
                                  ;priebeh
821D 0604           MVI  B,4      ;štyri možnosti vstupu klá-
                                  ;vesu
```

```
821F 210080          LXI  H,TABL ;adresa prvej tabuľky
                                 ;obdĺž.)
8222 112000          LXI  D,32   ;vzdialenosť tabuliek
8225 220081  N2:     SHLD 8100H  ;odlož adresu aktuálnej
                                 ;tab.
8228 B8              CMP  B      ;je v B-reg. kód stlač.
                                 ;kláv.?
8229 CA0A82          JZ   N1     ;ak áno, opakuj výber
                                 ;z tab.
822C 19              DAD  D      ;ak nie skús adresu ďal-
                                 ;šej tab.
822D 05              DCR  B      ;zníž parameter cyklu
822E C22582          JNZ  N2     ;a opakuj cyklus
8231 CDBC00          CALL ERROR  ;keď to prišlo až sem,
                                 ;tak bol stlačený nesprá-
                                 ;ny kláves
8234 76              HLT         ;zastav vykonávanie pro-
                                 ;gramu
                     ;nasledujú tabuľky priebehov
                     ORG  8000H  ;ich začiatok
8000 FFFFFFFF TABL:  DB   OFFH,OFFH,OFFH,OFFH; obdĺžniky
8004 FFFFFFFF        DB   OFFH,OFFH,OFFH,OFFH
8008 FFFFFFFF        DB   OFFH,OFFH,OFFH,OFFH
800C FFFFFFFF        DB   OFFH,OFFH,OFFH,OFFH
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  ;píla
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   0AOH,0A8H,0B0H,0B8H
```



```
8038 C0C8D0D8        DB   0C0H,0C8H,0D0H,0D8H
803C E0E8F0F8        DB   0E0H,0E8H,0F0H,0F8H
8040 00102030 TAB3:  DB   0,10H,20H,30H ;trojuholník
8044 40506070        DB   40H,50H,60H,70H
8048 8090A0B0        DB   80H,90H,0AOH,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 8099B1C7 TAB:   DB   80H,99H,0B1H,0C7H ;sínusovka
8064 DBEBF6FD        DB   0DBH,0EBH,0F6H,0FDH
8068 FFFDF6EB        DB   0FFH,0FDH,0F6H,0EBH
806C DBC7B199        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
```

Na začiatku programu sa nastaví obvod 8255 č.1 tak, že port P1C je výstupný (automaticky P1C0=0) a port P1B je tiež výstupný. Program je ďalej zostavený tak, že ihneď po spustení se generuje obdĺžnikový priebeh. Preto po naprogramovaní obvodu 8255 sa dvojica HL nastaví na začiatok tabuľky tohto priebehu. Vzhľadom na to, že chceme generovať ten istý priebeh pokiaľ nebude z klávesnice zvolený iný, je potrebné odpamätať začiatok tabuľky práve generovaného priebehu. Program potom automaticky vyberá stále tú istú tabuľku, pokiaľ ju z klávesnice nezmeníme. Na odpamätanie sa používa slovo na adrese 8100_H. V tomto programe je výber z tabuľky sekvenčný, preto netreba osobitne vypočítavať výkonnú adresu položky, stačí inkrementovať predchádzajúcu adresu. Po každom generovaní celej periódy sa testuje, či je stlačený kláves. Ak je,

testuje sa jeho kód. Ak nie je ani jeden z 1 až 4, hlási sa chyba textom "Err". Inak sa podľa zadaného kódu zistí adresa žiadanej tabuľky. Využíva sa pritom skutočnosť, že tabuľky sú v pamäti vzdialené o 32 byte. Frekvencia generovaných tónov je daná časom výberu celej tabuľky. V tomto prípade čas výberu jednej hodnoty trvá asi 30 us. Cyklus na jednu periódu trvá 32 krát, čo je asi 1 ms. Potom nasleduje podprogram SCAN, ktorý trvá asi 80 us, takže generovaný signál má frekvenciu asi 900 Hz. Je to dané aj tým, že každá tabuľka obsahuje práve jednu periódu daného priebehu.

Vo všetkých štyroch prípadoch zostáva frekvencia generovaného tónu rovnaká, avšak vplyvom rozličného tvaru priebehu sa počuteľne mení jeho farba. Poznamenajme, že všetky priebehy pozostávajú z jednotlivých stupňov tak, ako vyzerá vzorkovaný priebeh.

Poslednou možnosťou na generovanie tónov je univerzálny obvod 8253. Jeho programovanie je opísané v časti 6. Z vlastností obvodu vyplýva, že na generovanie tónov je najvhodnejšie použiť tretí režim. V ňom sa generuje signál so striedou 1:1 pričom perióda signálu je daná zadanou predvoľbou a frekvenciou vstupných impulzov. V ŠMS VÚVT sú na vstup privedené hodiny procesora ø2, ktoré majú frekvenciu 2048 kHz. Takto po naprogramovaní obvodu na režim č.3 treba zadať vhodnú predvoľbu, ktorou sa vydelí vstupná frekvencia. Hodnotu predvoľby pre určitý tón (danú frekvenciu) určíme zo vzťahu:

$$\text{predvoľba} = 2048.10^3 \;/\; \text{daná frekvencia v Hz.}$$

Napríklad ak chceme generovať signál s frekvenciou 1 kHz, bude hodnota predvoľby 2048. Túto predvoľbu treba v programe zapísať vo vhodnom tvare. Okrem toho počítadlo môže počítať dvojkovo alebo dekadicky. Preto môžeme zaviesť

do počítadla dvojkovú aj dekadickú predvoľbu. Pre frekvenciu 1 kHz je dekadická predvoľba 2048 a dvojková 800_H.

Počítadlo T2 je určené k automatickému A/D prevodníku, preto odporúčame použiť na generovanie tónov počítadlo T0 alebo T1, ktorého výstup je na kontaktovom poli vľavo. Reproduktor zapojíme na kontakty výstupu príslušného počítadla a GND resp. +5V.

Zostavme program na generovanie tónu s frekvenciou 1 kHz.

```
                        ;generovanie  1 kHz počítadlom 8253,T0
                        ORG   8200H   ;ukladacia adresa
8200 3E37   CP22:       MVI   A,37H   ;režim 3 pre T0, 2 byte,
                                      ;BCD
8202 D317               OUT   17H     ;výstup na 8253
8204 3E48               MVI   A,48H   ;LSB predvoľby
8206 D314               OUT   14H     ;počítadlo T0
8208 3E20               MVI   A,20H   ;MSB predvoľby
820A D314               OUT   14H     ;počítadlo T0
820C E7                 RST   4       ;volanie monitora
```

V tomto prípade sa generuje frekvencia zvláštnymi technickými prostriedkami, preto nie je procesor zamestnaný a po vykonaní tejto inicializácie možno pokračovať v inom programe. V príklade je použité dekadické počítanie. Študujúcemu čitateľovi odporúčame zmeniť program na generovanie toho istého tónu s dvojkovým počítaním.

Tón sa generuje trvale až do ďalšieho zavedenia riadiaceho slova, keď obvod čaká na predvoľbu (kláves RESET je neúčinný, pretože obvod 8253 nemá resetovací vstup).

Vidno, že z doteraz uvedených spôsobov generovania tónov je použitie obvodu 8253 najvhodnejšie, pretože zme-

nu frekvencie možno dosiahnuť iba zmenou predvoľby počítadla a generovanie samotné nezamestnáva procesor. Zostavme program ktorý generuje tóny ako hudobný nástroj v závislosti od stlačeného klávesu. Použijeme stupnicu F-dur a program zostavíme tak, aby klávesnica (klávesy 0 až F) pokryla 2 oktávy. Klávesu 0 nech prislúcha tón f^1, klávesu 1 tón g^1 atď. až po kláves F, ktorému prislúcha tón g^3. Navyše nech sa na ľavých dvoch pozíciách displeja zobrazuje názov generovaného tónu písmenom. Na pravom z týchto dvoch displejov nech je indikované zníženie tónu o poltón. V prípade stupnice F-dur to bude tón hes, čo sa zobrazí: Hb. Predvoľby frekvencií ako aj konštanty pre zobrazovanie názvov tónov (obrazy znakov na displeji) budú uložené v tabuľke. Do tabuľky budeme pristupovať podľa stlačeného klávesu. Každému klávesu budú v tabuľke prislúchať 4 byte:

- dva byte predvoľby pre daný tón
- dva byte s obrazom názvu tónu (písmená na displeji).

Hodnoty predvolieb boli vypočítané už uvedeným spôsobom pre dvojkové počítanie. Tabuľka musí byť zostavená tak, že najprv sa do obvodu 8253 zadáva nižší byte. Tón sa bude generovať iba pri stlačenom klávese. Ak sa stlačí riadiaci kláves, nech sa generuje nepočuteľný tón (s vysokou frekvenciou).

```
                    ;hudobný nástroj
                    ORG  8200H   ;začiatok
8200 3E36    CP23:  MVI  A,36    ;režim 3, 2 byte, T0,
                                 ;dvojkovo
8202 D317           OUT  17H     ;výstup do obvodu 8253
8204 CD5702  N1:    CALL SCAN    ;je stlačený kláves?
8207 D20482         JNC  N1      ;ak nie, opakuj test
820A 87             ADD  A       ;inak vynásob kód klávesu
```



```
820B 87             ADD  A       ;štyrmi (krok položiek
                                 ;je 4)
820C 210080         LXI  H,TAB   ;začiatok tabuľky
820F 0600           MVI  B,0     ;nulový MSB relatívnej
                                 ;adresy
8211 4F             MOV  C,A     ;relatívna adresa tabuľ-
                                 ;ky
8212 09             DAD  B       ;skutočná adresa položky
8213 7E             MOV  A,M     ;vyber LSB predvoľby
8214 D314           OUT  14H     ;výstup LSB na 8253, T0
8216 23             INX  H       ;ďalšia časť položky
8217  7E            MOV  A,M     ;vyber MSB predvoľby
8218 D314           OUT  14H     ;výstup MSB do T0
821A 23             INX  H       ;ďalšia časť položky
821B 7E             MOV  A,M     ;vyber obraz názvu tónu
821C 32F883         STA  83F8H   ;zobraz názov tónu
821F 23             INX  H       ;ďalšia časť položky
8220 7E             MOV  A,M     ;vyber druhú časť obrazu
8221 32F983         STA  83F9H   ;zobraz druhú časť názvu
                                 ;tónu
8224 CD5702  N2:    CALL SCAN    ;je ešte stlačený kláves?
8227 DA2482         JC   N2      ;ak áno, opakuj test
822A 3E00           MVI  A,0     ;inak zmaž displej
822C 32F883         STA  83F8H   ;inak zmaž displej
822F 32F983         STA  83F9H   ;inak zmaž displej
8232 C30082         JMP  CP23    ;opakuj cyklus
                    ;nasleduje tabuľka dát
                    ORG  8000H   ;adresa tabuľky
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 9508767C     DB   95H,8,76H,7CH
802C A5073900     DB   0A5H,7,39H,0
8030 CF065E00     DB   0CFH,6,5EH,0
8034 11067900     DB   11H,6,79H,0
8038 BA057100     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
```

Najprv sa nastaví riadiaci register obvodu 8253 pre T0 na dvojkové počítanie v režime č. 3 so zadaním dvoch byte. Po nastavení riadiaceho slova počítadlo negeneruje žiaden tón (čaká na predvoľbu). Ďalej sa testuje klávesnica podprogramom SCAN. Ak bol stlačený kláves, vytvorí sa v akumulátore štvornásobok jeho kódu. Vzdialenosť položiek v pamäti je totiž 4 byte. Podľa tejto hodnoty sa vyberú z tabuľky postupne všetky štyri časti položky. Prvé dva byte sú predvoľbou a druhé dva sú kódy obrazov názvov tónov. Ďalším volaním podprogramu SCAN sa zisťuje, či je kláves ešte stále stlačený. Ak áno, tón sa generuje trvale ďalej. Ak nie, vynuluje sa displej a program sa opakuje od začiatku. Tým sa znovu nastaví riadiace slovo obvodu 8253, zastaví sa generovanie tónu a čaká sa na stlačenie ďalšieho klávesu.

Výška tónu je zrejme závislá iba od hodnôt v tabuľke a preto zmenou konštánt sa môže dosiahnuť generovanie iných tónov, stupnice atď. Ak by program samočinne vyberal kódy z tabuľky, možno dosiahnuť hudobný nástroj "naprogramovaný" na určitú melódiu.

11. Cvičné programy s elektromotorom

ŠMS VÚVT obsahuje cvičnú periférium malého elektromotora, ktorý je možné ovládať výkonovým výstupom (pozri čl. 8). Na hriadeli elektromotora (ďalej len motor) je optický kotúč na prerušovanie svetla LED diódy. Prerušované svetlo je indikované fototranzistorom, takže možno aj merať rýchlosť otáčania motora. Zostava LED dióda-fototranzistor je pevnou súčasťou cvičnej periférie motora.

Uvedieme cvičné príklady na riadenie rýchlosti motora spojitou a impulznou reguláciou a tiež na meranie otáčok motora.

Jednou z možností na riadenie motora je spojité riadenie, pri ktorom sa do motora zavedie jednosmerný prúd potrebnej hodnoty. V ŠMS to možno dosiahnuť tak, že budeme riadiť prúd pretekajúci LED diódou optočlena v zostave výkonového zosilňovača. Na riadenie prúdu LED diódy použijeme premenlivé napätie z výstupu D/A prevodníka, takže použijeme reťazec D/A prevodník - optočlen - výkonový zosilňovač - motor. Takto možno riadiť budenie motora programom cez D/A prevodník. Riešenie je výhodné v tom, že procesor je riadením zamestnaný len pri zmene budenia (pri rovnakom budení sa nevyžaduje činnosť procesora). Nevýhoda je v tom, že optočlen so zosilňovačom tvorí nelineárny prvok a preto je takéto riadenie nelineárne a citlivé len v malom rozsahu.

Kontaktové polia a motor je potrebné prepojiť takto:

- MOT RET - GND, motor (svorka OV)
- MOT DRV - motor (svorka +5V)
- MOT SUP+ - +5V
- MOT CTL+ - ANALOG OUT
- MOT CTL- - nepripojené

Program zostavíme tak, že podprogramom ENTBY zadáme jeden byte, ten použijeme na budenie motora a v cykle budeme čakať na zadanie ďalšieho byte:

```
                    ;cvičný program na riadenie motora
                    ;cez D/A
                    ORG  8200H
8200 3E90    CP24:  MVI  A,90H    ;inicializácia portov
                                  ;P1B-out,
8202 D307           OUT  7        ;P1C-out
8204 CD3603  CYKL:  CALL ENTBY    ;vstup byte
8207 7D             MOV  A,L      ;presuň daný byte do
                                  ;akumulátora
8208 CD0E82         CALL MOT      ;podprogram budenia moto-
                                  ;ra
820B C30482         JMP  CYKL     ;znova na vstup byte
                    ;nasleduje podprogram výstupu do D/A
820E D305    MOT:   OUT  5        ;výstup byte do D/A
8210 C9             RET           ;návrat z podprogramu
```

Na začiatku programu treba nastaviť P1B na výstupný režim (bude výstup do D/A). P1C nastavíme tiež na výstupný režim, čím zároveň privedieme na P1C1 log. 0, takže netreba kontakt MOT CTL- zvlášť pripojiť na GND. Podprogramom ENTBY zadáme byte (dva hexa-klávesy ukončené riadiacim klávesom), ktorý sa automaticky zobrazí na displeji. Zadaný byte presunieme do akumulátora a v podprograme MOT ho prenesieme do D/A prevodníka.

Rýchlosť motora je závislá na zadanom byte a možno ju regulovať v malom rozsahu zadaných byteov, pričom rýchlosť je závislá aj od polohy potenciometra ANALOG OUT (vhodné je nastaviť ho do strednej polohy). Oblasť citlivej regulácie treba pokusne zistiť zadávaním rôznych hodnôt (závisí od vlastností konkrétneho optočlena a nastavenia potenciometra).

V praxi sa často používa impulzné budenie motora, ktoré je energeticky úspornejšie ako spojitá regulácia. Uvedieme príklad programu na impulzné budenie motora. Využijeme pritom programové riadenie s tým, že riadiť budeme logickým signálom (nie analógovým ako v predošlom príklade) a reguláciu rýchlosti docielime vhodnou šírkou impulzov v pomere k medzere medzi nimi. Vytvoríne cyklus, v ktorom časť cyklu bude budiť motor plným napätím a zvyšná časť cyklu nebude budiť motor. Vzájomný pomer oboch časov budeme riadiť hodnotou stlačeného klávesu.

```
                    ;cvičný program na impulzné riadenie
                    ;motora
                    ORG  8200H
8200 3E92    CP25:  MVI  A,92H    ;inicializácia portov:
8202 D307           OUT  7        ;P1C - OUT
8204 3E02    CYKL:  MVI  A,00000010B ;nastav P1C1=0
8206 D307           OUT  7        ;nastav P1C1=0
8208 97             SUB  A        ;vynuluj akumulátor
8209 FE00    CP:    CPI  0        ;testuj parameter cyklu
820B F5             PUSH PSW      ;uschovaj parameter cyklu
820C C21382         JNZ  DELA     ;preskoč nastavenie P1C1
                                  ;P1C1=1
820F 3E03           MVI  A,00000011B ;nastav P1C1=1
```

```
8211 D307            OUT  7       ;nastav P1C1=1
8213 3E10      DELA: MVI  A,10H    ;nastav oneskorenie asi
                                   ;110 us
8215 CD3802          CALL DELYA    ;podprogram oneskorenia
8218 F1              POP  PSW      ;obnov parameter cyklu
8219 3C              INR  A        ;inkrementuj parameter
                                   ;cyklu
821A C20982          JNZ  CP       ;opakuj cyklus ak ne-
                                   ;skončil
821D CD5702          CALL SCAN     ;vstup klávesu
8220 D20482          JNC  CYKL     ;ak nebol stlačený - po-
                     ;kračuj novým cyklom so starou hod-
                     ;notou
8223 07              RLC           ;presuň spodné bity do
                                   ;vrchných
8224 07              RLC
8225 07              RLC
8226 07              RLC
8227 320A82          STA  CP+1     ;ulož novú rozhod. hod-
                                   ;notu
822A C30482          JMP  CYKL     ;pokračuj novým cyklom
```

V programe najprv nastavíme P1C na výstupný režim, pretože budeme riadiť prúd LED diódy optočlena bitom P1C1. Preto treba svorku MOT CTL+ pripojiť na +5V. Cyklus riadenia začína tým, že sa nastaví plné budenie motora (P1C1=0) a potom beží cyklus 256 krát, pričom parametrom cyklu je akumulátor. Motor sa odbudí na nulu v čase, keď sa zistí zhoda okamžitej hodnoty parametra cyklu a zadanej hodnoty z klávesnice (zvyšok cyklu dobehne s P1C1=1, motor je odbudený). Zhoda sa testuje inštrukciou CPI na adrese 8209_H. Trvanie celého cyklu je upravedné programovým zdržaním pomocou podprogramu DELYA. Po ukončení cyklu sa testuje stav

klávesnice. Ak nebola stlačená, začne sa nový cyklus s nezmenenou hodnotou rozhodovania na vypnutie budenia motora. Ak bol stlačený niektorý kláves, jeho kód (0÷0F) sa posunie o štyri miesta doľava, čím sa obsiahnu hodnoty 00 ÷ F0 (s krokom 10_H). Takto upravená hodnota z klávesnice sa uloží priamo do inštrukcie rozhodovania na adrese 8209_H (táto technika nie je možná, ak je program uložený v ROM pamäti).

Uvedený program na riadenie rýchlosti motora poskytuje citlivejšiu reguláciu, neustále však zamestnáva procesor. Hoci na zadanie z klávesnice sa používa len šestnásť hodnôt (klávesov), riadenie má lepšie vlastnosti ako pri riadení cez D/A prevodník a optočlen. Program je však dlhší.

Iný príklad programu na impulzmú reguláciu otáčok motora s využitím programovateľných časovačov je uvedený v ďalšom. Princíp riadenia motora je rovnaký ako v predošlom prípade, rozdiel je v tom, že čas budenia motora a čas medzery zaobstaráva programovateľný časovač T0, ktorý uplynutie oboch časov oznámi procesoru prerušením. Procesor je teda zamestnaný len nastavovaním bitu P1C1 a štartovaním činnosti časovača. Zvyšok (prevažný) je k dispozícii na inú činnosť procesora. Zvolíme si časovač T0, takže pri jeho činnosti vznikne prerušenie typu RST 5. V tomto prípade je najvhodnejším režimom nultý režim. Program má časť inicializačnú, pracovný cyklus, ktorým sa zadáva kód žiadanej rýchlosti otáčania a obslužnú rutinu prerušenia, ktorá zabezpečuje striedanie plného a nulového budenia. Pracovný cyklus neustále testuje stlačenie klávesu a môžeme ním simulovať hlavný program, ktorý je prerušovaný časovačom:

```
                    ;cvičný program na automatické
                    ;impulzné riadenie rýchlosti motora
                    ORG  8200H   ;ukladacia adresa pro-
                                 ;gramu
8200 3E20   CP26:   MVI  A,20H   ;nultý režim nultého
                                 ;počítadla
8202 D317           OUT  17H     ;zadávať sa bude len MSB
8204 3E92           MVI  A,92H   ;port C do výstupného
                                 ;režimu
8206 D307           OUT  7       ;P1C výstupný (MOT CTL-)
8208 D30F           OUT  0FH     ;P2C výstupný (povolenie
                                 ;prer.)
820A 3E01           MVI  A,00000001B ;povol prerušenie
                                     ;pre T0
820C D30F           OUT  0FH     ;a zároveň nuluj zách.
                                 ;obvod
820F EF             RST  5       ;prvý skok do podprogra-
                                 ;mu MOT
                    ;nasleduje cyklus snímania z kláves-
                    ;nice
820F CD5702 CYKL:   CALL SCAN    ;vstup z klávesnice
8212 D20F82         JNC  CYKL    ;vráť sa ak nebol stla-
                                 ;čený kláves
8215 07             RLC          ;posuv kódu klávesu
8216 07             RLC          ;o štyri bity doľava
8217 07             RLC
8218 07             RLC
8219 CD9502         CALL DBYTE   ;zobraz zadaný kód
821C 3C             INR  A       ;zvýš ho o jedna
821D 323082         STA  KOD+1   ;ulož do obslužnej rutiny
8220 C30F82         JMP  CYKL    ;opakuj cyklus
                    ;nasleduje obsluha prerušenia
                    ORG  8228H   ;adresa skoku po RST 5
```



```
8228 F5     MOT:    PUSH PSW     ;uschovaj register A
8229 DB06           IN   6       ;aké bolo naposledy P1C1?
822B EE02           XRI  2       ;zmeň bit P1C1 na opačný
822D D306           OUT  6       ;výstup zmeneného P1C1
822F 3E80   KOD:    MVI  A,80H   ;kód žiadanej rýchlosti
8231 CA3582         JZ   M1      ;preskoč ak je impulz
                                 ;budenia
8234 2F             CMA          ;urob doplnok=čas medze-
                                 ;ry
8235 D314   M1:     OUT  14H     ;naplnenie počítadla T0
8237 3E01           MVI  A,1     ;nuluj záchytný obvod
8239 D30F           OUT  0FH     ;prerušenia od T0
823B F1             POP  PSW     ;obnov akumulátor
823C FB             EI           ;obnov stav EI po preru-
                                 ;šení
823D C9             RET          ;návrat z prerušenia
```

Program začína inicializáciou, v ktorej sa nastaví počítadlo T0 do nultého režimu (programové nastavenie časového intervalu s prerušením) a porty P1C a P2C do výstupného režimu (P1C1 na riadenie motora a P2C0 na povolenie prerušenia od T0). Do počítadla budeme zadávať len vyšší byte (stačí aj takáto presnosť nastavenia časového intervalu). Výstupom byte 00000001 na adresu $0F_H$ sa zároveň povolí prerušenie od T0 a tiež sa vynuluje záchytný preklápací obvod prerušenia (pozri čl.5). Štartovanie činnosti T0 sa vykoná až v obslužnej rutine prerušenia, preto ju prvý krát treba umele vyvolať cez RST 5. Vtedy sa ešte použije začiatočný kód rýchlosti 80_H, ktorý je programovaný v inštrukcii MVI A,80H na adrese $822F_H$. V hlavnom programe "CYKL" sa sníma znak z klávesnice (kód 00 ÷ 0F), ktorý sa hodnotovo expanduje posunutím na kódy 00 ÷ F0. Takto upravená hodnota sa zobrazí na displeji a vloží sa do inštrukcie, ktorá obsahuje kód žiadanej rýchlosti. Ukladá

sa však inkrementovaná hodnota, aby sa pri stlačení klávesu nula dosiahlo minimálne budenie (časovač pri zadaní nuly negeneruje najkratší časový interval, ale pri zadaní najmenšieho nenulového čísla). Obslužný podprogram prerušenia leží na adrese 8228_H v súlade s poznámkami v čl. 5. V ňom sa najprv uschová hodnota akumulátora, pretože sa s ňou manipuluje a po návrate do hlavného programu by to vadilo. Inštrukciou IN 6 sa získa naposledy vyslaný byte na port P1C, ktorého prvý bit sa zmení na opačný nonekvivalenciou XRI 2 a opäť sa vyšle na ten istý port (motor sa rozbehne resp. zastaví). Kód žiadanej rýchlosti sa použije na nastavenie časovača T0 (adr. 14_H). Ak sa jedná o pracovnú časť cyklu (motor je naplno budený), použije sa priama hodnota. Ak sa jedná o medzeru (motor je odbudený), použije sa doplnok k tejto hodnote, ktorý získame inštrukciou CMA. Na rozhodovanie sa využije výsledok inštrukcie XRI 2 (príznak Z=0 značí P1C1=0, t.j. motor je budený). Zabezpečíme tým konštantné trvanie celého cyklu s premenlivým pomerom času budenia a medzery. Na konci prerušovacieho podprogramu sa vynuluje záchytný obvod prerušenia od T0, obnoví sa obsah akumulátora, obnoví sa stav EI, čím sa nastavia podmienky pre výskyt ďalšieho prerušenia a návrat do prerušeného programu.

Program je ukážkou obsluhy technických prostriedkov, ktoré do istej miery pracujú samostatne a vyžadujú za istých okolností obsluhu procesora (podľa daného algoritmu), pričom si pozornosť vynucujú prerušením. Poznamenajme, že hlavný program nesmie na dlhší čas zotrvať v stave DI (v našom prípade sa DI v hlavnom programe nevyskytuje), pretože by sa tým nedodržali podmienky riadenia motora (predĺžili by sa tým časy). Zapojenie kontaktov (prepojení) je rovnaké ako v predošlom cvičnom programe (MOT CTL+ - +5V).

Cvičná periféria malého elektromotora obsahuje aj zostavu LED-dióda - fototranzistor. Pomocou nich možno merať rýchlosť otáčania elektromotora, pretože na hriadeli je priehľadný kotúč so 16 nepriehľadnými výsekmi, ktoré pri otáčaní prerušujú tok svetla na fototranzistor. Zostavíme cvičný program na meranie otáčok motora a zobrazovanie hodnoty na displeji. Na riadenie rýchlosti motora použijeme spojité riadenie cez D/A prevodník, pričom žiadaná hodnota rýchlosti sa bude vkladať cez klávesnicu. Program nebude rýchlosť regulovať, t.j. nebude uzavretá slučka medzi meradlom rýchlosti a D/A prevodníkom.

Hlavný program bude pozostávať z volaní modulov - podprogramov na inicializáciu, vstup dát z klávesnice, meranie otáčok a zobrazovanie a programové zdržanie (aby sa údaj na displeji nemenil príliš rýchlo):

```
                        ;hlavný program (meranie otáčok mo-
                        ;tora)
                        ORG  8200H     ;ukladacia adresa hlav.
                                       ;prog.
8200 CD0080   CP27:     CALL INIC      ;volanie inicializácie
8203 CD5702   CYKL:     CALL SCAN      ;je stlačený kláves?
8206 D20F82             JNC  C1        ;ak nie, preskoč vstup
                                       ;dát
8209 CD3603             CALL ENTBY     ;vstup byte pre D/A
820C CD1380             CALL DA        ;modul riadenia výstupu
                                       ;na D/A
820F CD1780   C1:       CALL MER       ;modul merania rýchlosti
                                       ;a zobr.
8212 CD4A80             CALL ZDRZ      ;modul zdržania asi 1/4
                                       ;sekundy
8215 C30382             JMP  CYKL      ;opakuj cyklus
```

V hlavnom programe sa najprv volá podprogram inicializácie portov, počítadla a nastavenie adresy pri prerušení. Potom sa testuje klávesnica. Ak je kláves stlačený, sníma sa jeden byte (pomocou ENTBY) a zavedie sa ako nová hodnota do D/A prevodníka podprogramom DA. Inak sa pokračuje modulom na meranie rýchlosti MER, ktorý aj zobrazuje nameranú hodnotu. Podprogram ZDRZ spomaľuje celú slučku asi o 0,25 s , aby bol údaj na displeji čitateľný.

Podprogram INIC nastavuje potrebné začiatočné hodnoty programovateľných obvodov. Treba vykonať tieto nastavenia:

- port P1B bude výstupný pre D/A prevodník
- port P1C bude výstupný (P1C1=MOT CTL-, má byť v LOG.0)
- port P2B bude vstupný, merací fototranzistor pripojíme na vonkajší vstup EXT 4 (t.j. P2B6)
- port P2C bude výstupný, povolíme prerušenie pre počítadlo T0
- počítadlo T0 bude merať časový interval, využijeme režim nula
- na adresu $83EC_H$ uložíme adresu obslužnej rutiny prerušenia (počítadlo T0 bude generovať prerušenie typu RST 5)

```
                        ORG  8000H  ;začiatok podprogramov
8000 3E90    INIC:      MVI  A,90H  ;nastavenie B-out, C-out
8002 D307               OUT  7      ;pre porty P1B a P1C
8004 3E92               MVI  A,92H  ;nastavenie B-in, C-out
8006 D30F               OUT  0FH    ;pre porty P2B a P2C
8008 3E20               MVI  A,20H  ;nultý režim pre T0
800A D317               OUT  17H    ;výstup riadiaceho slova
                                    ;pre 8253
800C 213D80             LXI  H,PRER ;adresa obsluhy preruše-
                                    ;nia
800F 22EC83             SHLD 83ECH  ;uloženie adresy po RST 5
8012 C9                 RET         ;návrat z podprogramu
```


Podprogram DA odovzdáva byte (z podprogramu ENTBY je v registri L) do D/A prevodníka:

```
8013 7D      DA:        MOV  A,L    ;presuň byte do akumu-
                                    ;látora
8014 D305               OUT  5      ;výstup byte do D/A pre-
                                    ;vodníka
8016 C9                 RET         ;návrat z podprogramu
```

Poznamenajme, že P1C0=log. 0 (automaticky po nastavení P1C do výstupného režimu), takže D/A prevodník dostáva dáta z portu P1B a nie z vlastného počítadla. Podobne je automaticky P1C1=log. 0, takže MOT CTL- možno považovať za uzemnené. Takto stačí prepojiť vodičom svorky ANALOG OUT - MOT CTL+ . Ostatné svorky treba pripojiť ako v cvičnom programe CP24.

Meranie rýchlosti vykonáme tak, že počítadlom T0 odmeriame určitý časový úsek a programovo spočítame, koľko impulzov zaregistroval fototranzistor na kotúči motora. Kotúč má 16 tmavých segmentov a 16 priehľadných segmentov, takže na jednu otáčku zaregistrujeme 32 prechodov fototranzistora do opačného logického stavu. Zvoľme na meranie časový úsek 1/32 sekundy (taký dlhý čas počítadlom T0 ešte možno dosiahnuť). Potom zistený počet zmien fototranzistora udáva priamo počet otáčok za sekundu. Počítať a zobrazovať nameraný údaj budeme dekadicky. Stav fototranzistora budeme zisťovať programovo cez svorku EXT 4. Preto treba svorky perifétie motora prepojiť takto:

- A - +5V
- K - GND
- E - GND
- KOL - EXT 4 a zároveň cez odpor asi 10 kOhm na +5V

Časový úsek 1/32 sekundy tvorí 31,25ms, čo zodpovedá slovu $FA00_H$ pre odmeriavacie počítadlo. Nižší byte je nulový,

preto stačí zadať iba vyšší byte:

```
8017 F5      MER:   PUSH  PSW    ;schovaj PSW
8018 3EFA           MVI   A,0FAH ;konštanta pre počítadlo
801A D314           OUT   14H    ;výstup do T0
801C 3E01           MVI   A,1    ;povolenie prerušenia
                                 ;len pre T0
801E D30F           OUT   0FH    ;výstup masky prerušenia
                                 ;do P2C
8020 FB             EI           ;nastav EI procesora
8021 97             SUB   A      ;nuluj akumulátor
8022 323C80  ZP:    STA   BYTE   ;zapíš doterajší počet
                                 ;impulzov
8025 DB0D    M1:    IN    0DH    ;vstup z P2B (P2B6=EXT 4)
8027 E640           ANI   40H    ;zisti P2B6
8029 CA2580         JZ    M1     ;vráť sa ak je EXT 4 = 0
802C DB0D    M2:    IN    0DH    ;vstup z P2B
802E E640           ANI   40H    ;zisti P2B6
8030 C22C80         JNZ   M2     ;vráť sa ak je EXT 4 = 1
8033 3A3C80         LDA   BYTE   ;doterajší počet impulzov
8036 C602           ADI   2      ;pripočítaj obidva stavy
                                 ;(0,1)
8038 27             DAA          ;dekadická úprava obsahu
                                 ;A-reg.
8039 C32280         JMP   ZP     ;opakuj do prerušenia od
                                 ;T0
803C 00      BYTE:  DB    0      ;napočítané impulzy
803D 11FB83  PRER:  LXI   D,83FBH;štvrtý displej zľava
8040 3A3C80         LDA   BYTE   ;vezmi napočítané impulzy
8043 CD9802         CALL  DBY2   ;a zobraz ich na displeji
8046 33             INX   SP     ;vráť ukazovateľ zásobní-
                                 ;ka
8047 33             INX   SP     ;vráť ukazovateľ zásobní-
                                 ;ka
```



```
8048 F1             POP   PSW    ;obnov akumulátor
8049 C9             RET          ;návrat z podprogramu MER
```

Na začiatku podprogramu MER sa zadá konštanta časového intervalu 31,25 ms do počítadla T0 a povolí sa jeho prerušenie. Pamäťová bunka BYTE slúži na uloženie napočítaných impulzov a preto treba na začiatku merania obsah vynulovať. Potom nasledujú dva cykly M1 a M2, z ktorých jeden registruje stav odkrytého fototranzistora a druhý stav zakrytého fototranzistora. Po ich uplynutí treba k doterajšiemu počtu pripočítať dvojku (stav odkrytý, stav zakrytý) a upraviť na dekadický tvar. Tento cyklus sa opakuje dovtedy, kým neuplynie daný časový interval, čo sa prejaví prerušením od T0.

V tomto programe sa vyskytuje jediné prerušenie (od T0), ktoré sa obslúži podprogramom PRER. Jeho úlohou je vybrať napočítané impulzy a zobraziť ich v ľavej časti displeja. Podprogram je zaujímavý tým, že z neho sa riadenie nevracia do prerušeného modulu MER, ale priamo do hlavného programu. Pretože po RST 5 ostala v zásobníku návratová adresa do modulu MER, treba ukazovateľ zásobníka umele upraviť (zvýšiť) inštrukciami INX SP. Odporúčame študujúcemu čitateľovi premyslieť odôvodnenosť postupnosti inštrukcií na adresách $8046_H \div 8049_H$ (obsah zásobníka v okamihu po prerušení).

Podprogram ZDRZ má za úlohu vytvoriť programové zdržanie asi 0,25 s, čo dosiahneme 256-násobným volaním podprogramu DELAY:

```
804A 0600    ZDRZ:  MVI   B,0    ;nuluj register  B
804C CD3602  Z1:    CALL  DELAY  ;zdržanie  1 ms
804F 05             DCR   B      ;dekrementuj parameter
                                 ;cyklu
```

```
8050 C24C80          JNZ  Z1     ;vráť sa ak neskončil
                                 ;cyklus
8053 C9              RET         ;návrat z podprogramu
```

Uvedený program má z hľadiska riadenia rýchlosti motora podobné vlastnosti ako cvičný program CP24, preto treba skusmo nájsť oblasť citlivého riadenia (závisí od polohy potenciometra ANALOG OUT, je vhodné dať ho do strednej polohy). Diplej v ľavej časti zobrazuje rýchlosť otáčania v ot/s v dekadickej forme. Poznamenajme, že údaj nad 100 ot/s sa zobrazuje nesprávne (zodpovedá nad 6000 ot/min). V pravej časti displeja sa zobrazuje zadaný kód z klávesnice (zabezpečí podprogram ENTBY). Zadávanie hodnoty je dané vlastnosťami podprogramu ENTBY - zadať dva hexadecimálne klávesy ukončené riadiacim klávesom.

Meranie v podprograme MER trvá asi 32 ms, čo je relatívne dlhá doba oproti časovej konštante rozbiehania motora. Preto tento spôsob merania otáčok nie je vhodný na vytvorenie uzavretej slučky regulovania rýchlosti motora. Vzhľadom na spôsob inkrementácie v podprograme, sú namerané hodnoty párne čísla. Odporúčame čitateľovi navrhnúť, ako odstrániť tento nedostatok (akú inú nevýhodu bude mať navrhnuté riešenie?). Nulová rýchlosť otáčania sa zobrazí ako nula na displeji. Čas merania nie je závislý od rýchlosti motora.

V nasledujúcom cvičnom programe uvedieme ukážku zaradenia mikroprocesora do regulačnej slučky riadiaceho zariadenia. Budeme regulovať rýchlosť motora podľa žiadanej hodnoty z klávesnice. Program bude zostavený tak, že bude merať otáčky motora a porovnaním so žiadanou hodnotou sa bude generovať riadiaca veličina pre motor. Otáčky motora budeme merať tak, že zistíme čas trvania jedného segmentu kotúča na hriadeli motora. To je metóda oveľa rýchlejšia

ako v predchádzajúcom príklade, výsledkom je však čas trvania jednej otáčky, nie počet otáčok za sekundu. Motor budeme riadiť impulzným spôsobom (plne budený, nebudený). Na riadenie využijeme D/A prevodník, hoci impulzne možno riadiť aj bitom P1C1. Predpokladáme však, že čitateľ bude mať záujem laborovať s programom zmenením algoritmu riadenia a prípadne bude potrebovať spojitý výstup budenia motora. Pri laborovaní stačí zmeniť podprogram REGU. Prepojenie kontaktových polí preto ostáva rovnaké ako v predošlom cvičnom programe. Hlavný program je podobný ako v predchádzajúcom príklade:

```
                     ;hlavný program regulátora otáčok
                     ;motora
                     ORG  8200H  ;ukladacia adresa pro-
                                 ;gramu
8200 CD0080  CP28:   CALL INIC   ;inicializačný modul
8203 CD5702  CYKL:   CALL SCAN   ;je stlačený kláves?
8206 D21382          JNC  C1     ;ak nie, choď na meranie
8209 97              SUB  A      ;daj nulový byte do
820A D305            OUT  5      ;D/A prevodníka
820C CD3603          CALL ENTBY  ;vstup byte (žiadaná
                                 ;hodnota)
820F 7D              MOV  A,L    ;presuň byte do akumulá-
                                 ;tora
8210 327780          STA  ZIHO+1 ;a zapíš do regulačného
                                 ;modulu
8213 CDOF80  C1:     CALL MER    ;modul na meranie rých-
                                 ;losti
8216 CD7380          CALL REGU   ;modul regulácie rýchlos-
                                 ;ti
8219 CD5580          CALL ZOBR   ;modul zobrazovania úda-
                                 ;ja
821C C30382          JMP CYKL    ;opakuj cyklus
```

Na začiatku programu sa iniciujú porty a nastaví sa adresa prerušovacej rutiny. Potom sa testuje či je stlačený kláves. Ak nie je, pokračuje sa meraním rýchlosti, reguláciou otáčok a zobrazením rýchlosti. Ak je stlačený, zastaví sa motor a čaká sa na vstup byte pomocou ENTBY. Zadaný byte sa prenesie priamo do modulu regulácie otáčok (= žiadaná hodnota rýchlosti). Potom nasleduje modul merania rýchlosti, ktorý nechá výsledok v pamäťovej bunke na adrese SKHO (skutočná hodnota rýchlosti). Regulačný modul na základe žiadanej a skutočnej hodnoty rýchlosti generuje riadiacu veličinu pre motor. Modul REGU má byť zaradený hneď za modulom MER, aby nenastalo oneskorenie medzi meraním a riadením rýchlosti. Modul ZOBR zobrazuje nameraný údaj rýchlosti. Pracuje tak, že zobrazuje každú 256. hodnotu, aby boli údaje vizuálne postihnuteľné (aby sa nemenili príliš rýchlo). V tomto prípade nemožno kvôli zobrazovaniu spomaliť cyklus hlavného programu tak ako v predošlom programe, lebo by sa tým znemožnilo riadenie podľa žiadanej hodnoty (riadiaci algoritmus musí byť rýchlejší ako časová konštanta motora).

Inicializačný modul musí pripraviť stav portov P1B na výstupný (výstup do D/A prevodníka), P1C na výstupný (P1C0=0 na nastavenie spôsobu práce D/A prevodníka, P1C1=0 na nastavenie MOT CTL- na log. 0), P2B na vstupný (P2B6=EXT 4 na registráciu stavov fototranzistora na meranie rýchlosti) a PC2 na výstupný režim (P2C2=1 na povolenie prerušenia počítadla T2). Okrem toho sa na adresu $83EA_H$ uloží adresa prerušovacieho podprogramu:

```
                    ORG  8000H  ;podprogramy uložíme od
                                ; 8000H
8000 3E90   INIC:   MVI  A,90H  ;riadiace slovo B-out,
                                ;C-out
```



```
8002 D307           OUT  7      ;pre 8255 č. 1
8004 3E92           MVI  A,92H  ;riadiace slovo B-in,
                                ;C-out
8006 D30F           OUT  0FH    ;pre 8255 č.2
8008 214E80         LXI  H,PRER ;adresa prerušovacej ru-
                                ;tiny
800B 22EA83         SHLD 83EAH  ;po RST 6 - ulož na
                                ;83EAH
800E C9             RET         ;návrat z podprogramu
```

V podprograme MER sa využíva počítadlo T2 a jeho riadiace slovo sa zadáva v module MER, preto nemusí byť v module INIC. Počítadlo T2 generuje prerušenie typu RST 6, čomu zodpovedá adresa $83EA_H$ (modul INIC).

Modul MER má za úlohu odmerať čas trvania jedného priehľadného výseku na kotúči motora. Čas zistíme odčítaním obsahu počítadla T2, ktoré je priebežne dekrementované hodinovým signálom procesora $\phi 2$.

```
800F 3EB0   MER:    MVI  A,0B0H ;režim č. 0 pre T2
8011 D317           OUT  17H    ;výstup riadiaceho slova
                                ; 8253
8013 3E04           MVI  A,4    ;povolenie prerušenia
                                ;len pre T2
8015 D30E           OUT  0EH    ;výstup na P2C
8017 FB             EI          ;nastav EI procesora
8018 CD6C80         CALL STRT   ;štartovanie počítania T2
801B DB0D   M1:     IN   0DH    ;vstup z P2B
801D E640           ANI  40H    ;zisti P2B6 = EXT 4
801F CA1B80         JZ   M1     ;čakaj ak je priehľadný
                                ;výsek
8022 DB0D   M2:     IN   0DH    ;vstup z P2B
8024 E640           ANI  40H    ;zisti P2B6 = EXT 4
```

```
8026 C22280          JNZ  M2       ;čakaj ak je tmavý výsek
8029 CD6C80          CALL STRT     ;štartuj časové počítad-
                                   ;lo T2
802C DB0D      M3:   IN   0DH      ;vstup z P2B
802E E640            ANI  40H      ;zisti P2B6 = EXT 4
8030 CA2C80          JZ   M3       ;čakaj ak je priehľadný
                                   ;výsek
8033 3E80            MVI  A,80H    ;vzorkuj počítadlo T2
8035 D317            OUT  17H      ;výstup do 8253
8037 DB16            IN   16H      ;vstup LSB z T2
8039 2F              CMA           ;vytvor doplnok k LSB
803A 6F              MOV  L,A      ;daj LSB do L-registra
803B DB16            IN   16H      ;vstup MSB z T2
803D 2F              CMA           ;vytvor doplnok k MSB
803E 67              MOV  H,A      ;daj MSB do H-registra
803F 0606            MVI  B,6      ;delenie HL číslom 64
8041 CD2E02    POS:  CALL SHLRZ    ;delenie HL číslom 64
8044 05              DCR  B        ;delenie HL číslom 64
8045 C24180          JNZ  POS      ;delenie HL číslom 64
8048 7D              MOV  A,L      ;významný byte
8049 328580    ODL:  STA  SKHO     ;ulož nameranú rýchlosť
804C F3              DI            ;už netreba prerušenie
                                   ;od T2
804D C9              RET           ;návrat z modulu merania
                     ;nasleduje program po prerušení od T2
804E 3EFF      PRER: MVI  A,0FFH   ;kód nulovej rýchlosti
8050 33              INX  SP       ;nebude RET do bodu pre-
                                   ;rušenia
8051 33              INX  SP       ;nebude RET do bodu pre-
                                   ;rušenia
8052 C34980          JMP  ODL      ;ulož kód nulovej rých-
                                   ;losti
```


Počítadlo T2 nemá záchytný obvod prerušenia, čo je vhodné v tomto prípade (výstup T2 je v uvedenom režime na trvalej hodnote log. 1 alebo 0). Na meranie časového intervalu programom je vhodný režim č. 0, preto ho treba na začiatku podprogramu MER nastaviť. Čas trvania jedného segmentu kotúča odmeriame tak, že na začiatku segmentu nastavíme v počítadle hodnotu $FFFF_H$ a na konci segmentu zistíme obsah počítadla, ktoré je priebežne dekrementované hodinami ∅2. Ak sa však kotúč netočí, treba po istom čase meranie prerušiť a konštatovať nulovú rýchlosť. To dosiahneme tak, že ak vznikne prerušenie od T2 (teda od začiatku merania uplynulo už viac ako 32 ms), ukončíme meranie a nastavíme kód nulovej rýchlosti. Toto opatrenie je dané dĺžkou počítadla (16 bitov) a frekvenciou hodín ∅2 (2048 kHz), preto jeho cyklus trvá 32 ms. Praktický účinok takéhoto riešenia je ten, že všetky rýchlosti motora menšie ako 1 ot/s sa vyhodnotia ako nulové (32 ms počítadla T2 krát 32 segmentov na kotúči). V programe MER preto treba pripraviť prerušovací systém a procesor na prerušenie od T2 (hoci sa využíva len ak kotúč stojí alebo sa točí veľmi pomaly). Potom nasleduje registrácia začiatku priehľadného segmentu kotúča. Na to slúžia synchronizačné cykly M1 a M2. Po nich treba naštartovať počítadlo T2 a potom registrovať koniec priehľadného úseku v cykle M3. Ak kotúč stojí, program by uviazol v niektorom cykle M1 alebo M2. Preto treba naštartovať počítadlo T2 aj pred cyklom M1. Novým štartovaním na adrese 8029_H sa začne dekrementácia znova od obsahu $FFFF_H$. Ak kotúč stojí, počítadlo T2 po čase 32 ms preruší a vyjde sa zo slučky M1 alebo M2. Po skončení meraného priehľadného segmentu (po cykle M3) treba zistiť okamžitý stav počítadla T2, čo možno dosiahnuť povelom na vzorkovanie jeho obsahu a čítaním oboch byteov. Pretože počítadlo odpočítava, treba urobiť rozdiel začiatočnej hodnoty počí-

tania (samé jedničky) a konečnej hodnoty. Vzhľadom na hodnotu prvého operandu netreba odpočítavať, stačí vytvoriť jednotkový doplnok konečnej hodnoty inštrukciou CMA, ktorou sa každý bit zmení na opačný. Týmto zásahom získame v dvojici HL napočítaný čas jedného segmentu kotúča. Jedná sa o počet polmikrosekúnd na segment kotúča (perióda ø2 je 0,488 us). Z hľadiska používateľa je vhodnejšia forma údaju rýchlosti napríklad počet milisekúnd na otáčku kotúča. Na to by bolo treba násobenie (delenie) určitým prepočítavacím koeficientom. Jeho hodnota je blízka hodnote 1/64 (32 segmentov na kotúči krát počet mikrosekúnd na periódu ø2 deleno počet mikrosekúnd v jednej milisekunde) a preto násobenie presným číslom nahradíme (s dosť veľkou presnosťou) posuvom o 6 bitov doprava (t.j. delením číslom 64). Na posúvanie využijeme podprogram SHLRZ. Na vyjadrenie rýchlosti potom stačí nižší byte výsledku (pre reálne rýchlosti od 4 ot/s = kód FF_H až do napr. 6000 ot/s = kód $0A_H$) a ten sa uloží do pamäti na adresu SKHO (skutočná hodnota rýchlosti). Po nameraní rýchlosti (ak kotúč nestál) treba zabrániť prerušeniu od T2, lebo toto by nastalo v neskoršej dobe po dopočítaní T2 do nulového obsahu (meranie už bolo dávno vykonané). Ak kotúč stál, vznikne po 32 ms od naštartovania T2 prerušenie, takže riadenie prejde do programu PRER. V takomto prípade treba nastaviť kód najmenšej (nulovej) rýchlosti a prejsť na jeho uloženie na SKHO. Z programu PRER sa netreba vrátiť do bodu prerušenia v podprograme MER a preto treba upraviť (zvýšiť) ukazovateľ zásobníka (prerušením sa do neho uložila návratová adresa). Inštrukciou RET na adrese $804D_H$ nastane návrat do hlavného programu.

Je zrejmé, že podprogram MER zisťuje prevrátenú hodnotu rýchlosti otáčania. Čas merania je pritom závislý na rýchlosti otáčania a najdlhší je, ak sa kotúč netočí

(32 ms). Podmienkou presného merania je, aby boli segmenty na kódovom kotúči zhodné s presne vyjadrenými (ostrými) okrajmi. Aj v tomto prípade treba zaradiť medzi svorku EXT 4 a +5V odpor asi 10 kOhm, aby Schmittov obvod na vstupe EXT 4 pracoval spoľahlivo. Podprogram STRT použitý v podprograme MER je uložený na adrese $806C_H$ a má za úlohu naplniť obvody počítadla T2 hodnotou $FFFF_H$.

```
806C 3EFF     STRT:  MVI  A,0FFH  ;jednotková slabika
806E D316            OUT  16H     ;jej výstup do T2 ako
                                  ;LSB
8070 D316            OUT  16H     ;jej výstup do T2 ako
                                  ;MSB
8072 C9              RET          ;návrat zo STRT
```

Podprogram na reguláciu REGU má za úlohu porovnať žiadanú hodnotu rýchlosti motora so skutočnou hodnotou a podľa výsledku porovnania budiť motor. Rozhodovanie je dané tým, že sa porovnávajú <u>periódy</u> otáčania motora (a nie ot/s). Motor budeme riadiť (budiť) impulzne cez D/A prevodník jednoduchým algoritmom tak, že ak je skutočná perióda otáčania menšia ako žiadaná, treba motor úplne odbudiť a naopak. Ak sa zhoduje skutočná a žiadaná hodnota, nemá sa meniť riadiaca veličina.

```
8073 3A8580   REGU:  LDA  SKHO    ;vezmi údaj skutočnej
                                  ;hodnoty
8076 FE00     ZIHO:  CPI  0       ;porovnaj so žiadanou
                                  ;hodnotou
8078 C8              RZ           ;návrat ak je zhoda
8079 D28080          JNC  BUD     ;skoč ak sú nízke otáčky
807C 97              SUB  A       ;ak sú vysoké, nulu aku-
                                  ;mulátor
```

```
807D C38280          JMP  VON     ;a vyšli údaj von do D/A
8080 3EFF     BUD:   MVI  A,0FFH  ;nastav plné budenie
8082 D305     VON:   OUT  5       ;pošli byte do D/A
8084 C9              RET          ;návrat z podprogramu
8085 00       SKHO:  DB   0       ;uložená skutočná hodno-
                                   ;ta
```

V podprograme REGU sa žiadaná hodnota periódy otáčania ukladá z hlavného programu priamo do inštrukcie porovnania CPI na adrese 8076_H.

Podprogram zobrazovania nameranej hodnoty zobrazuje údaj z pamäťovej bunky SKHO, do ktorej ukladá modul MER. Problém je v tom, že údaje sa menia príliš rýchlo a pri priebežnom zobrazovaní by boli nečitateľné. Spomalenie celej slučky je neprijateľné z hľadiska riadenia. Preto podprogram ZOBR zobrazuje len každú 256. nameranú hodnotu:

```
8055 3A6780   ZOBR:  LDA  PARZ    ;vezmi parameter cyklu
                                   ;zobrazovania
8058 3C              INR  A       ;a inkrementuj ho
8059 326780          STA  PARZ    ;ulož novú hodnotu para-
                                   ;metra
805C C0              RNZ          ;návrat, ak ešte nebol
                                   ;256. cykl.
805D 3A8580          LDA  SKHO    ;vezmi nameranú hodnotu
8060 11FB83          LXI  D,83FBH ;štvrtý displej zľava
8063 CD9802          CALL DBY2    ;zobraz skutočnú hodnotu
8066 C9              RET          ;návrat z podprogramu
8067 FF       PARZ:  DB   0FFH    ;parameter zobrazovania
```

Parameter zobrazovania je v pamäťovej bunke PARZ. Jej obsah sa cyklicky inkrementuje a pri výskyte nulového obsahu sa údaj nameranej hodnoty zobrazí v ľavej časti displeja. Čas merania rýchlosti je závislý od rýchlosti a preto kolíše aj frekvencia zobrazovania (najvyššia je pri najvyšších otáčkach). Aby sa na prvú hodnotu nečakalo príliš dlho, je na začiatku programu nastavený parameter cyklu na hodnotu FF_H.

Uvedený cvičný program ukazuje možnosť zapojenia procesora do jednoduchej regulačnej slučky. Rýchlosť motora možno regulovať v širokom rozsahu (lineárna regulácia periódy).

12. Cvičné programy s analógovým prevodníkom

Periféria analógového prevodníka ŠMS je opísaná v čl. 7. Poskytuje pre používateľa možnosť vykonať D/A a A/D prevod, pravda nie súčasne. Ďalej opíšeme niekoľko programov s využitím analógového prevodníka. Pri laborovaní s týmito programami treba používať vstup resp. výstup analógového signálu. V čl. 14 sú uvedené niektoré námety na použitie cvičných prídavných zariadení, ktoré nie sú štandardnou súčasťou ŠMS a možno ich výhodne použiť pri cvičnom programovaní (ŠMS neobsahuje napr. meniteľný zdroj analógového signálu).

Analógový prevodník je najjednoduchšie využiteľný na D/A prevod. Zostavíme program na zadávanie byte z klávesnice a prevod na analógové napätie. Podľa obr. 7.1 treba nastaviť P1C0=0 a na port P1B priviesť byte. To dosiahneme jednoduchým programom:

```
                     ;cvičný program na D/A prevod
                     ORG  8200H  ;ukladacia adresa progra-
                                 ;mu
8200 3E90     CP29:  MVI  A,90H  ;riadiaci byte B-out,
                                 ;C-out
```

```
8202 D307            OUT  7       ;výstup na 8255 č.1
8204 CD3603  CYKL:   CALL ENTBY   ;vstup byte z klávesnice
8207 7D              MOV  A,L     ;presuň zadaný byte
8208 D305            OUT  5       ;výstup do D/A prevodníka
820A C30482          JMP  CYKL    ;opakuj cyklus
```

Stav P1C0=0 dosiahneme nastavením P1C do výstupného režimu. Spôsob zadávania byte je daný vlastnosťami podprogramu ENTBY. V čase, keď procesor zotrváva v ENTBY, sa na výstupe D/A prevodníka generuje napätie naposledy vyslaného byte (zadanú kombináciu zachytí register P1B obvodu 8255). Výstupné napätie možno indikovať niektorou cvičnou perifériou, napr. LED - diódou alebo meracím prístrojom.

Zostavme podprogram na A/D prevod, ktorý bude riadený programom. Použijeme pritom metódu postupnej inkrementácie z čl. 7. Podprogram bude od nuly cyklicky zvyšovať výstupné napätie D/A prevodníka. Proces sa zastaví vtedy, keď komparátor zistí vyššie napätie D/A prevodníka ako je vstupné analógové napätie. Vtedy aj budeme príslušný kód naposledy vyslaný do D/A prevodníka považovať za nameranú hodnotu.

```
                     ORG  8000H   ;začiatočná adresa pod-
                                  ;programu
8000 C5      ADI:    PUSH B       ;uschovaj BC dvojicu
8001 97              SUB  A       ;nuluj akumulátor
8002 D305    CYKL:   OUT  5       ;výstup do D/A
8004 47              MOV  B,A     ;schovaj vyslaný kód
8005 3E20            MVI  A,20H   ;kód zdržania asi 240 us
8007 CD3802          CALL DELYA   ;programové zdržanie po-
                                  ;dľa A-reg.
800A DB0D            IN   0DH     ;vstup z P2V (kvôli P2B3)
800C E608            ANI  00001000B; maska pre P2B3
800E C21780          JNZ  HTV     ;ak P2B3=1, skoč na hotovo
```



```
8011 78              MOV  A,B     ;vezmi schovaný kód
8012 C601            ADI  1       ;inkrementuj testovaný
                                  ;kód
8014 D20280          JNC  CYKL    ;opakuj ak nebolo prete-
                                  ;čenie
8017 78      HTV:    MOV  A,B     ;vezmi schovaný kód
                                  ;výsledný
8018 C1              POP  B       ;obnov BC dvojicu
8019 C9              RET          ;návrat z podprogramu
```

Podprogram A/D programového prevodu používa register B, preto ho treba pred prepísaním na začiatku podprogramu uschovať. Cyklus inkrementácie začneme od obsahu akumulátora 00_H inštrukciou SUB A. Register B slúži na dočasné odloženie obsahu akumulátora, kým sa testuje odpoveď komparátora. Cyklus začína vyslaním byte do D/A prevodníka. Bezprostredne potom nemožno testovať výstup komparátora, treba zaistiť programové zdržanie kvôdli rýchlosti D/A prevodníka a komparátora (pozri čl.7). Po uplynutí doby asi 240 us testujeme stav komparátora, ktorého výstup je pripojený na bit P2B3. Ak zistíme jednotkový stav, prevod je hotový (napätie D/A prevodníka je už vyššie ako vstupné analógové napätie). Ak prevod ešte nie je hotový, treba inkrementovať schovaný kód a znova ho vyslať do D/A prevodníka. Môže však nastať situácia, že vstupné analógové napätie je vyššie ako najvyššie generovateľné napätie D/A prevodníka. Potom je vhodné, aby sa A/D prevodom odovzdal kód FF_H. Preto treba testovať pretečenie obsahu pri inkrementovaní, a ak nastalo, treba prevod ukončiť. Toto zabezpečí inštrukcia JNC na adrese 8014_H (inak by sa A/D prevod neukončil). Kvôli nastaveniu príznaku CY pri inkrementovaní treba použiť inštrukciu ADI a nie INR A (je kratšia, ale nenastavuje uvedený príznak). Na konci podprogramu treba z B-registra prevziať výsledný kód, ob-

noviť dvojicu BC a vykonať návrat. Je zrejmé, že podprogram odovzdáva nameranú hodnotu v akumulátore. Trvanie A/D prevodu závisí od meraného napätia a najdlhšie je pri najvyšších napätiach (minimálne 256 x 240us). Prevod sa ukončí pri akomkoľvek vstupnom napätí a nameraný kód je závislý na nastavení potenciometra ANALOG IN (obr. 7.1).

Činnosť uvedeného podprogramu možno sledovať v režime STEP (nepoužíva sa prerušenie). Potom je však vhodné vynechať programové oneskorenie DELYA. Funkciu vyskúšame hlavným programom, ktorý aj inicializuje porty a nameranú hodnotu aj zobrazí v pravej časti dipleja:

```
                        ;testovací program pre podprogram
                        ;AD1
                        ORG  8200H   ;ukladacia adresa pro-
                                     ;gramu
8200 3E90     CP30:     MVI  A,90H   ;riadiace slovo B-out,
                                     ;C-out
8202 D307               OUT  7       ;výstup riadenia do 8255
                                     ;č.1
8204 3E92               MVI  A,92H   ;riadiace slovo B-in
8206 D30F               OUT  0FH     ;výstup riadenia do 8255
                                     ;č.2
8208 CD0080   AD:       CALL AD1     ;volanie A/D prevodu
820B CD9502             CALL DBYTE   ;zobrazenie nameranej
                                     ;hodnoty
820E C30882             JMP  AD      ;opakuj meranie
```

Na začiatku treba nastaviť P1B do výstupného režimu (pre D/A), P1C do výstupného režimu (tým sa automaticky nastaví P1C0=0 na nastavenie spínača v obvode ZN 425E na obr. 7.1). Na zistenie výstupu komparátora potrebujeme nastaviť port P2B do vstupného režimu (P2B3 = výstup z komparátora). V súlade s poznámkou v [1] na str. 68 použijeme riadiace slovo 92_H. Tento zásah prakticky nemusíme vykonať, pretože po výskyte signálu RESET (napr. po zapnutí napájania ŠMS) sa všetky porty nastavia do vstupného režimu. Cyklus hlavného programu meria vstupné napätie a zobrazuje ho na displeji. Na vstup analógového signálu môžeme použiť niektorú cvičnú periférium z čl. 14.

Podprogram AD1 má tú nevýhodu, že je pri ňom procesor celkom zamestnaný. Doplňujúce obvody ŠMS umožňujú vykonať A/D prevod metódou postupnej inkrementácie automaticky. Princíp je vysvetlený v čl. 7. Zostavíme program, v ktorom sa bude A/D prevod riadiť automaticky a jeho ukončenie sa do procesora bude hlásiť prerušením od analógového komparátora. Procesor pri automatickom A/D prevode môže zotrvávať v hlavnom programe, ktorý je prerušovaný len pri ukončení prevodu:

```
                        ;program s automatickým A/D prevod-
                        ;níkom
                        ORG  8200H   ;adresa hlavného progra-
                                     ;mu
8200 CD0080   CP31:     CALL INIC    ;inicializácia portov
                                     ;atď.
8203 F7                 RST  6       ;prvý štart podprogramu
                                     ;PRER
8204 3A2D80   CYKL:     LDA  MER     ;vezmi nameranú hodnotu
8207 CD9502             CALL DBYTE   ;zobrazenie nameranej
                                     ;hodnoty
820A C30482             JMP  CYKL    ;opakuj zobrazovanie
```

Hlavný program neustále zobrazuje nameranú hodnotu, ktorá sa pri prerušení (ukončení A/D prevodu) uloží do pamäťovej bunky MER. Štartovanie nového prevodu sa vykonáva v podprograme PRER, preto ho prvýkrát treba spustiť umele

inštrukciou RST 6 (inak sa do neho skáče pri prerušení).

Podprogram na inicializáciu vykonáva tieto činnosti:
- inicializácia portov
- inicializácia časovača T2 pre automatický A/D prevodník
- nastavenie spínača v D/A prevodníku (pripojenie počítadla)
- uloženie adresy prerušovacej rutiny do tabuliek monitora

```
                        ORG   8000H   ;ukladacia adresa pod-
                                       ;programov
8000 3E92     INIC:     MVI   A,92H    ;riadiace slovo B-in,
                                       ;C-out
8002 D307               OUT   7        ;výstup riadenia do 8255
                                       ;č.1
8004 D30F               OUT   0FH      ;výstup riadenia do 8255
                                       ;č.2
8006 3EB5               MVI   A,0B5H   ;2. režim pre T2 dekadic-
                                       ;ky
8008 D317               OUT   17H      ;výstup riadenia do 8253
800A 3E80               MVI   A,80H    ;LSB počítadla T2
800C D316               OUT   16H      ;výstup LSB do T2
800E 3E04               MVI   A,04H    ;MSB počítadla T2
8010 D316               OUT   16H      ;výstup MSB do T2
8012 3E01               MVI   A,1      ;pripojenie počítadla
                                       ;k D/A
8014 D306               OUT   6        ;prevodníku (t.j. P1CO=1)
8016 211D80             LXI   H,PRER   ;adresa prerušovacej ru-
                                       ;tiny
8019 22EA83             SHLD  83EAH    ;ulož do tabuľky monito-
                                       ;ra
801C C9                 RET            ;návrat z podprogramu
```


Port P1B bude vstupný (vstup z D/A), P1C výstupný (nastaviť P1CO=1), P2B vstupný a P2C výstupný (povolenie prerušenia). Pre automatický A/D prevodník treba generovať počítadlom T2 impulzy s periódou 200 - 400 us. To dosiahneme nastavením T2 do režimu č. 2 s dekadickým počítaním. Periódu 240 us získame počítacou hodnotou 0480 (240 = - 480.0,5), pretože perióda ø2 je asi 0,5 us. Potom treba uložiť adresu prerušenia do tabuliek monitora a nastaviť P1CO=1.

Prerušovacia rutina má za úlohu prevziať z D/A prevodníka nameranú hodnotu, uložiť ju do pamäťovej bunky MER a pripraviť nové meranie:

```
801D F5       PRER:   PUSH  PSW      ;schovaj PSW
801E DB05             IN    5        ;vstup nameranej hodnoty
8020 322D80           STA   MER      ;ulož výsledok do MER
8023 3E07             MVI   A,00000111B ;nuluj počítadlo
                                       ;D/A prev.
8025 D30F             OUT   0FH      ;a povol prerušenie
                                       ;(P2C3=1)
8027 CD3802           CALL  DELYA    ;oneskorenie 7x7,5 us
802A F1               POP   PSW      ;obnov PSW
802B FB               EI             ;obnov stav procesora EI
802C C9               RET            ;návrat z prerušenia
```

Pri prerušení od komparátora sa zastaví počítadlo T2 a v počítadle D/A prevodníka je nameraná hodnota, ktorú možno čítať portom P1B. Nové meranie sa pripraví tak, že sa vynuluje počítadlo v D/A prevodníku a obnoví sa možnosť prerušenia od komparátora (P2C3=1 a EI). V okamihu po nulovaní počítadla D/A prevodníka je nastavená ešte stará hodnota napätia D/A prevodníka a aj komparátora. Preto je pred povolením stavu EI zaradené krátke oneskorenie na

ustálenie oboch analógových výstupov. Oneskorenie sa dosahuje podprogramom DELYA so začiatočným obsahom akumulátora 07_H (t.j. asi 53 us).

Uvedený program je príkladom riadenia periférnych technických prostriedkov mikropočítača, ktoré prerušením oznamujú ukončenie svojej činnosti. Počas činnosti môže procesor vykonávať inú činnosť. Poznamenajme, že ak je vstupné analógové napätie príliš veľké (za potenciometrom ANALOG IN je vyššie ako najvyššie napätie D/A prevodníka 2,55V), komparátor nikdy nedosiahne na výstupe log. 1 a preto ani nevznikne prerušovací signál o ukončení A/D prevodu (počítadlo v D/A prevodníku je cyklické). Naopak, záporné vstupné napätie sa kóduje ako najmenšie zobraziteľné napätie, t.j. kódom 00_H. Aj v tomto prípade je čas merania závislý na meranej hodnote a najväčší je pri najvyšších napätiach (až 256 x 240 us).

Nevýhodou metódy postupnej inkrementácie je jej malá rýchlosť pri jednoduchom riadení. Metóda postupnej aproximácie je oveľa rýchlejšia, vyžaduje však náročnejšie riadenie. V ŠMS ju možno demonštrovať len programovo. Zostavíme podprogram programového riadenia A/D prevodu postupnej aproximácie, ktorá je opísaná v čl. 7. Podprogram AD2 uložíme na to isté miesto ako AD1 a na jeho overenie funkcie použijeme program CP30. Základom podprogramu je osemkrokový cyklus, v ktorom sa v každom kroku pokusne nastavuje ďalší bit meraného kódu a podľa signálu komparátora sa ponechá jeho hodnota alebo zmení na opačnú.

```
                ;podprogram riadenia postupnej apro-
                ;ximácie A/D
                ORG  8000H  ;ukladacia adresa pod-
                            ;programu
8000 C5   AD2:  PUSH B      ;schovaj BC
```



```
8001 97         SUB  A      ;nuluj akumulátor
8002 0680       MVI  B,10000000B ;nastav najvyšší
                             ;bit kódu
8004 B0   CYKL: ORA  B      ;prepíš nastavovaný bit
                            ;do A-reg.
8005 D305       OUT  5      ;vyšli kód do D/A
8007 4F         MOV  C,A    ;schovaj akumulátor do
                            ;C-reg.
8008 3E30       MVI  A,30H  ;oneskorenie DELYA asi
                            ;360 us
800A CD3802     CALL DELYA  ;podprogram oneskorenia
800D DB0D       IN   0DH    ;vstup portu P2B (kvôli
                            ;P2B3)
800F E608       ANI  08H    ;zisti P2B3
8011 79         MOV  A,C    ;obnov kód v akumulátore
8012 CA1780     JZ   OBSK   ;obskoč zmenu bitu
8015 A8         XRA  B      ;zmeň naposledy nastave-
                            ;ný bit
8016 4F         MOV  C,A    ;schovaj nový kód do
                            ;C-reg.
8017 78   OBSK: MOV  A,B    ;priprav obsah B na po-
                            ;sunutie
8018 0F         RRC         ;posuň o bit doprava
8019 DA2180     JC   HTV    ;skoč ak je hotovo
801C 47         MOV  B,A    ;inak vráť posunuté do
                            ;B-reg.
801D 79         MOV  A,C    ;obnov meraný kód
801E C30480     JMP  CYKL   ;opakuj cyklus
8021 79   HTV:  MOV  A,C    ;presuň výsledok do aku-
                            ;mulátora
8022 C1         POP  B      ;obnov BC
8023 C9         RET         ;návrat z podprogramu
```

Uvedený A/D prevod začína s hodnotou nula v akumulátore. Najprv sa nastaví najvyšší bit akumulátora pomocou registra B, v ktorom je kombinácia 10000000. Tento kód sa vyskúša vyslaním do D/A prevodníku a po nutnom oneskorení (kvôli rýchlosti D/A prevodníka a komparátora) sa podľa výstupu komparátora (bit P2B3) jeho hodnota ponechá alebo sa zmení inštrukciou XRA B na adrese 8015_H. Tým sa zistila správna hodnota najvyššieho bitu. Register C sa pritom používa na úschovu obsahu akumulátora. Zmena konkrétneho bitu akumulátora funguje tak, že nonekvivalenciou XRA sa ponechajú bity kde má register B nuly a zmení sa ten bit (v našom prípade len jeden), kde je v B registri jednotka (ten bit akumulátora sa naposledy nastavoval). Na konci cyklu sa obsah B-registra posunie o miesto doprava a rovnakým spôsobom sa zistí ďalší bit meraného kódu. Cyklus (celé meranie) končí vtedy, ak pri posúvaní obsahu B registra vyjde jednička mimo osembitový rozsah. To by bol už deviaty cyklus a preto sa inštrukciou JC skočí na koniec podprogramu, kde sa nameranou hodnotou naplní akumulátor, obnoví sa BC dvojica a vykoná sa návrat z podprogramu.

Uvedený podprogram postupnej aproximácie je o málo dlhší (v pamäti) ako AD1, je však podstatne rýchlejší, pretože každý prevod sa vykoná len na osem cyklov (D/A prevod a komparácia). Zároveň je príkladom, ako možno modifikovať činnosť okolia mikropočítača zmenou programu (pri hardverovom riešení by bolo treba zmeniť technické prostriedky A/D prevodníka). Prevod sa ukončí vždy (aj pri vyššom vstupnom napätí), pričom záporné napätia sa kódujú kódom 00_H a nadmerné napätia kódom FF_H. Čas prevodu nie je závislý na veľkosti vstupného napätia. Aj v tomto prípade treba použiť vhodnú cvičnú periférium na vstup analógového signálu (čl. 14). Nameraná hodnota je závislá na nastavení potenciometra ANALOG IN.

Ďalej uvedieme program na ukážku obsluhy prerušenia z dvoch zdrojov prerušenia. Jedným z nich bude automatický A/D prevodník a druhým bude externý vstup EXT 4. Hlavný program bude mať inicializačnú časť a jednoduchý prázdny cyklus, ktorým budeme simulovať bežiaci program prerušovaný udalosťami (externý vstup, A/D prevod ukončený). Prerušovacia rutina má rozoznať, ktoré prerušenie nastalo (z ktorého zdroja), pretože hardverovo obidve prerušenia generujú RST 6. Na displeji sa má zobrazovať nameraná analógová hodnota a počet impulzov, ktorý doteraz nastal na vstupe EXT 4. Inicializačná časť hlavného programu je podobná ako v cvičnom programe CP31, preto je aj rovnako programovaná:

```
                        ;program na obsluhu dvoch prerušení
                  ORG   8200H   ;ukladacia adresa pro-
                                ;gramu
8200 3E92   CP32: MVI   A,92H   ;riadiace slovo B-in,
                                ;C-out
8202 D307         OUT   7       ;pre 8255 č.1
8204 D30F         OUT   0FH     ;pre 8255 č.2
8206 3EB5         MVI   A,0B5H  ;2. režim pre T2 deka-
                                ;dicky
8208 D317         OUT   17H     ;výstup riadenia pre 8253
820A 3E80         MVI   A,80H   ;LSB počítadla T2
820C D316         OUT   16H     ;výstup LSB do T2
820E 3E04         MVI   A,4     ;MSB počítadla T2
8210 D316         OUT   16H     ;výstup MSB do T2
8212 3E01         MVI   A,1     ;pripojenie počítadla
                                ; k D/A
8214 D306         OUT   6       ;prevodníku (P1C0=1)
8216 3E07         MVI   A,00000111B ;nulovanie D/A pre-
                                ;vodníka
```

```
8218 D30F          OUT   0FH      ;a povolenie prerušenia
                                  ;prevod.
821A 3E09          MVI   A,00001001B ;nulovanie záchyt.
                                  ;obvodu
821C D30F          OUT   0FH      ;a povolenie prer. pre
                                  ;EXT 4
821E C31E82  CYKL: JMP   CYKL     ;prázdny cyklus (hlavný
                                  ;prog.)
```

Inicializačnú časť nebudeme komentovať, je podobná ako v CP31. Na adrese 8216_H je štartovanie automatického A/D prevodu (nulovanie počítadla v D/A a povolenie prerušenia P2C3=1 od analógového komparátora). Na adrese $821A_H$ je príprava stavu pre prerušenie od EXT 4 (nulovanie záchytného preklápacieho obvodu a povolenie prerušenia P2C4=1 od EXT 4). Prázdny cyklus predstavuje prerušovaný hlavný program.

Prerušovacia rutina má za úlohu rozoznať zdroj prerušenia. Možno to vykonať testovaním byte prerušenia, ktorý možno čítať z portu P2B. Bit P2B4 signalizuje prerušenie od EXT 4 a bit P2B3 prerušenie od A/D prevodníka. Obsluha prerušenia spočíva zo zobrazenia nameranej hodnoty A/D prevodu resp. počtu impulzov na EXT 4.

```
                   ORG   8230H    ;adresa skoku pri RST 6
8230 DB0D    PRER: IN    0DH      ;vstup byte prerušenia
8232 E610          ANI   00010000B ;testuj P2B4
8234 CA4882        JZ    P1       ;preskoč ak P2B4=0
8237 3A5D82        LDA   BYTE     ;vezmi doterajší počet
                                  ;impulzov
823A 3C            INR   A        ;inkrementuj počet
823B 325D82        STA   BYTE     ;ulož počet
823E 11F983        LXI   D,83F9H  ;ľavé dve pozície dis-
                                  ;pleja
```



```
8241 CD9802        CALL  DBY2     ;zobraz počet impulzov
8244 3E09          MVI   A,9      ;nulovanie záchytného
                                  ;obvodu
8246 D30F          OUT   0FH      ;a povolenie prer. pre
                                  ;EXT 4
8248 DB0D    P1:   IN    0DH      ;vstup byte prerušenia
824A E608          ANI   00001000B ;testuj P2B3
824C CA5B82        JZ    RET      ;preskoč ak P2B3=0
824F DB05          IN    5        ;vstup nameranej hodnoty
                                  ;z P1B
8251 CD9502        CALL  DBYTE    ;zobrazenie nameranej hod-
                                  ;noty
8254 3E07          MVI   A,7      ;nulovanie D/A prevodníka
8256 D30F          OUT   0FH      ;a povolenie prerušenia
                                  ;od kompar.
8258 CD3802        CALL  DELYA    ;oneskorenie 7x7,5 us
825B FB      RET:  EI             ;obnov EI po prerušení
825C C9            RET           ;návrat z podprogramu
                                  ;prerušenia
825D 00      BYTE: DB    0        ;počet impulzov na EXT 4
```

Podprogram PRER na obsluhu prerušenia má dve časti, z ktorých prvá obsluhuje vstup EXT 4 a druhá ukončenie činnosti A/D prevodníka. Ak sa príslušné prerušenie nevyskytlo, zodpovedajúca časť programu sa obskočí inštrukciou JZ. Po obsluhe určitého prerušenia treba pripraviť podmienky na ďalšie prerušenie. V D/A prevodníku treba vynulovať jeho počítadlo a v obvodoch vstupu EXT 4, treba vynulovať záchytný register. Obsluha A/D prevodníka je podobná ako v programe CP31. Je zrejmé, že ak vzniknú naraz obidve prerušenia, najprv sa obslúži vstup EXT 4. Počas obsluhy automatického A/D prevodníka sa registrujú prípadné impulzy na EXT 4 (vďaka záchytnému obvodu, pozri čl.5), napriek tomu, že procesor je v stave DI (po prerušení). Registruje

sa, pravda, len jeden impulz (môže byť však veľmi krátky).

Uvedený program ilustruje programové riešenie úlohy v prípade, že existuje viac zdrojov prerušení a treba programovo rozoznať, ktoré skutočne nastali a spracovať ich podľa určených priorít. Dokonca je možné počas spracovania niektorého prerušenia akceptovať ďalšie prerušenie (dôležitejšie) a prejsť na jeho spracovanie. Po spracovaní dôležitejšieho sa vykoná návrat do menej dôležitej obslužnej rutiny prerušenia a z nej po skončení do hlavného programu. Získať informácie o aktivite zdrojov prerušenia možno vstupom z portu P2B a zamedziť ich uplatnenie (aj jednotlivo) možno bitmi portu P2C. Konkrétne programové riešenie závisí od riešenia technických prostriedkov.

Pri laborovaní s programom je vhodné použiť cvičné prídavné zariadenia analógového a diskrétneho vstupu (potenciometer, prepínač alebo snímač na hriadeli motora).

V laboratórnej praxi sa často vyskytuje úloha zaznamenať určitý dej, ktorý má príliš veľkú rýchlosť v porovnaní s dostupnými registračnými prístrojmi. Tu sa využíva mikropočítač na rýchly analógový vstup s ukladaním číslicových údajov do pamäti. Po ukončení procesu možno (mimo reálneho času) údaje z pamäti vyberať a menšou rýchlosťou generovať pomocou D/A prevodníka do pomalých registračných prístrojov alebo možno tieto údaje mikropočítačom priamo spracovať alebo v číslicovej forme na vhodnom médiu (diernej páske) preniesť na väčší počítač. Tak možno pomocou mikropočítača dosiahnuť novú kvalitu v sortimente funkcií určitého prístroja (pri neveľkých nákladoch).

Uvedieme demonštračný program na rýchly analógový vstup s uložením dát do určenej oblasti pamäti. Ako zdroj

rýchleho analógového deja použijeme cvičný elektromotor, ktorý je pri otáčaní zdrojom napätia (jednosmerný motor s magnetom). Motor rozbehneme pripojením na +5V a po vypnutí napájania sa na jeho svorkách objaví napätie úmerné postupne sa znižujúcim otáčkam (zotrvačnosťou beží asi 1 s).

Na meranie použijeme modul AD2 riadenia A/D prevodu postupnou aproximáciou, ktorý je uvedený v tomto článku. Rýchlosť odoberania vzoriek napätia zvolíme na 8 ms (vzorkovacia perióda). Údaje budeme ukladať od adresy 8050_H v množstve 256 vzoriek (byteov). Ukladanie začneme až vtedy, keď sa vyskytne menšia nameraná hodnota ako $C0_H$. Takto bude meranie synchronizované od vstupnej analógovej veličiny (pokles napätia zrejme synchronizuje odpojenie motora). Vzorkovaciu periódu nastavíme časovačom T1 s využitím prerušenia. V prerušovacej rutine sa budú namerané údaje ukladať do pamäti.

```
                        ;rýchly analógový vstup s uložením
                        ;dát
                        ORG   8200H   ;ukladacia adresa pro-
                                      ;gramu
8200 3E90     CP33:     MVI   A,90H   ;riadenie 8255 B-out,
                                      ;C-out
8202 D307               OUT   7       ;výstup riadenia pre
                                      ;8255 č.1
8204 3E92               MVI   A,92H   ;riadenie 8255  B-in,
                                      ;C-out
8206 D30F               OUT   0FH     ;výstup riadenia pre
                                      ;8255 č.2
8208 3E64               MVI   A,64H   ;riadenie 8253, režim
820A D317               OUT   17H     ;č.2 pre počítadlo č.1,
                                      ;len MSB
```

```
820C 3E40          MVI  A,40H  ;MSB pre T1 (8 ms perió-
820E D315          OUT  15H    ;výstup MSB do T1
8210 215080        LXI  H,8050H ;ukladacia adresa dát
8213 0600          MVI  B,0    ;nastavenie parametra
                               ;cyklu
8215 CD0080   N1:  CALL AD2    ;štart A/D prevodníka
8218 CD9502        CALL DBYTE  ;zobrazenie nameraného
                               ;údaju
821B FEC0          CPI  0C0H   ;jeho testovanie na veľ-
                               ;kosť
821D D21582        JNC  N1     ;vráť sa pri veľkom vstu-
                               ;pe A/D
8220 3E03          MVI  A,00000011B ;nuluj záchytný re-
                                    ;gister a
8222 D30F          OUT  0FH    ;povol prerušenie pre T1
8224 76       N2:  HLT         ;čakaj na pokyn od T1
                               ;(preruš.)
8225 04            INR  B      ;inkrementuj parameter
                               ;cyklu
8226 C22482        JNZ  N2     ;opakuj cyklus 256 krát
8229 E7            RST  4      ;návrat do monitora
```

Na začiatku programu sú inicializované porty tak, ako to vyžaduje podprogram AD2. Počítadlo T1 je nastavené do režimu č.2 (generovanie krátkych impulzov s danou periódou) s periódou 8 ms (zadaná hodnota 4000_H). Dvojica HL slúži ako ukazovateľ ukladania dát do pamäti, register B je parametrom cyklu na odpočítanie 256 meraní. V slučke N1 sa zobrazuje a priebežne testuje veľkosť nameraného napätia. Keď kód prvýkrát presiahne hodnotu (smerom k nižším číslam) $C0_H$, nastavia sa podmienky na prerušenie od T1 (taktovanie 8 ms). Na prerušenie sa vyčkáva inštrukciou HLT. Po každom meraní sa zvyšuje parameter cyklu a po 256 meraniach sa riadenie odovzdá monitoru (cez RST 4).

Prerušovacia rutina je umiestnená na adrese 8230_H (tam sa pokračuje po RST 6). Jej úlohou je vykonať meranie a zaviesť údaj do pamäti. Na konci podprogramu sa nastavia podmienky pre ďalšie prerušenie od T1 (vynulovať záchytný obvod a nastaviť EI).

```
                   ;podprogram prerušenia
                   ORG  8230H  ;adresa po RST 6
8230 CD0080  PRER: CALL AD2    ;vykonaj meranie A/D
8233 77            MOV  M,A    ;prenes údaj do pamäti
8234 23            INX  H      ;zvýš ukazovateľ pamäti
8235 3E03          MVI  A,3    ;nuluj záchytný register
8237 D30F          OUT  0FH    ;a povol prerušenie od T1
8239FB             EI          ;nastav EI
823A C9            RET         ;návrat z prerušenia
```

Svorky 0V a +5V periférie motora prepojíme s kontaktovými poliami takto:

- 0V - GND
- +5V - ANALOG IN

Program použijeme tak, že ďalším vodičom spojíme kontaktové pole +5V a ANALOG IN, čím sa motor rozbehne. Potom spustíme program (od adresy 8200_H) a odpojíme budenie motora (ručne). Pri zotrvačnom dobehu motora sa na jeho svorkách generuje napätie. Ukladanie dát sa začne vtedy, ak prvá hodnota poklesne pod $C0_H$ a bude trvať 256 x 8ms = 2,048 s. Uložené dáta možno prezerať prostriedkami monitora (prezeranie pamäti) od adresy 8050_H do $814F_H$. Nameraný priebeh vykazuje trvalý pokles k nule so superponovanými výkyvmi, ktoré sú spôsobené konštrukciou motora (trojpólový rotor). Vzorkovaciu periódu možno skrátiť zadaním iného byte v in-

štrukcii na adrese $820C_H$. Nesmie však byť kratšia ako čas A/D merania (osemkrát 360 us = 2,88 ms). Zadaním byte 20_H do počítadla T1 dosiahneme periódu vzorkovania 4 ms.

Výstup uložených dát dosiahneme iným programom. Zostavíme ho tak, že bude vyberať uložené dáta od adresy 8050_H, kód prevedie D/A prevodníkom na napätie a údaj zároveň zobrazí na displeji. Vyberieme 256 byteov uložených v programe CP33 rýchlosťou 1 vzorka za sekundu (rýchle ukladanie, pomalé vyberanie). Časový interval 1 s dosiahneme programovo použitím podprogramu DELYH z čl. 3, ktorý umiestnime na adresu $5F82_H$:

```
                        ;program na analógový výstup ulože-
                        ;ných dát
                        ORG  8240H  ;ukladacia adresa
8240 3E90     CP34:     MVI  A,90H  ;riadenie 8255 B-out,
                                    ;C-out
8242 D307               OUT  7      ;výstup riadenia pre
                                    ;8255 č.1
8244 015080             LXI  B,8050H; začiatok uložených dát
8247 1600               MVI  D,0    ;parameter cyklu vybera-
                                    ;nia dát
8249 21E803   CYKL:     LXI  H,1000 ;časová konštanta pre
                                    ;DELYH
824C 0A                 LDAX B      ;vyber ďalší byte dát
824D D305               OUT  5      ;výstup do D/A
824F C5                 PUSH B      ;schovaj ukazovateľ pa-
                                    ;mäti (dáta)
8250 D5                 PUSH D      ;schovaj parameter cyk-
                                    ;lu
8251 CD9502             CALL DBYTE  ;zobraz dáta na displeji
8254 D1                 POP  D      ;obnov D
8255 C1                 POP  B      ;obnov BC
8256 03                 INX  B      ;zvýš ukazovateľ dát
                                    ;v pamäti
```



```
8257 CD5F82             CALL DELYH  ;programové zdržanie
                                    ; 1 s
825A 14                 INR  D      ;zvýš parameter cyklu
                                    ;vyberania
825B C24982             JNZ  CYKL   ;opakuj cyklus, ak ne-
                                    ;skončil
825E E7                 RST  4      ;späť do monitora
```

Na začiatku programu sa nastavia porty tak, ako to vyžaduje D/A prevod. Dvojica BC slúži ako ukazovateľ pri vyberaní dát z pamäti (ich začiatok z programu CP33 je 8050_H) a register D slúži ako parameter 256-násobného cyklu vyberania z pamäti. Podprogram DELYH oneskoruje podľa kódu v dvojici HL, ktorá sa interpretuje ako počet milisekúnd. V našom prípade nech je perióda 1 s. Kód sa vyberie z pamäti, vyšle sa do D/A prevodníka a tiež sa zobrazí na displeji. Podprogram DBYTE kazí obsahy registrov A,C,D,E a preto ich treba uschovať do zásobníka (len tie, na ktorých záleží). Potom treba zvýšiť ukazovateľ pamäti, urobiť zdržanie 1 s a cyklus opakovať podľa registra D 256-krát. Celý výstup dát trvá 256 sekúnd a potom sa riadenie odovzdá monitoru.

```
                        ;programové zdržanie podľa obsahu
                        ;HL
                        ORG  825FH  ;umiestnenie
825F CD3602   DELYH:    CALL DELAY  ;oneskorenie 1 ms
8262 2B                 DCX  H      ;zníž parameter slučky
8263 7C                 MOV  A,H    ;prenes vyšší byte pa-
                                    ;rametra
8264 B5                 ORA  L      ;nie sú jednotky v L?
8265 C25F82             JNZ  DELYH  ;opakuj slučku kým HL
                                    ;nie je nula
8268 C9                 RET         ;návrat z podprogramu
```

Podprogram DELYH napočíta potrebný počet milisekúnd. Nulový obsah dvojice HL sa zisťuje logickou operáciou ORA testovaním na jedničky.

Študujúcemu čitateľovi odporúčame premysliť a odskúšať modifikáciu programov CP33 a CP34 tak, aby bolo možné zadávať vzorkovaciu periódi (zvlášť pri vstupe a pri výstupe) a počet ukladaných údajov. Zrejme bude treba pritom optimalizovať umiestnenie programov a podprogramov (aby sa pre dáta vytvoril súvislý čo najväčší pamäťový priestor).

Mikropočítač možno v prístrojoch využiť aj na priame generovanie analógového priebehu podľa daného predpisu. Príkladom môže byť cvičný program na generovanie jedného lineárne narastajúceho rampového priebehu. Nech sa tvorí D/A prevodníkom napätie od najnižšieho po najvyššie s časovým krokom 0.1 s, t.j. celá rampa bude trvať 25,6 sekúnd. Program umiestnime od adresy 8270_H:

```
                    ;program na generovanie lineárneho
                    ;priebehu
                    ORG  8270H   ;adresa programu
8270 3E90   CP35:   MVI  A,90H   ;riadenie 8255 B-out,
                                 ;C-out
8272 D307           OUT  7       ;výstup riadenia 8255
                                 ;č.1
8274 0600           MVI  B,0     ;parameter cyklu gene-
                                 ;rovania
8276 216400 CYKL:   LXI  H,100   ;parameter oneskorenia
                                 ;0,1 s
8279 78             MOV  A,B     ;generovaný kód
```



```
827A D305           OUT  5       ;výstup do D/A
827C CD9502         CALL DBYTE   ;zobrazenie kódu
827F CD5F82         CALL DELYH   ;oneskorenie podľa HL
8252 04             INR  B       ;zvýš parameter cyklu
8253 C27682         JNZ  CYKL    ;opakuj cyklus podľa B
8256 E7             RST  4       ;späť do monitora
```

Parametrom cyklu generovanie je register B. Na zdržanie 0,1 s použijeme podprogram DELYH z predošlého programu (musí byť v pamäti). Kód generovaného napätia sa zároveň zobrazuje na displeji. Ak inštrukciu JNZ na adrese 8253_H nahradíme inštrukciou JMP, dosiahneme trvalé generovanie pílového priebehu. Je zrejmé, že možno využiť výpočtovú kapacitu mikroprocesora na generovanie zložitejšieho analógového priebehu a pomocou počítadiel s prerušeniami docieliť presnejšie sledovanie reálneho času (pri použití DELYH sme neuvažovali čas potrebný na vykonanie inštrukcií programu).

Čitateľovi odporúčame zostaviť program na generovanie trojuholníkového trvalého priebehu (nie pílového) s danou časovou konštantou.

13. Cvičné programy bez cvičných prídavných zariadení

Mikropočítač pracuje v aplikáciách často v reálnom čase, t.j. programátor potrebuje zisťovať reálny čas alebo merať časové úseky. Na to potrebuje mikropočítač mať generátor časových impulzov (s vyššou frekvenciou), programovateľné počítadlá a podprogramy na obsluhu týchto technických prostriedkov.

Zostavíme cvičný program s jednoduchou funkciou - elektronické stopky. Ako zdroj časových impulzov použijeme kryštálom riadený signál procesora ø2, ktorého frekvenciu vydelíme vhodným číslom pomocou programovateľného počítadla. Na displeji budeme zobrazovať v dekadickom tvare uplynuté sekundy, minúty a hodiny.

Hlavný program je jednoducho štruktúrovaný:

```
                    ; elektronické stopky
                    ORG  8200H    ; začiatočná adresa
8200 CD0982  CP36:  CALL NUL      ; nulovanie stopiek
8203 CD1282         CALL INIC     ; inicializácia počítadla
                                    a portu
8206 C30682  CYKL:  JMP  CYKL     ; simulácia hlavného pro-
                                    gramu
```

Podprogram NUL nastaví čas "stopiek" na nulovú hodnotu (bude sa registrovať čas od spustenia programu). Podprogram INIC inicializuje počítadlo T0 a povoľuje jeho prerušenie. Procesor nie je zamestnaný odpočítavaním času (vybavuje len prerušenia od T0) a preto môže zotrvávať v hlavnom programe, čo budeme simulovať prázdnym cyklom na adrese 8206_H.

Na uloženie okamžitej hodnoty času použijeme tri byte pamäti (hodiny, sekundy, minúty). Tieto pamäťové bunky

treba vynulovať:

```
                    ORG  8209H    ;adresa podprogramov
8209 0603    NUL:   MVI  B,3     ;potrebný počet cyklov
820B 217D82         LXI  H,HOD   ;adresa pamäťových buniek
                                   (čas)
820E CD8C02         CALL CLRLP   ;nulovanie úseku pamäti
8211 C9             RET          ;návrat z podprogramu
```

Na nulovanie použijeme monitorový podprogram CLRLP, ktorý nuluje oblasť nižšie od adresy v HL dvojici v počte byte, ktorý je zapísaný v registri B.

Pri inicializácii treba nastaviť počítadlo T0 (budeme ho používať ako pracovné) do režimu trvalého generovania vhodnej frekvencie. Z hľadiska funkcie by sa hodila frekvencia 1 Hz. To však nie je možné, lebo počítadlo má len 16 bitov (=2^{16} stavov) a vstupná frekvencia je až 2048 kHz (=2^{11} kHz).
Najdlhší generovateľný interval je teda $2^{16}/2^{11}$ = 32 ms. Bude preto treba vytvoriť programové počítadlo modulo 32, čím získame potrebné jednosekundové intervaly. Potom však treba, aby sa počítadlom 8253 generovala frekvencia 1/32 s, t.j. 31,25 ms. Tomu zodpovedá počítacia hodnota $FA00_H$. Okrem toho pri inicializácii treba nulovať aj programové počítadlo (ktoré má modulo 32), ktoré má parameter cyklu v pamäťovej bunke PAR. Nakoniec treba povoliť prerušenie od T0 (P2C0=1) a nulovať záchytný preklápací obvod prerušenia T0:

```
                    ORG  8212H
8212 3E24    INIC:  MVI  A,24H    ;režim č.2 pre T0 (binár-
                                   ne)
8214 D317           OUT  17H     ;výstup riadenia pre 8253
8216 3EFA           MVI  A,0FAH  ;konštanta pre 31,25 ms
8218 D314           OUT  14H     ;výstup len MSB pre T0
```

```
821A 97          SUB   A       ; nuluj akum.
821B 327A82      STA   PAR     ;nuluj programové počítadlo
821E 3E92        MVI   A,92H   ;riadenie 8255  B-in, C-out
8220 D30F        OUT   0FH     ;výstup riadenia pre 8255
                               ;č.2
8222 3E01        MVI   A,1     ;nastavenie P2C0=1 a záro-
                               ;veň
8224 D30F        OUT   0FH     ;nulovanie záchytného obvodu
8226 C9          RET           ;návrat z podprogramu
```

Podprogram prerušenia má počítať modulo 32 (výskyt prerušení od T0) a každú sekundu pripočítať ďalšiu sekundu. Počítanie sekúnd a minút má byť dekadické modulo 60 a hodín modulo 24.

```
                 ORG 8228H   ; adresa po RST 5 od T0
8228 3A7A82  PRER: LDA PAR    ;parameter programového
                               ;počítadla
822B 3C          INR A        ;inkrementácia parametra
822C FE20        CPI 32       ;bolo už 32  prerušení?
822E C27182      JNZ K        ;ak nie tak iba ulož para-
                               ;meter
8231 11FF83      LXI D,83FFH  ;inak nahraj adresu disple-
                               ;ja
8234 3A7B82      LDA SEK      ;vezmi sekundy
8237 3C          INR A        ;inkrementuj ich
8238 27          DAA          ;uprav dekadický tvar
8239 FE60        CPI 60H      ;bolo už 60 sekúnd?
823B C24082      JNZ  S1      ;ak nie, obskoč nulovanie
823E 3E00        MVI  A,0     ;nuluj sekundy
8240 327B82  S1:  STA SEK     ;ulož novú hodnotu sekúnd
8243 CD7E82      CALL DBY     ;zobraz sekundy (akumulá-
                               ;tor)
8246 C27082      JNZ  K1      ;preskoč ak nebol prenos
                               do minút
```



```
8249 3A7C82      LDA  MIN     ;vezmi minúty
824C 3C          INR  A       ;inkrementuj ich
824D 27          DAA          ;uprav dekadický tvar
824E FE60        CPI  60H     ;bolo už 60 minút?
8250 C25582      JNZ  M1      ;ak nie, obskoč nulovanie
8253 3E00        MVI  A,0     ;nuluj minúty
8255 327C82  M1: STA MIN      ;ulož novú hodnotu minút
8258 CD7E82      CALL DBY     ;zobraz minúty (akumulátor)
825B C27082      JNZ K1       ;preskoč, ak nebol prenos
                               ;hodín
825E 3A7D82      LDA  HOD     ;vezmi hodiny
8261 3C          INR  A       ;inkrementuj ich
8262 27          DAA          ;uprav dekadický tvar
8263 FE24        CPI  24H     ;bolo už 24 hodín?
8265 C26A82      JNZ  H1      ;ak nie, obskoč nulovanie
8268 3E00        MVI  A,0     ;nuluj hodiny
826A 327D82  H1: STA HOD      ;ulož novú hodnotu hodín
826D CD7E82      CALL DBY     ;zobraz hodiny (akumulátor)
8270 97      K1: SUB  A       ;nuluj parameter prog.počí-
                               ;tadla
8271 327A82  K:  STA PAR      ;a ulož novú hodnotu
8274 3E01        MVI A,1      ;nastav P2C0=1 a zároveň nu-
8276 D30F        OUT 0FH      ;luj záchytný obvod preruše-
                               ;nia
8278 FB          EI           ;povoľ prerušenie procesora
8279 C9          RET          ;návrat z prerušenia
827A       PAR:  DS  1        ;programové počítadlo
827B       SEK:  DS  1        ;sekundy
827C       MIN:  DS  1        ;minúty
827D       HOD:  DS  1        ;hodiny
827E F5    DBY:  PUSH PSW     ;schovaj príznaky
827F CD9802      CALL DBY2    ;zobrazenie (kazí príznaky)
8282 1B          DCX  D       ;vynechaj jeden displej
```

```
8283 F1          POP  PSW    ;obnov príznaky
8284 C9          RET         ;návrat z DBY
```

Programové počítadlo má parameter cyklu v pamäťovej bunke na adrese $827A_H$ (PAR). Celkove je činnosť podprogramu PRER zrejmá z jeho komentárov. Poznamenáme iba toľko, že podprogram monitora DBY2 kazí príznaky a preto je použitý pomocný podprogram DBY, ktorý túto vlastnosť nemá. Okrem toho sa v ňom vytvorí medzera medzi jednotlivými údajmi na displeji (pomocou DCX D). Na konci podprogramu sú nastavené podmienky pre uplatnenie ďalšieho prerušenia.

Vidno, že podprogram PRER má tri veľmi podobné časti na obsluhu počítania sekúnd, minút a hodín. Preto by bolo možné program optimalizovať z hľadiska spotreby pamäti použitím cyklu. Túto úpravu ponecháme na študujúceho čitateľa.

Uvedený cvičný program je použiteľný na meranie časového úseku od istej udalosti. V našom prípade je ňou spustenie programu od adresy 8200_H. Je zrejmé, že čas možno merať čítaním z buniek SEK, MIN a HOD. Takto majú k časovej informácii prístup všetky programy mikropočítača. Nižšie rády časovej miery možno zistiť čítaním z bunky PAR (zlomky sekundy 1/32) a z počítadla TO (počet polmikrosekúnd) s využitím vzorkovania jeho obsahu. V prípade potreby možno meranie času započať znova pomocou podprogramov NUL a INIC.

Inou úlohou je zostaviť hodiny reálneho času. Od stopiek sa líšia tým, že bežia trvale a ich údaj súhlasí napr. so SEČ. Zostavíme taký program s použitím predošlých programov. Zmena spočíva v tom, že na začiatku nebudeme nulovať čas hodín, ale ho nastavíme podľa vstupov z klávesnice. Na to použijeme cvičný program CP37, ktorý naplní bunky SEK, MIN a HOD a priamo prejde na inicializáciu

v predošlom programe. Na vstup využijeme podprogram ENTBY so zobrazovaním:

```
                        ;nastavenie času v stopkách
                        ORG  8290H   ;začiatok programu
8290 CD0982   CP37:     CALL NUL     ;predbežné nulovanie
                                     ;času
8293 217D82             LXI  H,HOD   ;adresa prvej bunky
8296 1E03               MVI  B,3     ;počet cyklov slučky
8298 CDAA82   CYK:      CALL ZOBR    ;zobrazenie všetkého
                                     ;(s, min, h)
829B E5                 PUSH H       ;schovaj HL (ENTBY kazí)
829C CD3603             CALL ENTBY   ;vstup byte z klávesnice
829F 7D                 MOV  A,L     ;prenes do A
82A0 E1                 POP  H       ;obnov HL - adresa bunky
82A1 77                 MOV  M,A     ;zanes byte do pamäti
82A2 2B                 DCX  H       ;adresa ďalšej bunky
82A3 1D                 DCR  E       ;parameter cyklov slučky
82A4 C29882             JNZ  CYK     ;ak treba, opakuj slučku
82A7 C30382             JMP  8203H   ;choď na stopky bez nu-
                                     ;lovania
                        ;podprogram zobrazovania celého
                        ;displeja
82AA E5       ZOBR:     PUSH H       ;schovaj HL
82AB D5                 PUSH D       ;schovaj DE
82AC 0603               MVI  B,3     ;počet cyklov
82AE 11FF83             LXI D,83FFH  ;pravý displej
82B1 217B82             LXI H,SEK    ;adresa najnižšej bunky
82B4 7E       CYKZ:     MOV A,M      ;vezmi obsah bunky
82B5 CD9802             CALL DBY2    ;a zobraz ju na prísluš-
                                     ;nej pozícii
82B8 23                 INX  H       ;ďalšia bunka
82B9 1B                 DCX  D       ;vynechaj jeden displej
82BA 05                 DCR  B       ;parameter cyklov
```

```
82BB C2B482        JNZ  CYKZ    ;opakuj cyklus trikrát
82BE D1            POP  D       ;obnov DE
82BF E1            POP  H       ;obnov HL
82C0 C9            RET          ;návrat z podprogramu
                                ;ZOBR
```

Hodiny, minúty a sekundy sa zadávajú v cykle CYK pomocou podprogramu ENTBY (zadať dve číslice ukončené červeným klávesom). Po každom zadaní sa zobrazia údaje na príslušných miestach. Po zadaní posledného sa prejde do predošlého programu od miesta, kde začína inicializácia činnosti hodín. Podprogram ZOBR zobrazuje údaje na celom displeji. Používa sa pritom podprogram DBY2, ktorý kazí obsahy registrov (preto niektoré dôležité treba uschovať). Pri zobrazovaní sa využíva skutočnosť, že podprogram DBY2 automaticky dekrementuje ukazovateľ displeja, dvojicu DE. Kvôli prehľadnosti sú dvojčíslia oddelené prázdnym displejom (zabezpečí DCX D).

Uvedený cvičný program použijeme tak, že ho spustíme od adresy 8290_H a zadáme správny čas už uvedeným spôsobom. Potom hodiny pokračujú v chode a čas sa trvale zobrazuje na displeji. Aj v tomto prípade by ktorýkoľvek iný program mohol zistiť okamžitý čas čítaním z buniek SEK, MIN a HOD.

Ďalší cvičný program bude mať za úlohu nájsť v súbore viacerých čísel v rozsahu jedného byte najväčšie číslo. Nech túto funkciu plní podprogram, pričom v dvojici HL bude adresa začiatku súboru byteov, počet byteov bude v registri B a čísla sa budú interpretovať ako celé bez znamienka. Pri výstupe z podprogramu nech je hľadané maximum v registri C a jeho adresa v dvojici DE:

```
                ;podprogram na hľadanie maxima v súbore
                ORG  8000H   ;adresa podprogramu
```



```
8000 E5     CP38:  PUSH H     ;prenes obsah HL a daj ho
8001 D1            POP  D     ;preventívne do DE
8002 0E00          MVI  C,0   ;priprav do C absolútne
                              ;minimum
8004 7E     CYKL:  MOV  A,M   ;vezmi byte z pamäti
8005 B9            CMP  C     ;porovnaj s doterajším ma-
                              ;ximom
8006 DA0C80        JC   N     ;skok ak nie je nové maxi-
                              ;mum
8009 4F            MOV  C,A   ;inak presuň nové maximum
800A E5            PUSH H     ;prenes HL do
800B D1            POP  D     ;dvojice DE
800C 23     N:     INX  H     ;adresa ďalšej pamäťovej
                              ;bunky
800D 05            DCR  B     ;parameter cyklu
800E C20480        JNZ  CYKL  ;opakuj cyklus podľa B
8011 C9            RET        ;návrat z podprogramu
```

Činnosť podprogramu vyskúšame programom, ktorý pripraví v B registri hodnotu 10 a v dvojici HL adresu 8100_H. Maximum a jeho adresa sa zobrazia na displeji. Program končí prázdnym cyklom, takže údaje ostanú vysvietené na displeji. Tento ukončovací cyklus možno nahradiť aj inštrukciou HLT, prepínač STEP/AUTO však musí byť v polohe AUTO /po monitorovom prerušení by sa pokračovalo za inštrukciou HLT/. Pred spustením programu treba do oblasti $8100_H \div 8109_H$ ručne zapísať kontrolné údaje (pomocou monitora). Po kontrole funkcie sa činnosť programu ukončí klávesom RESET.

```
                ;kontrolný program k vyhľadávaniu ma-
                ;xima
                ORG  8200H   ;začiatok programu
```

```
8200 210081    CP39:  LXI   H,8100H  ;začiatok dát
8203 060A             MVI   B,10     ;počet byteov
8205 CD0080           CALL  8000H    ;podprogram hľadania
                                     ;maxima
8208 D5               PUSH  D        ;prenes nájdenú adre-
8209 E1               POP   H        ;su do dvojice HL
820A 79               MOV   A,C      ;prenes zistené ma-
                                     ;ximum
820B CD9502           CALL  DBYTE    ;zobraz maximum
820E CDD102           CALL  DWORD    ;zobraz adresu maxima
8211 C31182    K:     JMP   K        ;prázdny koncový
                                     ;cyklus
```

Podobné úlohy na vyhľadanie určitého údaja podľa daného predpisu sa vyskytujú pri spracovaní napr. súboru čísel získaných rýchlym analógovým vstupom (súbor vzoriek) z analógového procesu (experimentu) a tiež pri spracovaní agendy viacerých údajov (napr. skladové hospodárstvo - administratívne mikropočítače).

Okrem riadenia periférií je ďalšou významnou súčasťou programového vybavenia súbor aritmetických modulov. Zostavíme niekoľko z nich s typickými funkciami. Mikropočítač poskytuje funkciu 8-bitového a 16-bitového sčítania, ktoré je vhodné (priamo) len na sčítanie celých čísel bez znamienka. Napríklad rozhodovanie o pretečení nie je jednoduché a treba ho robiť programovo. Zostavíme modul na sčítanie dvoch 16-bitových celých čísel se znamienkom. Dvojica HL bude vo funkcii akumulátora a druhý operand nech je v dvojici DE. Požadujeme, aby sa príznakom CY indikovalo pretečenie výsledku (nezobraziteľná hodnota). Pretečenie nastane vtedy, ak obidva operandy majú rovnaké znamienko a zároveň výsledok má iné znamienko ako operandy. Sčítavať budeme v doplnkovom kóde, takže znamienko čísla je v najvyššom bite zobrazenia. Moduly na aritmetické operácie uložíme od adresy 8300_H. Používateľ ich môže použiť pri tvorení svojich cvičných programov.


```
                       ;modul na 16-bitové binárne sčí-
                       ;tanie so znamienkom (CY=1 zna-
                       ;čí pretečenie)
                       ORG   8300H ;adresa aritmetických
                                   ;podprog.
8300 7C SU16:  MOV   A,H    ;vezmi znamienko operandu
8301 19        DAD   D      ;pričítaj DE k HL
8302 AA        XRA   D      ;porovnaj znamienka operandov
8303 F8        RM           ;návrat, ak boli rôzne znamienka
                            ;operandov
8304 7C        MOV   A,H    ;vezmi znamienko výsledku a
8305 AA        XRA   D      ;porovnaj so znamienkom operandu
8306 F0        RP           ;návrat, ak sú zhodné znamienka
                            ;operandu a výsledku
8307 37        STC          ;inak nastav CY = 1
8308 C9        RET          ;návrat z podprogramu
```

Na začiatku podprogramu sa odloží znamienko operandu v dvojici HL a vykoná sa 16 bitové binárne sčítanie so znamienkom v doplnkovom kóde. Inštrukciou XRA na adrese 8302_H sa zistí, či boli rôzne znamienka operandov a zároveň sa nuluje prenosový príznak. Ak boli rôzne, nehrozí pretečenie výsledku a možno urobiť návrat z podprogramu. Ak boli rovnaké, treba zistiť, či nie je iné znamienko výsledku ako operandov. To sa vykoná na adresách 8304_H a 8305_H. Ak sú znamienka zhodné, možno urobiť návrat. Ak nie, treba kvôli indikácii nastaviť prenosový príznak a urobiť návrat z podprogramu. Pri používaní tohto podprogramu platí zásada, že výsledok v HL je správny práve len vtedy, ak je CY=0. Podprogram kazí obsah registra A a ostatné príznaky nie sú správne nastavené. Je zaujímavé, že netre-

ba použiť prenos pri sčítaní v inštrukcii DAD.

Význam príznaku CY je spojený skôr s operáciou sčítania (alebo inou) a preto sme jeho tvorenie v uvedenom podprograme zaručili. Ostatné príznaky (Z, S), ktoré sa často používajú súvisia skôr s hodnotou čísla ako s aritmetickou operáciou. Preto zostavíme modul na tvorenie príznakov Z, S podľa 16 bitového čísla, ktoré je v dvojici HL. Takýto podprogram je použiteľný pre podprogram SU16, treba ho zaradiť za ním (po ňom). Číslo v dvojici HL sa opäť interpretuje ako celé v doplnkovom kóde.

```
                        ;nastavenie príznakov C, Z, S podľa
                        ;výsledku z SU16
                  ORG   8309H     ;bude za SU16
8309 4F     PRIZ: MOV   C,A       ;schovaj akumulátor
830A 3E00         MVI   A,0       ;nuluj budúce príznaky
830C 17           RAL             ;nastav CY, ak bolo CY=1
830D 47           MOV   B,A       ;schovaj doterajší obsah
                                  ;Akum.
830E 7C           MOV   A,H       ;testuj dvojicu HL
830F B5           ORA   L         ;na nulový obsah
8310 78           MOV   A,B       ;obnov doterajšie "prízna-
                                  ;ky"
8311 C21683       JNZ   N         ;obskoč, ak HL nie je nula
8314 F640         ORI   40H       ;nastav "Z" bit"príznakov"
8316 47     N:    MOV   B,A       ;schovaj doterajšie
                                  ;"príznaky"
8317 7C           MOV   A,H       ;vezmi znamienko z HL
8318 E680         ANI   80H       ;osamostatni ho
831A B0           ORA   B         ;daj k nemu ostatné
                                  ;"príznaky"
831B 41           MOV   B,C       ;presuň odložený obsah v C
831C 4F           MOV   C,A       ;daj hotové príznaky do
                                  ;nižšieho byte dvojice BC
```



```
831D C5           PUSH B          ;ulož C do zásobníka
831E F1           POP  PSW        ;čo bolo v C, daj do
                                  ;FLAG-ov
831F C9           RET             ;návrat z podprogramu
```

Podprogram PRIZ kazí obsah dvojice BC, takže používateľ v prípade potreby použije pred jeho volaním inštrukciu PUSH B. Toto nemožno urobiť v prípade registra A, lebo obnovením inštrukciou POP PSW by sa premazali aj príznaky, ktoré sa práve mali nastaviť. Preto na začiatku podprogramu PRIZ sa odloží obsah A do registra C (neskôr do B). Tým sa nepríjemná povinnosť neštandardne uschovávať akumulátor rieši v PRIZ. Najprv sa vynuluje akumulátor, pretože ten bude slúžiť na tvorenie budúcich príznakov. Príznak CY vytvorený (predtým volaným) podprogramom SU16 by sa nemal stratiť a preto sa uloží do akumulátora ako prvý inštrukciou RAL. Obsah A sa schová do B a testuje sa HL na nulu. Ak v HL je nula, nastaví sa bit č. 6 akumulátora (budúci Z-príznak). Potom sa vezme znamienko obsahu HL (7. bit H), osamostatní sa a pridajú sa k nemu ostatné doteraz vytvorené príznaky. Teraz treba obsah akumulátora presunúť do registra príznakov procesora. To sa vykoná nepriamo tak, že sa hotové príznaky presunú do nižšieho byte BC dvojice (C-register) a cez zásobník sa inštrukciami PUSH B a POP PSW dostanú do registra príznakov. Tým sa zároveň obnoví pôvodný obsah akumulátora, lebo do B sa medzitým presunul jeho pôvodný obsah (predtým v C).

Podprogram PRIZ kazí obsah dvojice BC a príznaky AC, P, ktoré nie sú veľmi dôležité.

Aby sme mohli použiť aj funkciu aritmetického odčítania, stačí zostaviť podprogram na vytvorenie záporného čísla v zhromažďovači aritmetických modulov - v dvojici

HL. Teda treba vytvoriť dvojkový doplnok k 16 bitovému obsahu HL. Okrem toho žiadame, aby príznak CY indikoval nesprávnu funkciu, keď sa dosiahne nezobraziteľná hodnota. To nastane pri pokuse získať kladnú hodnotu záporného čísla 8000_H, ktorú nemožno v 16 bitovom rozsahu zobraziť. V takomto prípade nech sa nastaví najbližšia zobraziteľná hodnota $7FFF_H$ (nastala chyba len o jednotku) a nech sa chyba hlási cez príznak preplnenia CY=1. Pri zmene znamienka použijeme zmenu každého bitu na opačný s s pričítaním jednotky do najnižšieho rádu:

```
                    ;podprogram na zmenu znamienka obsahu HL
                    ORG    8320H    ;začiatočná adresa
8320 7D   ZNAM:     MOV    A,L      ;daj nižší byte HL
8321 2F             CMA             ;zmeň všetky bity
8322 6F             MOV    L,A      ;vráť nižší byte do L
8323 7C             MOV    A,H      ;daj vyšší byte
8324 F5             PUSH   PSW      ;schovaj pôvodné znamienko
8325 2F             CMA             ;zmeň všetky bity
8326 67             MOV    H,A      ;vráť vyšší byte do H
8327 23             INX    H        ;pričítaj jedničku
8328 F1             POP    PSW      ;obnov pôvodné znamienko
8329 A4             ANA    H        ;rovnaké znamienka pred
                                    ; a po?
832A F0             RP              ;ak nie, tak návrat
832B 2B             DCX    H        ;uprav obsah HL na 7FFFH
832C 37             STC             ;nastav signalizačný prí-
                                    ;znak
832D C9             RET             ;návrat z podprogramu
```

Inštrukciou ANA sa testuje zhodnosť znamienok pred a po operácii zmeny znamienka. Ak sú obidve znamienka záporné, nastal uvedený nepriaznivý prípad a preto treba nastaviť CY=1 a obsah HL upraviť z 8000_H na $7FFF_H$. Ak sú obidve

znamienka kladné, všetko je v poriadku (operandum bola nula). Ak sú znamienka rôzne, operácia dopadla dobre.

Pri aritmetických úkonoch sa vyžaduje rozhodovať o veľkosti dvoch čísel. Inštrukcie porovnania procesora však interpretujú čísla ako celé kladné (8-bitové) a preto treba porovnanie vykonať programovo. Zostavíme modul porovnania dvoch 16 bitových celých čísel so znamienkom tak, že jedno z nich je v dvojici DE a druhé v dvojici HL (podľa doterajších konvencií je to akumulátor). Požadujeme, aby sa nezmenili obidva operandy a príznaky nech sa nastavujú s rešpektovaním znamienok takto:

- obsah DE menší ako obsah HL: CY=0 Z=0
- obsah DE rovnaký ako obsah HL: CY=0 Z=1
- obsah DE väčší ako obsah HL: CY=1 Z=0

```
                    ;modul porovnania 16 bit. čísel so znam.
                    ORG    832EH    ;adresa podprogramu
832E 47     POI6:   MOV    B,A      ;schovaj A do B
832F 7C             MOV    A,H      ;daj vyšší byte "akumulá-
                                    ;tora"
8330 AA             XRA    D        ;je zhodnosť znamienok
                                    ;operand.?
8331 F23883         JP     N        ;ak áno, skoč inde
8334 7A             MOV    A,D      ;vyšší byte 2.operandu
8335 BC             CMP    H        ;por. s vyšším byte 1.ope-
                                    ;randu
8336 78             MOV    A,B      ;obnov A
8337 C9             RET             ;návrat z podprogramu
                    ;nasleduje časť, keď operandy majú
                    ;rovnaké znamienko
8338 7C     N:      MOV    A,H      ;daj vyšší byte "akumulá-
                                    ;tora"
8339 BA             CMP    D        ;porov. s vyšším byte 2.
                                    ;oper.
```

```
833A 78          MOV   A,B     ;preventívne obnov akumu-
                               ;látor
833B C0          RNZ            ;ak rozhodol vyšší byte,
                               ;hotovo
833C 7D          MOV   A,L     ;inak skús porovnať niž-
                               ;šie byte
833D BB          CMP   E       ;oboch operandov
833E 78          MOV   A,B     ;obnov akumulátor
833F C9          RET            ;návrat z podprogramu
```

Najprv treba odlíšiť prípad, keď sú zhodné znamienka operandov. Ak sú, možno ich porovnávať inštrukciou CMP (vlastnosť doplnkového kódu) tak, že najprv sa porovnajú vyššie slabiky. Pri zhode treba ešte porovnať nižšie slabiky. Ak znamienka nie sú zhodné, celkom iste stačí porovnávať len vyšší byte. Komplikácia je v tom, že záporné číslo sa binárne javí väčšie ako kladné. Preto treba porovnávať opačne: nie "H" s "D", ale "D" s "H". Úschova obsahu akumulátora sa v tomto podprograme robí z rovnakých dôvodov ako v predošlom podprograme PRIZ.

Uvedené podprogramy SU16, PRIZ, ZNAM a PO16 tvoria základ súboru aritmetických podprogramov so 16 bitovými celými číslami so znamienkom. Je zrejmé, že možno urobiť podobný systém napr. pre 32 bitové čísla. Pomocou nich je možné vykonávať základné aritmetické operácie a tvoriť podprogramy vyšších matematických funkcií. Doplnkový kód (binárny) je vhodný na využitie bežnej binárnej sčítačky (v mikroprocesore). Pri vstupe a výstupe dát v spolupráci s človekom je vhodná iná sústava čísel. Najlepšie sa hodí zhustený tvar BCD kódovania (dve dekadické číslice v jednom byte). Potom zrejme treba vybudovať aritmetické programové moduly s dekadickou aritmetikou alebo používať prevodníky kódov BCD a doplnkového kódu napr. 16 bitového.

Na overenie funkcie niektorých uvedených programových modulov zostavíme cvičný program na sčítanie a odčítanie 16 bitových čísel. Funkcia programu bude takáto:

- vymaže sa displej
- vstup z klávesnice prvého operandu a jeho zobrazenie vľavo
- vstup druhého operandu a jeho zobrazenie vpravo na displeji
- porovnajú sa veľkosti oboch operandov a väčší z nich sa označí bodkou na displeji
- čaká sa na stlačenie niektorého klávesu (okrem C)
- potom sa urobí rozdiel a súčet operandov a výsledky sa zobrazia na displeji
- po stlačení niektorého klávesu sa cyklicky opakuje od prvého bodu

Pritom nech sa pretečenie výsledku indikuje textom ERR a pri zhodnosti operandov nech sa bodkou neoznačí ani jeden. Stlačenie klávesu "C" vždy bude značiť prechod na začiatok programu /chybne zadané číslo/. Funkciu rozdielu dosiahneme zmenou znamienka operandu (ZNAM), súčet vykonáme sčítavacím podprogramom (SU16) a na porovnanie použijeme posledne uvedený podprogram (PO16).

```
                         ;program na overenie funkcie aritme-
                         ;tických podprogramov SU16, ZNAM,
                         ;PO16
                         ORG   8200H    ;adresa začiatku
8200 CD8702    CP40:     CALL  CLEAR    ;vymaž displej
8203 11F883              LXI   D,83F8H  ;ľavý displej
8206 CD5882              CALL  VST      ;vstup 1.operandu do HL
8209 E5                  PUSH  H        ;schovaj 1. operand
820A CD5882              CALL  VST      ;vstup 2. operandu
```

```
820D D1          POP  D      ;daj 1. operand do DE
820E CD2E83      CALL PO16   ;porovnaj operandy
8211 CA1F82      JZ   GET    ;ak sú rovnaké, preskoč
                             ;bodku
8214 01FB83      LXI B,83FBH ;štvrtý displej zľava
8217 DA1B82      JC   N1     ;ak prvý bol väčší, obskoč
821A 03          INX  B      ;inak daj ukazovateľ na
                             ;piaty displej zľava
                             ;(2.operand)
821B 0A      N1: LDAX B      ;zober kód z piateho disple-
                             ;ja zľava (alebo štvrtého)
821C F680        ORI  80H    ;a doplň ho bodkou (7.bit)
821E 02          STAX B      ;daj kód naspäť na zobraze-
                             ;nie
821F CD3D02  GET: CALL GETKY ;čakaj na stlačenie klávesu
8222 FE17        CPI  17     ;nebol kláves "C" ?
8224 CA0082      JZ   CP40   ;ak bol, všetko od začiatku
8227 E5          PUSH H      ;schovaj operand
8228 D5          PUSH D      ;aj druhý
8229 CD0083      CALL SU16   ;vytvor súčet operandov
822C D23582      JNC  N2     ;ak nebolo pretečenie, po-
                             ;kračuj
822F CDBC00      CALL ERROR  ;vypíš text "Err", ak bolo
8232 C34F82      JMP  K      ;a choď na koniec cyklu
8235 CDD102  N2: CALL DWORD  ;zobraz súčet operandov
8238 D1          POP  D      ;obnov 1. operand
8239 E1          POP  H      ;obnov 2. operand
823A CD2083      CALL ZNAM   ;obráť znamienko 2.operandu
823D CD0083      CALL SU16   ;urob súčet upravených ope-
                             ;randov
8240 D24982      JNC  N3     ;ak nebolo pretečenie, po-
                             ;kračuj
8243 CDBC00      CALL ERROR  ;ak bolo, vypíš "Err"
```



```
8246 C34F82      JMP  K      ;a choď na koniec cyklu
8249 11FF83  N3: LXI  D,83FFH ;pravý displej pre DWD2
824C CDD402      CALL DWD2   ;zobraz rozdiel operandov
824F CD5702  K:  CALL SCAN   ;čakaj na stlačenie klávesu
8252 D24F82      JNC  K      ;ak nebol stlačený, testuj
                             ;znova
8255 C30082      JMP  CP40   ;pokračuj od začiatku
```

Na vstup operandov slúži podprogram VST opísaný ďalej. Bodka na označenie väčšieho operandu sa generuje tak, že sa vezme obraz číslice na displeji (je v pamäťovej bunke ktorá prísluší danému displeju) a doplní sa jednotkou v 7. bite (ním sa kóduje bodka, pozri [1] str. 47). Poloha rozhodovaného displeja sa určuje inštrukciou LXI B na adrese 8214_H resp. INX B na adrese $821A_H$. Dvojica BC pritom slúži ako ukazovateľ na vyberanie a uloženie obrazu číslice. Volanie podprogramu GETKY na adrese $821F_H$ slúži len na čakanie na stlačenie klávesu (pokyn používateľa "počítaj a zobraz výsledok"). Stlačením klávesu "C" sa zruší ďalšia činnosť a pokračuje sa od začiatku. Pri pretečení (SU16 ho hlási v CY) sa použije podprogram ERROR. Podprogramom ZNAM sa obráti znamienko druhého operandu, takže podprogramom DWD2 sa zobrazí rozdiel operandov. Aj tu sa pretečenie výsledku hlási textom "Err". Cyklus končí čakaním na stlačenie klávesu, ktoré tu značí "pokračuj novým zadaním operandov".

Podprogram VST má za úlohu vykonať vstup 16 bitového čísla z klávesnice a priebežne ho zobrazovať na displeji. Pri vstupe sa očakávajú 4 hexadecimálne číslice. Ak sa zadá iný kláves, celé zadávanie sa zruší a prejde sa na začiatok. Celý podprogram VST má štvornásobný cyklus, v ktorom sa zadajú štyri číslice. Do dvojice HL sa štyri bity (jedna číslica) presúvajú pomocou posunu akumulátora (RAL)

a posunu HL (DAD H). To sa deje v podcykle na adrese 8270_H, ktorý je tiež štvornásobný. Na adrese 8272_H sa rozhoduje, či sa do najnižšieho bitu HL má dať jednička alebo nie/podľa pôvodného obsahu akumulátora/:

```
                        ;podprogram VST na vstup operandov
                        ORG  8258H    ;hneď za hlavným progra-
                                      ;mom
8258 0604       VST:    MVI  B,4      ;parameter cyklu VST
825A C5         CIS:    PUSH B        ;radšej ho schováme
825B CD3D02             CALL GETKY    ;daj číslicu z klávesni-
                                      ;ce
825E FE10               CPI  10H      ;bol kláves hexadecimál-
                                      ;ny?
8260 D20082             JNC  CP40     ;ak nie, choď od začiat-
                                      ;ku
8263 E5                 PUSH H        ;schovaj HL kvôli DISPR
8264 CDA602             CALL DISPR    ;zobraz zadanú číslicu
8267 E1                 POP  H        ;obnov HL
8268 13                 INX  D        ;zvýš ukazovateľ disple-
                                      ;ja
8269 13                 INX  D        ;ešte raz, lebo DISPR
                                      ;znížil DE
826A 07                 RLC           ;presuň nižší polbyte
                                      ;do vyššieho
826B 07                 RLC           ;    -"-
826C 07                 RLC           ;    -"-
826D 07                 RLC           ;    -"-
826E 0604               MVI  B,4      ;počet podcyklov (pre-
                                      ;sun bitov)
8270 29         POS:    DAD  H        ;posuň HL o bit doľava
8271 17                 RAL           ;posuň aj A-reg. a daj
                                      ;do CY
8272 D27682             JNC  NUL      ;preskoč, ak bola nula
```



```
8275 23                 INX  H        ;ak nie, daj do nultého bitu "1"
8276 05         NUL:    DCR  B        ;už prebehli štyri bity?
8277 C27082             JNZ  POS      ;ak nie, opakuj podcyklus
827A C1                 POP  B        ;obnov parameter hlav. cyklu
827B 05                 DCR  B        ;už prebehli štyri cykly?
827C C25A82             JNZ  CIS      ;ak nie, opakuj cyklus
827F C9                 RET           ;návrat z podprogramu VST
```

Po spustení programu (od adresy 8200_H) treba zadať 8 hexa-číslic (prvé štyri pre prvý operand). Okamžite na to sa rozsvieti bodka pri väčšom z nich (ak nie sú rovnaké). Po stlačení hociktorého klávesu (okrem "C") sa vykonávajú obidve operácie, takže na ľavom displeji je súčet a v pravom rozdiel v doplnkovom kóde. Týmto kontrolným programom treba vyskúšať všetky typické prípady vzájomného pomeru operandov (polarita, veľkosť).

Na doplnenie základných aritmetických podprogramov uvedieme programový modul na násobenie 16 bitových dvojkových čísel so znamienkom. Podľa zavedenej koncepcie z predošlých programov je dvojica HL vo funkcii akumulátora výsledkov a v dvojici DE je druhý operand. Modul násobenia NA16 vynásobí operandy v DE a HL a výsledok uloží späť do HL. Ak výsledok násobenia nie je zobraziteľný v 16 bitovom rozsahu, treba hlásiť "preplnenie" nastavením príznaku CY. Teda výsledok násobenia očakávame ako 16 bitové číslo so znamienkom. Vyšších 16 bitov výsledku nebudeme brať do úvahy (musia byť nevýznamné, inak sa hlási preplnenie). V programe násobenia najprv zjednotíme znamienka operandov na kladné, potom vykonáme násobenie pripočítavaním jedného operandu s posuvom a nakoniec upravíme správne znamienko výsledku. Pritom operand v DE dvojici sa násobením nemá zmeniť. Takto sme previedli násobenie čísel so znamienkom na násobenie kladných čísel. Algoritmus násobe-

nia kladných čísel objasníme zápisom "ručného" násobenia. Použijeme na ukážku 5-bitové operandy 10111 a 00101, ktoré budeme interpretovať ako kladné čísla. Násobenie vykonáme tak, že vytvoríme násobky ľavého operandu s jednotlivými rádmi pravého operandu a takto získané medzivýsledky sčítame (pri násobení platí distributívny zákon). Tento postup vykonáme prakticky tak, že sčítame postupne posunutý ľavý operand. Počet posunutí a sčítaní je zhodný s počtom bitov pravého operandu.

```
                10111  .  00101
                 10111
                00000
               10111
              00000
             00000
             -------
23 x 5 = 115  001110011
```

Násobenie začneme od najnižšieho bitu pravého operandu. Ak je v niektorom bite pravého operandu nula, namiesto ľavého operandu pripočítame nulu / 00000 /. Takto posúvanie zodpovedá váham jednotlivých bitov pravého operandu a pripočítanie nuly zodpovedá násobeniu nulou v danom ráde pravého operandu. Postup je výhodný v tom, že prvým medzivýsledkom je určeným najnižší bit výsledku, prvým a druhým medzivýsledkom je určený druhý bit výsledku atď. (preto násobíme od najnižšieho bitu pravého operandu).

Vyššie opísaný postup násobenia použijeme v module násobenia tak, že podľa bitov operandu v HL dvojici budeme postupne pripočítavať obsah DE dvojice (druhý operand násobenia) k medzivýsledkom, ktoré budeme udržiavať v BC dvojici (preto ju v tomto príklade budeme nazývať zhro-

maždovačom). Hotové bity výsledku budeme posúvať zľava doprava do dvojice HL. Nenastane pritom kolízia s operandom v HL dvojici, pretože tento budeme priebežne posúvať doprava, aby sme zistili hodnotu ďalšieho násobiaceho bitu tohto operandu. Takto sa využijú registre B, C, D, E, H, L. Akumulátor bude slúžiť ako počítadlo 16 cyklov posúvania a sčítania.

```
                        ;modul na násobenie 16 bitových čísel
                        ORG  8340H  ;hneď za modulom P016
8340 229E83  NA16:  SHLD HL     ;odlož prvý operand do pa-
                                ;máti
8343 7C             MOV  A,H    ;daj znamienkový bit prvé-
                                ;ho operandu
8344 B7             ORA  A      ;je to kladné číslo?
8345 F24B83         JP   N1     ;ak áno, obskoč zmenu zna-
                                ;mienka
8348 CD2083         CALL ZNAM   ;zmeň znamienko v HL
834B EB      N1:    XCHG        ;daj druhý operand (bol
                                ;v DE)
834C 22A083         SHLD DE     ;aj ten odlož do pamäti
834F 7C             MOV  A,H    ;daj znamienkový bit dru-
                                ;hého operandu
8350 B7             ORA  A      ;je to kladné číslo?
8351 F25783         JP   N2     ;ak áno, obskoč zmenu zna-
                                ;mienka
8354 CD2083         CALL ZNAM   ;zmeň znamienko na kladné
8357 3E10    N2:    MVI  A,10H  ;nastav žiadaný počet
                                ;cyklov
8359 010000         LXI  B,0    ;nuluj zhromaždovač medzi-
                                ;výsledkov
                        ;nasleduje cyklus posuvu a sčítavania
835C F5      CYKL:  PUSH PSW    ;schovaj parameter cyklov
```

```
8350 CD8C83          CALL LSBL   ;zisti najnižší bit HL
8360 CA6683          JZ   POS    ;ak je nula, obskoč sčíta-
                                 ;nie
8363 CD9083          CALL PLUS   ;pričítaj DE k BC (zhro-
                                 ;maždov.)
8366 CD9783    POS:  CALL POSB   ;posuň medzivýsl. doprava
                                 ;a najnižší bit (hotový)
                                 ;daj do CY
8369 CD2F02          CALL SHLRC  ;posuň pravý operand dopra-
                     ;va a do najvyššieho bitu daj ďalší
                     ;hotový bit
836C F1              POP  PSW    ;obnov parameter cyklov
836D 3D              DCR  A      ;a dekrementuj ho
836E C25C83          JNZ  CYKL   ;opakuj cyklus 16 krát
8371 EB              XCHG        ;potom obnov operand v DE
8372 2AA083          LHLD DE     ;potom obnov operand v DE
8375 EB              XCHG        ;potom obnov operand v DE
                     ;nasleduje detekcia pretečenia
8376 78              MOV  A,B    ;test či BC = 0
8377 B1              ORA  C      ;test či BC = 0
8378 C28A83          JNZ  ERR    ;vyšších 16 bitov výsled-
                                 ;ku nie je nula
837B B4              ORA  H      ;je najvyšší bit HL nula?
837C FA8A83          JM   ERR    ;ak nie, je to pretečenie
                     ;nasleduje nastavenie znamienka výsled-
                     ;ku
837F 3A9F83          LDA  HL+1   ;daj najvyšší bit prvého
                                 ;operandu
8382 AA              XRA  D      ;je zhodný so znamienkom
                                 ;druhého operandu?
8383 F28983          JP   RET    ;ak áno, výsledok má byť
                                 ;kladný
8386 CD2083          CALL ZNAM   ;inak zmeň znamienko vý-
                                 ;sledku
```



```
8389 C9        RET:  RET         ;návrat z modulu násobenia
838A 37        ERR:  STC         ;ohlás pretečenie výsledku
838B C9              RET         ;a tiež návrat
                     ;nasleduje podprogram testovania najniž-
                     ;šieho bitu HL (pravý operand)
838C 3E01      LSBL: MVI  A,1    ;maska pre najnižší bit
838E A5              ANA  L      ;čo je v najnižšom bite L?
838F C9              RET         ;návrat z podprogramu
                     ;nasleduje podprogram pričítania DE
                     ;k BC
8390 79        PLUS: MOV  A,C    ;daj LSB dvojice BC
8391 83              ADD  E      ;a pričítaj k nemu LSB z DE
8392 4F              MOV  C,A    ;vráť do BC LSB výsledku
                                 ;sčítania
8393 78              MOV  A,B    ;daj MSB dvojice BC
8394 8A              ADC  D      ;pričítaj MSB z DE plus
                                 ;prenos
8395 47              MOV  B,A    ;vráť MSB výsledku sčítania
8396 C9              RET         ;návrat z podprogramu
                     ;nasleduje podprogram posúvania BC
                     ;dvojice
8397 78        POSB: MOV  A,B    ;daj MSB dvojice BC
8398 1F              RAR         ;posuň doprava, použi CY
8399 47              MOV  B,A    ;vráť MSB do BC
839A 79              MOV  A,C    ;daj LSB dvojice BC
839B 1F              RAR         ;posuň doprava, použi CY
839C 4F              MOV  C,A    ;vráť LSB do BC
839D C9              RET         ;návrat z podprogramu
                     ;nasleduje oblasť dát programu
839E           HL:   DS   2      ;pamäťové slovo na ulože-
                                 ;nie HL
83A0           DE:   DS   2      ;pamäťové slovo na ulože-
                                 ;nie DE
```

Na začiatku programu sa pôvodné operandy uložia do pamäti (ešte ich bude treba). Obidva sa pritom upravia na absolútnu hodnotu pomocou testovania najvyššieho bitu a použitím podprogramu ZNAM z cvičného programu CP40. Potom nasleduje 16 násobný cyklus, v ktorom sa operand v DE postupne pripočítava (podľa hodnoty bitov operandu v HL) k zhromažďovaču BC. Pred cyklom sa zhromažďovač vynuluje (medzivýsledky násobenia). Cyklus pozostáva z testovania aktuálneho bitu operandu v HL (podprogram LSBL), pripočítavania operandu v DE k medzivýsledkom (podprogram PLUS) a z posúvania dvojice HL, v ktorej je časť pravého operandu a časť už hotového výsledku. Dvojica HL sa posúva doprava, takže aktuálny bit pravého operandu je vždy v nultom bite L registra. Podľa toho funguje podprogram LSBL. Dvojica DE sa k BC pripočítava podprogramom PLUS v závislosti od nastaveného príznaku Z v podprograme LSBL. Podprogram PLUS sčítava 16 bitové čísla bez znamienka po byteoch. Prenos z nižšieho byte sa zohľadní v inštrukcii ADC na adrese 8394_H. Prenos z vyššieho byte nemôže nastať, pretože obidva sčítance majú v najvyššom bite nulu (v DE je kladné číslo teda s nulou na začiatku a BC bolo pred sčítaním posunuté doprava). Medzivýsledok sa v BC posúva doprava podprogramom POSB. Najprv sa v akumulátore posunie vyšší byte inštrukciou RAR. Pritom pred jej vykonaním je v príznaku CY nula (zabezpečené v podprograme LSBL inštrukciou ANA a v podprograme PLUS tým, že z vyššie uvedených dôvodov nenastáva pri sčítaní prenos). Preto sa najvyšší bit BC pri posune naplní nulou. Najnižší bit B registra sa cez príznak CY prenesie (využitím ďalšej inštrukcie RAR) do registra C. Najnižší bit C registra (ďalší hotový bit výsledku) sa cez príznak CY (nastaví sa po ukončení podprogramu POSB) prenesie do dvojice HL v podprograme monitora SHLRC. Po 16 cykloch

je v dvojici HL celých 16 bitov výsledku. Vyšších 16 bitov výsledku (výsledok násobenia má 32 bitov) je v dvojici BC, podľa dohody ich však nebudeme uvažovať (výsledok nezobraziteľný v 16 bitoch interpretujeme ako pretečenie).

Po násobení obnovíme operand v DE dvojici (nemá sa násobením meniť). Pretečenie výsledku nastane vtedy, ak po násobení ostane v BC dvojici nenulový obsah. To možno zistiť logickým súčtom oboch byteov BC dvojice. Pretečenie nastane aj vtedy, ak najvyšší bit v HL je nenulový (výsledkom má byť kladné číslo z rozsahu 0000_H až $7FFF_H$). Toto zistíme inštrukciou ORA H a testovaním S príznaku (pred vykonaním inštrukcie ORA H je akumulátor iste nulový, lebo pred ňou je inštrukcia JNZ). V oboch uvedených prípadoch treba nastaviť príznak CY (na signalizáciu pretečenia).

Ak nenastalo pretečenie, treba nastaviť správne znamienko výsledku. To sa vykoná operáciou neekvivalencie najvyšších bitov oboch operandov. Pri rôznych znamienkach treba zmeniť znamienko výsledku podprogramom ZNAM. Pri riadnom ukončení podprogramu NA16 je príznak CY iste nulový, pretože buď sa vynuloval inštrukciou XRA na adrese 8382_H alebo sa správne nastavil v podprograme ZNAM.

Poznamenajme, že na začiatku podprogramu NA16 možno umiestniť testovanie oboch operandov na nulu, čím vylúčime násobenie nulou (zvýši sa priemerná rýchlosť výpočtu, nulou netreba násobiť ale priamo nastavíme nulový výsledok). Cyklus posúvania a sčítavania CYKL je zapísaný schválne štrukturálne kvôli prehľadnosti (volanie podprogramov). V tomto prípade totiž podprogramy nie je vhodné použiť, lebo sú volané iba na jednom mieste hlavného programu. Ich telá by mohli byť zaradené priamo do hlavného cyklu, čím by sa urýchlil výpočet a znížili nároky na pamäť (vypadli by inštrukcie CALL a RET).

Študujúcemu čitateľovi odporúčame stanoviť časovú dĺžku trvania jednej operácie násobenia (priemerne).

Vlastnosti zostaveného programového modulu na násobenie overíme cvičným programom CP41, ktorý vynásobí dve 16 bitové čísla so znamienkom zadané z klávesnice. Na vstup operandov použijeme podprogram VST z cvičného programu CP40. Pri pretečení nech sa zobrazí text Err:

```
                        ;cvičný program na overenie funkcie
                        ;NA16
                        ;ORG  8200H    ;začiatok programu
8200 CD8702  CP41:      CALL CLEAR     ;zmaž displej
8203 11F883             LXI  D,83F8H   ;ľavý displej
8206 CD5882             CALL VST       ;vstup prvého operandu
8209 E5                 PUSH H         ;schovaj prvý operand
820A CD5882             CALL VST       ;vstup druhého operandu
820D D1                 POP  D         ;prvý operand daj do DE
820E CD3D02             CALL GETKY     ;daj povel na násobenie
8211 CD4083             CALL NA16      ;stlačením niektorého
                                       ;klávesu
8214 D21D82             JNC ZOBR       ;ak nepretieklo, zobraz
                                       ;výsledok
8217 CDBC00             CALL ERROR     ;a keď áno, napíš "Err"
821A C32082             JMP  K         ;a choď na koniec cyklu
821D CDD102  ZOBR:      CALL DWORD     ;zobraz výsledok na
                                       ;displeji
8220 CD5702  K:         CALL SCAN      ;čakaj na nové zadáva-
8223 D22082             JNC  K         ;nie operandov (stlače-
                                       ;nie klávesu)
8226 C30082             JMP  CP41      ;opakuj cyklus
```

Na začiatku programu sa pomocou podprogramu VST naplnia operandami dvojice DE a HL. Potom sa po stlačení

hociktorého klávesu vykoná násobenie a zobrazí sa výsledok. Pri pretečení sa hlási text Err. Pri použití tohto cvičného programu treba uložiť všetky potrebné podprogramy (aj ZNAM a VST z cvičného programu CP40).

Funkciu násobenia overíme zadávaním dvojíc štvormiestnych hexadecimálnych čísel. Na uľahčenie testovania správnosti funkcie uveďme tieto ekvivalenty:

$5 = 0005_H$, $-5 = FFFB_H$

$6 = 0006_H$, $-6 = FFFA_H$

$30 = 001E_H$, $-30 = FFE2_H$

$1000 = 03E8_H$ $\quad 1000 \times 1000 = 1\,000\,000$

Číslo 1 000 000 už nie je v 16 bitovom rozsahu zobraziteľné (najvyššie zobraziteľné číslo je $7FFF_H = 32\,767$).

Dvojková aritmetika so znamienkom je vhodná na použitie v automatizovaných systémoch použitých na spracovanie veličín z okolia, napr. pri spracovaní analógových veličín v riadiacej sústave. S dvojkovými číslicovými údajmi dobre pracujú aj napr. analógové prevodníky. Dvojkové čísla sú však nie príliš vhodné pri výmene údajov s človekom, ktorý je zvyknutý pracovať s desiatkovými číslami s explicitne vyjadreným znamienkom. Ak je mikropočítač použitý napr. v automatizovanom systéme skladového hospodárstva, výmena informácií s človekom musí byť z dôvodov kvalifikačného zamerania pracovníkov výlučne s desiatkovými číslami. Ak prevažuje desiatkové zobrazenie čísel, je výhodné mať aj aritmetické programové moduly pracujúce s desiatkovými číslami. Kompromisom môže byť súbor modulov na dvojkovú aritmetiku, ktorý obsahuje obojsmerné programové prevodníky medzi desiatkovými a dvojkovými číslami.

Ako príklady aritmetických funkcií procesora zostavíme niekoľko programových modulov na aritmetické výpočty v desiatkovej sústave. Na zobrazenie čísel použijeme zhustené kódovanie BCD v 16 bitovom rozsahu bez znamienka. Použijeme teda štvorciferné kladné desiatkové čísla v pevnej rádovej čiarke. V praxi sa používajú aj iné dekadické zobrazenia (napr. len jedna BCD číslica v jednom byte s kódom znamienka na začiatku). Pre potreby nášho príkladu sa obmedzíme na kladné čísla v zhustenom tvare v rozsahu 16 bitov. Takto obsiahneme číselný rozsah 0000 až 9999.

Základným programovým modulom je podprogram na sčítanie dvoch desiatkových čísel. Dohodnime, že opäť dvojica HL bude vo funkcii akumulátora simulovaného desiatkového procesora a dvojica DE bude obsahovať druhý operand. V module sčítania sa pričíta obsah dvojice DE k dvojici HL, pričom obidva operandy sa interpretujú ako kladné dekadické čísla. Pri sčítaní môže vzniknúť výsledok nezobraziteľný v 16 bitovom rozsahu (viac ako štvorciferné číslo) a tento stav treba signalizovať príznakom CY ako preplnenie. Mikroprocesor 8080 je na takéto výpočty vybavený inštrukciou DAA, ktorou možno po sčítaní desiatkových čísel upraviť obsah akumulátora opäť na desiatkové číslo. Napríklad pri dvojkovom sčítaní (len také inštrukcie má mikroprocesor) byteov 06 + 08 vznikne výsledok 0E. Inštrukcia DAA upraví obsah akumulátora pričítaním 6 do nižšieho polbyte, takže vznikne číslo 14, ktoré pri dekadickom interpretovaní predstavuje správny výsledok. Podrobne je vykonávanie inštrukcie DAA opísané napr. v [3]. Dekadické upravenie výsledku sa opiera o správne nastavené príznaky, ktoré vznikajú pri sčítaní. Sčítavací podprogram SU10 zostavíme takto:


```
                    ;sčítavanie dekadických kladných čísel
                    ;v zhustenom tvare v 16 bitovom rozsahu
                    ORG  8300H  ;začiatok dekad.podprogramov
8300 7D   SU10:     MOV  A,L    ;daj LSB operandu v HL
8301 83             ADD  E      ;pričítaj k nemu LSB z DE
8302 27             DAA         ;uprav dekadický tvar
8303 6F             MOV  L,A    ;vráť LSB výsledku do HL
8304 7C             MOV  A,H    ;daj MSB operandu v HL
8305 8A             ADC  D      ;pričítaj MSB z DE a prípad-
                                ;ne CY
8306 27             DAA         ;uprav dekadický tvar MSB
8307 67             MOV  H,A    ;vráť MSB výsledku do HL
8308 17             RAL         ;daj CY do akumulátora
8309 E601           ANI  1      ;nastav S príznak do nuly
830B 1F             RAR         ;vráť pôvodnú hodnotu do CY
830C C9             RET         ;návrat z SU10
```

Desiatkové sčítanie sa vykoná po byteoch na dva razy, pričom pri druhom sčítaní sa inštrukciou ADC uplatní prípadný prenos z nižšieho byte. Pri obooh sčítaniach sa uplatní dekadická úprava výsledku inštrukciou DAA. Preplnenie pri desiatkovom sčítaní môže vzniknúť pri sčítavaní vyšších byte (MSB). Bude sa signalizovať príznakom CY, ktorý sa nastaví inštrukciou DAA na adrese 8306_H. Okrem toho požadujeme (z dôvodov kompatibility s ďalšími podprogramami desiatkovej aritmetiky), aby sa v tomto podprograme signalizoval vždy kladný výsledok príznakom S (sčítavame kladné čísla). Nastavenie príznaku S = plus dosiahneme inštrukciou ANI, ktorá v najvyššom bite (znamienkovom) vytvorí nulu. Príznak CY si pritom odložíme inštrukciami RAL a RAR do akumulátora tak, aby sa inštrukciou ANI nezmenila jeho hodnota. Takto modul SU10 správne nastavuje príznaky CY, S , ostatné nie sú nastavené správne. Pri

sčítavaní sa obsah operandu v DE nemení.

Pri aritmetických operáciách treba rozhodovať aj podľa toho, či určité číslo je alebo nie je nulové. Zostavme modul na testovanie nulového obsahu simulovaného desiatkového akumulátora - dvojice HL. Príznaky CY a S nastavené v iných podprogramoch by sa nemali zmeniť a príznak Z by sa mal naplniť správnou hodnotou. Modul nazveme PR10:

```
                    ;testovanie HL na nulu
                    ORG  830DH  ;hneď za SU10
830D F5     PR10:   PUSH PSW    ;daj akum. a príznaky do BC
830E C1             POP  B      :daj akum. a príznaky do BC
830F 3E81           MVI  A,81H  :maska pre príznaky CY a S
8311 A1             ANA  C      ;daj do akum. len CY a S
                                ;bit
8312 4F             MOV  C,A    ;do C vráť CY, S a nulový Z
8313 7C             MOV  A,H    ;je aspoň v jednom bite dvo-
8314 B5             ORA  L      ;jice HL jednotka ?
8315 C21C83         JNZ  N      ;ak áno, nechaj Z=0
8318 3E40           MVI  A,40H  ;inak nastav Z=1 v C-re-
831A B1             ORA  C      ;gistri
831B 4F             MOV  C,A    ;inak nastav Z=1 v C-re-
                                ;gistri
831C C5     N:      PUSH B      ;prenes nastavené príznaky
831D F1             POP  PSW    ;akum. späť do PSW
831E C9             RET         ;návrat z PR10
```

V podprograme sa predtým nastavené príznaky udržujú v C registri, obsah akumulátora je v B registri. Príznak Z sa v C registri vynuluje a potom sa skúma, či je to pravda. Ak áno, možno obnoviť akumulátor a register príznakov. Ak je obsah HL nulový, treba ešte predtým nastaviť Z príznak v C registri. Takto sa uchovajú hodnoty

príznakov CY a S a podľa obsahu HL sa správne nastaví Z príznak. Obsah akumulátora sa nezmení, lebo sa uchováva v B registri.

Okrem sčítania sa často požaduje operácia odčítania. V našom prípade treba odčítavať celé desiatkové čísla bez znamienka (kladné). Zostavíme modul na odčítanie operandu v DE (menšiteľ) od obsahu HL (menšenec). Podmienkou opäť je, aby sa operandy interpretovali ako desiatkové čísla. Pri odčítaní nemôže nastať preplnenie (kladné operandy), môže však nastať záporný výsledok. Podľa doterajšej dohody sú všetky čísla kladné a preto nech je výsledkom odčítania absolútna hodnota rozdielu. Znamienko výsledku nech sa signalizuje v S príznaku. Príznak CY je po odčítaní vždy nulový. Príznak Z nebudeme nastavovať, možno ho získať po odčítaní použitím podprogramu PR10. Množina inštrukcií 8080 neumožňuje priamo desiatkovo odčítať, preto využijeme osvedčený postup: pričítame doplnkový kód (desiatkový) menšiteľa. To predpokladá mať k dispozícii desiatkové sčítanie - je ním modul SU10. Odčítavať nulu nebudeme:

```
                    ;desiatkové 16 bitové odčítanie
                    ORG  831FH  ;hneď za PR10
831F EB     RO10:   XCHG        ;zisti, či je v DE (menši-
                                ;teľ)
8320 AF             XRA  A      ;nula a nastav CY=0, S=0
8321 CD0D83         CALL PR10   ;nula a nastav CY=0, S=0
8324 EB             XCHG        ;a nezmeň pritom DE, HL
8325 C8             RZ          ;ak je v DE nula, návrat
8326 D5             PUSH D      ;schovaj menšiteľa (neme-
                                ;niť!)
8327 CD3B83         CALL DO10   ;vytvor v DE desiatkový
                                ;doplnok
832A CD0083         CALL SU10   ;vytvorí sa rozdiel
```

```
832D D1          POP  D      ;obnov menšiteľa
832E D23383      JNC  NEG    ;skoč ak bol záporný výsledok
8331 AF          XRA  A      ;nastav príznak S=plus
8332 C9          RET          ;a návrat z RO10
8333 EB     NEG: XCHG        ;výsledok bol záporný, treba
8334 CD3B83      CALL DO10   ;vytvoriť jeho kladnú hodno-
8337 EB          XCHG        ;tu
8338 F680        ORI  80H    ;nastav príznak S=záporný
833A C9          RET         ;návrat z RO10
```

Na začiatku podprogramu sa modulom PR10 testuje menšiteľ na nulu, pričom je dočasne v HL dvojici. Ak je nulový, odčítavanie sa nebude vykonávať a preto sa inštrukciou XRA A preventívne nastaví CY=0 a S=plus. Podprogramom DO10 sa v DE dvojici vytvorí doplnok menšiteľa, ktorý sa v module SU10 pričíta (desiatkovo!) k menšencu v HL. Pri kladnom výsledku (pritom je vždy CY=1) sa nastaví S=plus a vykoná sa návrat. Pri zápornom výsledku nevznikne pri sčítaní prenos. V takomto prípade sa skočí na návestie NEG, kde sa z výsledku pomocou podprogramu DO10 vytvorí kladné číslo (doplnok k zápornému číslu). Nakoniec sa nastaví príznak S=mínus. Takto sa vytvorí v HL dvojici absolútna hodnota rozdielu, pričom znamienko sa signalizuje príznakom S. Keďže sa všetky čísla interpretujú ako kladné, môžu obidva operandy nadobúdať hodnoty 0000 až 9999.

Vytvorenie desiatkového doplnku k operandu v DE dvojici sa vykoná analogicky ako v dvojkovej sústave: vytvoríme doplnok do čísla 10000_D. Prakticky však treba postupovať inak, nie zmenou každého bitu na opačný a pričítaním jednotky. Číslo 10000_D nevieme v 16 bitovom rozsahu vytvoriť, preto odčítame 9999 - DE a pričítame jedničku. Uvedený rozdiel treba robiť desiatkovo. Desiatkové odčítanie k dispozícii nemáme, preto ho nahradíme šestnástkovým

odčítaním. Nijaká chyba tým nevznikne, lebo v každom ráde sa robí odčítanie 9 - X kde X je číslica z rozsahu 0 až 9 a pre takéto operandy dáva desiatkové a šestnástkové odčítanie formálne ten istý výsledok. Po odčítaní treba k DE pričítať jednotku. Nemožno to jednoducho urobiť inštrukciou INX D, lebo by v DE mohla vzniknúť číslica A (ak by pred inkrementáciou bola v najnižšom ráde deviatka). Preto využijeme služby podprogramu SU10, pričom v HL dvojici pripravíme obsah 0001:

```
                 ;vytvorenie doplnku k DE (dekadicky)
                 ORG  833BH  ;hneď za RO10
833B 3E99   DO10: MVI  A,99H ;doplňovaný byte
833D 93          SUB  E      ;odčítaj LSB operandu v DE
833E 5F          MOV  E,A    ;vráť do DE LSB výsledku
833F 3E99        MVI  A,99H  ;doplňovaný byte
8341 9A          SUB  D      ;odčítaj MSB
8342 57          MOV  D,A    ;vráť do DE MSB výsledku
8343 E5          PUSH H      ;schovaj na chvíľku HL
8344 210100      LXI  H,1    ;priprav jednotku na pričí-
                             ;tanie
8347 CD0083      CALL SU10   ;pričítaj jedničku
834A EB          XCHG        ;a výsledok daj do DE
834B E1          POP  H      ;obnov operand v HL
834C C9          RET         ;návrat z DO10
```

Pri odčítaní 9999 - DE použijeme inštrukciu SUB na sčítanie obidvoch byte, pretože medzi jednotlivými desiatkovými rádmi nevzniká prenos. Pri pričítaní jednotky podprogramom SU10 vznikne prenos len pri tvorení doplnku k nule (0000 - 0 + 1), čo však nevytvorí chybu (výsledkom je opäť nula). Preto vznik prenosu netreba osobitne ošetrovať.

Ďalšou jednoduchou operáciou v desiatkovej aritmetike je porovnanie veľkosti dvoch čísel. Zostavíme modul na takéto porovnanie, pričom jeden operand je v dvojici DE a druhý v "akumulátore" HL. V module nastavíme príznaky Z a CY podľa veľkosti oboch čísel tak, že ak v dvojici HL je väčšie číslo, nastavíme CY=0, Z=0. Ak sú zhodné, nastavíme CY=0, Z=1 a ak je v HL menšie číslo, nastavíme CY=1, Z=0. Pri porovnaní môžeme použiť dvojkové porovnávanie bez znamienka inštrukciou CMP, pretože táto dáva pri dvojkovom porovnaní rovnaké výsledky ako požadujeme. Najprv porovnáme vyššie byte operandov. Ak sa nerozhodne ktoré je väčšie, treba ešte porovnať aj nižšie byte. Obidva operandy v DE a HL sa pritom nezmenia:

```
                    ;desiatkové porovnanie dvoch čísel
                    ORG  834DH   ;hneď za DO10
834D 7C   PO10:     MOV  A,H     ;daj MSB z HL
834E BA             CMP  D       ;je väčší MSB z DE ?
834F C0             RNZ          ;ak sú rôzne, hotovo po-
                                 ;rovnanie
8350 7D             MOV  A,L     ;inak ešte porovnaj LSB
8351 BB             CMP  E       ;LSB oboch operandov
8352 C9             RET          ;a návrat z PO10
```

Podprogram PO10 nastavuje nesprávne príznak S.

Zložitejšou operáciou je násobenie, ktorou doplníme súbor aritmetických modulov desiatkovej aritmetiky. Opäť operandy nech sú v DE a HL a výsledok nech sa uloží do "akumulátora" HL. Podľa doterajších zvyklostí sa operandy interpretujú ako čísla bez znamienka v rozsahu 16 bitov. Výsledok je 32 bitový, uvažovať budeme len 4 platné číslice, t.j. opäť 16 bitový rozsah. Pri násobení nech sa testuje preplnenie (nezobraziteľný výsledok s viac ako

4 číslicami) a hlási v príznaku CY. Výsledok je vždy kladný (kladné operandy) a preto vždy bude S príznak indikovať kladné číslo. Príznak nuly výsledku nebudeme signalizovať (tak ako v moduloch SU10 a RO10). Kvôli prehľadnosti použijeme štruktúrovaný zápis programu.

Pri násobení použijeme algoritmus rozkladu na sčítance podľa distributívneho zákona:

0025 x 0150 = 0025.1000.0 + 0025.100.1 + 0025.10.5 +

+ 0025.1.0

Vidno, že budeme potrebovať tieto operácie: desiatkové sčítanie, násobenie číslami 1, 10, 100, 1000 (=posuv o rád) a násobenie jednomiestnym číslom (číslice pravého operandu). Sčítanie vykonáme známym spôsobom s použitím inštrukcie DAA, posuv o desiatkový rád vykonáme posuvom o štyri bity a násobenie jednomiestnym číslom vykonáme opakovaním pričítaním v cykle:

```
                    :desiatkové násobenie dvoch čísel
                    ORG  8353H   ;hneď za PO10
8353 010000 NA10:   LXI  B,0     ;nuluj dvojicu medzi-
                                 ;výsledkov sčítania
8356 3E04           MVI  A,4     ;bude cyklus 4 sčítaní
8358 F5             PUSH PSW     ;schovaj parameter cyklu
8359 CD7583         CALL DCIS    ;zisti aktuálnu číslicu
                                 ;násobiteľa v HL
835C B7             ORA  A       ;je to nulová číslica?
835D C47983         CNZ  PLUS    ;ak nie, vykonaj pričítanie
                                 ;násobenca v DE k BC
8360 DA7183         JC   ERR     ;preplnenie pri sčítaní
8363 CD8E83         CALL POS4    ;posuň BC a HL o rád do-
                                 ;prava
```

```
8366 F1            POP  PSW    ;obnov parameter cyklu
8367 3D            DCR  A      ;zníž parameter cyklu
8368 C25883        JNZ  CYKL   ;opakuj cyklus
836B 78            MOV  A,B    ;je dvojica
836C B1            ORA  C      ;BC nulová? (vyšších 16 bi-
                               ;tov výsledku)
836D C27383        JNZ  ERR1   ;ak nie, je to preplnenie
8370 C9            RET         ;návrat z NA10 (bez pre-
                               ;pln.)
8371 33       ERR: INX  SP     ;nebolo POP-nuté PSW!
8372 33            INX  SP     ;nebolo POP-nuté PSW!
8373 37            STC         ;signalizuj preplnenie pri
                               ;vykonávaní násobenia
8374 C9            RET         ;návrat pri preplnení
```

V podprograme NA10 slúži dvojica BC ako zhromaždovač medzivýsledkov pri sčítaní dielčích súčinov. Modul má cyklus (prebehne 4 krát), v ktorom sa násobí štyrmi číslicami operandu v HL dvojici. Po každom takomto dielčom násobení sa medzivýsledky v BC posunú o rád (4 bity) doprava a nižšie 4 bity sa presunú do dvojice HL, ktorá sa tiež posunie o rád doprava. Presunuté 4 bity (jeden desiatkový rád) tvoria už časť výsledku a preto po ukončení štyroch cyklov je v HL celý výsledok. Takto je aktuálna číslica násobiteľa vždy v najnižších 4 bitoch HL. Algoritmus je analogicky ako v module NA16 na dvojkové násobenie. Preplnenie výsledku vznikne vtedy, ak nastane preplnenie pri pričítaní DE k BC alebo vtedy, ak po násobení nie je v BC nula (vyššie rády súčinu). To sa v podprograme NA10 testuje a podľa toho sa nastaví príznak CY. Pri normálnom ukončení je CY=0 a S=plus (zabezpečí inštrukcia ORA C na adrese $836C_H$). Ak sa vyskočí z cyklu na návestie ERR, treba posunutím SP kompenzovať neprebehnutú

inštrukciu POP PSW.

Podprogram DCIS má za úlohu zistiť najnižšiu desiatkovú číslicu v HL a uložiť ju do akumulátora:

```
                   ;presun nižšej číslice z HL do akumulá-
                   ;tora
                   ORG  8375H  ;hneď za NA10
8375 7D      DCIS: MOV  A,L    ;daj dve nižšie číslice
8376 E60F          ANI  0FH    ;nechaj len najnižšiu čísli-
                               ;cu
8378 C9            RET         ;návrat z DCIS
```

Podprogram PLUS má desiatkovo pričítať násobenca v DE k medzisúčtom v BC. Obsah akumulátora určuje, koľkokrát má pričítanie prebehnúť (=násobenie najnižšou číslicou v HL). Ak sa pri sčítaní vyskytlo preplnenie z BC, treba to signalizovať príznakom CY.

```
                   ;pričítavanie DE k BC podľa obsahu
                   ;akum.
                   ORG  8379H  ;hneď za DCIS
8379 F5      PLUS: PUSH PSW    ;schovaj počet pričítaní
837A 79            MOV  A,C    ;daj LSB z BC
837B 83            ADD  E      ;pričítaj LSB z DE
837C 27            DAA         ;uprav na desiatkový tvar
837D 4F            MOV  C,A    ;vráť LSB výsledku sčíta-
                               ;nia
837E 78            MOV  A,B    ;daj MSB z BC
837F 8A            ADC  D      ;plus MSB z DE plus CY
                               ;z LSB
8380 27            DAA         ;uprav na desiatkový tvar
8381 47            MOV  B,A    ;vráť MSB výsledku sčítania
8382 DA8B83        JC   PREP   ;skoč pri preplnení
8385 F1            POP  PSW    ;obnov parameter cyklu
```

```
8386 3D           DCR  A      ;zníž parameter cyklu
8387 C27983       JNZ  PLUS   ;opakuj cyklus
8382 C9           RET          ;návrat z PLUS (normálny)
838B 33     PREP: INX  SP     ;nebolo POP-nuté PSW!
838C 33           INX  SP     ;nebolo POP-nuté PSW!
838D C9           RET          ;návrat pri preplnení
```

Podprogram POS4 má za úlohu posunúť dvojicu BC aj HL o desiatkový rád doprava (o 4 bity), pričom najnižšie 4 bity z BC sa majú presunúť do najvyšších 4 bitov HL. Je to totiž ďalšia hotová desiatková číslica výsledku. Na posúvanie HL použijeme monitorový podprogram SHLRC a na presun bitov medzi BC a HL (aj medzi B a C) použijeme príznak CY. Každý jednotlivý posun sa vykoná inštrukciou RAR pomocou akumulátora. Posúvať sa bude po bitoch v cykle, ktorý prebehne 4 krát:

```
                  ;posuv BC a HL o desiatkovú číslicu
                  ;doprava
                  ORG  838EH  ;hneď za PLUS
838E 3E04   POS4: MVI  A,4    ;počet cyklov posunutia
8390 F5     CYPO: PUSH PSW    ;schovaj parameter cyklov
8391 78           MOV  A,B    ;vezmi B a posuň o bit do-
8392 1F           RAR         ;prava, vysunutým bitom
                              ;naplň CY
8393 47           MOV  B,A    ;vráť posunutý B
8394 79           MOV  A,C    ;vezmi C a posuň doprava,
8395 1F           RAR         ;CY do najvyššieho bitu C,
8396 4F           MOV  C,A    ;najnižší bit C do CY
8397 CD2F02       CALL SHLRC  ;posuň HL doprava, použi CY
839A F1           POP  PSW    ;obnov parameter cyklov
839B 3D           DCR  A      ;zníž parameter cyklov
839C C29083       JNZ  CYPO   ;opakuj cyklus posuvu
839F C9           RET         ;návrat z POS4
```


Pri posuve sa najvyšší bit B napĺňa nulou, lebo POS4 nasleduje po inštrukcii JC (takže CY je nulové) a inštrukciou POP PSW na adrese $839A_H$ sa tento stav pred každým ďalším cyklom nastaví znova (inštrukcia DCR A príznak CY nemení).

Uvedený algoritmus násobenia je stredne rýchly a zložitý. Principiálne by totiž bolo možné dekadické násobenie (aj dvojkové) vykonať cyklickým spočítavaním násobenca. Je to však časove náročný postup, pretože v najhoršom prípade by cyklus 16 bitového dekadického sčítania prebehol 9999 krát. V prípade algoritmu z NA10 prebehne najviac 4x9 sčítaní (štyri rády, v každom ráde násobiteľa môže byť až číslica 9). Rozsahom je však takýto podprogram väčší.

Na overenie funkcie podprogramov desiatkovej aritmetiky SU10, RO10, NA10, a PO10 zostavíme cvičný program CP42, v ktorom sa zadajú dva desiatkové operandy a podľa želania sa vykoná ich súčet, rozdiel alebo súčin. Voľbu želanej operácie zadáme z klávesnice stlačením príslušného klávesu takto:

"M" : odčítanie (mínus)
"S" : sčítanie
"N" : násobenie.

Desiatkové štvormiestne operandy zadáme pomocou podprogramu VST z cvičného programu CP40. Pri zadávaní sa zobrazia na displeji. Po zadaní obooh operandov sa podprogramom PO10 rozhodne ktorý je väčší a ten sa označí na displeji bodkou. Potom sa zvolí žiadaná operácia (až po zadaní oboch operandov) a výsledok sa zobrazí na displeji vpravo. Ak vychádza rozdiel záporný, treba výsledok označiť znamienkom mínus (vodorovný segment ľavého displeja). Ak je výsledok nezobraziteľný (preplnenie),

treba zobraziť text "Err". Podprogram pripomína kalkulátor a má pracovať cyklicky. Cvičný program CP42 vychádza dlhší ako CP40, preto by svojím koncom zasahoval do podprogramu VST z cvičného programu CP40. Preto cvičný program CP42 rozdelíme na časť výpočtov a časť zobrazenia výsledku. Časť výpočtov umiestnime na adresu 8200_H a časť zobrazenia za podprogram VST na adresu 8280_H.

```
                        ;overenie funkcie desiatkových arit-
                        ;metických modulov
                        ORG  8200H  ;ukladacia adresa
8200 CD8702  CP42:      CALL CLEAR  ;zmaž displej
8203 11F883             LXI  D,83F8H ;ľavý displej
8206 CD5882             CALL VST    ;vstup prvého operandu
8209 E5                 PUSH H      ;schovaj prvý operand
820A CD5882             CALL VST    ;vstup druhého operandu
820D D1                 POP  D      ;obnov prvý operand
820E EB                 XCHG        ;prvý op. v HL, druhý v
                                    ;DE
820F CD4D83             CALL P010   ;porovnaj operandy
8212 CA2082             JZ   GET    ;ak sú rovnaké, obskoč
8215 01FB83             LXI  B,83FBH ;displej prvého operandu
8218 D21C82             JNC  N1     ;ak je prvý väčší, obskoč
821B 03                 INX  B      ;do BC displej druhého op.
821C 0A         N1:     LDAX B      ;vezmi obraz displeja a
821D F680               ORI  80H    ;doplň ho bodkou (7.bit)
821F 02                 STAX B      ;vráť obraz do displeja
8220 CD3D02     GET:    CALL GETKY  ;vstup klávesu operácie
8223 FE10               CPI  10H    ;bolo "M"?
8225 CA3582             JZ   ROZ    ;ak áno, skoč na rozdiel
8228 FE13               CPI  13H    ;bolo "S"?
822A CA3B82             JZ   SUC    ;ak áno, skoč na súčet
822D FE15               CPI  15H    ;bolo "N"?
```



```
822F CA4182             JZ   NAS    ;ak áno, skoč na násobe-
                                    ;nie
8232 C32082             JMP  GET    ;inak opakuj vstup kláve-
                                    ;su
8235 CD1F83     ROZ:    CALL R010   ;vykonaj odčítanie
8238 C38082             JMP  ZOBR   ;a zobraz výsledok
823B CD0083     SUC:    CALL SU10   ;vykonaj sčítanie
823E C38082             JMP  ZOBR   ;zobraz výsledok
8241 CD5383     NAS:    CALL NA10   ;vykonaj násobenie
8244 C38082             JMP  ZOBR   ;zobraz výsledok
```

Prvý zadaný operand je HL dvojici, druhý v DE dvojici. Dvojica BC ukazuje na ľavý displej (na adrese 8215_H) a ak sa zistí že obsah DE je väčší (CY=1 po P010), tak sa ukazovateľ BC zvýšením premiestni na druhý operand. Displej adresovaný dvojicou BC sa doplní bidkou (7.bit obrazu na displeji). Takto sa označí väčší operand. Potom sa program vetví podľa zadaného klávesu. Ak sa zadá neočakávaný kláves (všetky okrem M, S, N), opakuje sa vstup z klávesnice (program naň nereaguje).

Po prvej časti cvičného programu treba uložiť od adresy 8258_H podprogram VST uvedený v cvičnom príklade CP40. Za ním na adresu 8280_H umiestnime zobrazovaciu časť cvičného programu CP42.

```
                        ;zobrazenie výsledkov z CP42
                        ORG  8280H  ;za VST
8280 DAA182     ZOBR:   JC   ERR    ;ak bolo niekde preplne-
8283 F5                 PUSH PSW    ;nie schovaj S-príznak
8284 E5                 PUSH H      ;schovaj výsledok (kazí
                                    ;CLEAR)
8285 CD8702             CALL CLEAR  ;zmaž displej
8288 11FF83             LXI  D,83FFH;pravý displej
```

```
828B E1              POP  H       ;obnov výsledok
828C CDD402          CALL DWD2    ;zobraz výsledok vpravo
828F F1              POP  PSW     ;obnov S príznak
8290 F29882          JP   K       ;ak je výsledok kladný,
                                  ;koniec
8293 3E40            MVI A,40H    ;inak rozsvieť na ľavom
8295 32FB83          STA 83FBH    ;displeji znamienko "-"
                                  ;(6.bit)
8298 CD5702    K:    CALL SCAN    ;testuj stlačenie klávesu
829B D29882          JNC  K       ;ak nebol stlačený, tes-
                                  ;tuj znova
829E C30082          JMP  CP42    ;všetko od začiatku
82A1 CDBC00    ERR:  CALL ERROR   ;zobraz "Err"
82A4 C39882          JMP  K       ;ako normálny koniec
```

Preplnenie sa v aritmetických moduloch jednotne signalizuje v príznaku CY, čo sa využije pri indikovaní chyby rozsvietením textu "Err". Podľa príznaku S sa nastaví znamienko výsledku tak, že sa rozsvieti 6. bit obrazu displeja s adresou $83FB_H$. Po zobrazení výsledku sa čaká na stlačenie klávesu (ďalšie operandy).

Z uvedených aritmetických programových modulov je zrejmé, že výpočty možno vykonávať úspešbe v dvojkovej aj desiatkovej sústave. Základné aritmetické moduly možno použiť na zostavenie zložitejších funkcií (delenie, umocnenie, trigonometrické funkcie). Pritom sa v niektorých prípadoch bude vyžadovať použiť mierku (v obidvoch prípadoch sme interpretovali celé čísla). Je zrejmé, že analogicky možno zostaviť moduly aj na väčší rozsah ako 16 bitov. Dokonca možno zostaviť moduly na premenlivý počet bytov operandov. Napríklad tak, že operandy sú uložené v pamäti a vstupmi aritmetického modulu sú adresy operandov a ich dĺžka. Študujúcemu čitateľovi, ktorý sa zaoberá

zostavovaním aritmetických modulov, odporúčame na základe skúseností z uvedených programových modulov zapísať a odladiť dvojkový alebo desiatkový modul s jednoduchou funkciou (sčítanie, odčítanie), ktorý pracuje s premenlivou dĺžkou operandov. Vstupom nech sú adresy operandov (napr. v registroch), ich dĺžka (tiež v registri) a výstupom nech je výsledok v dvojici HL a nastavené príznaky (CY pri preplnení, Z pri nulovom výsledku a S ak sa pracuje s číslami so znamienkom). Pre najnáročnejšie aplikácie treba zostaviť programové moduly pracujúce s s pohyblivou rádovou čiarkou s potrebnou dĺžkou mantisy a exponentu.

Ďalšou funkciou aritmetických modulov je prevod z dekadickej sústavy do šestnástkovej a naopak. Takáto činnosť je potrebná vtedy, ak výmena informácií prebieha v inej sústave ako spracovanie údajov. Zostavíme modul na prevod hexadecimálnych čísel v rozsahu 16 bitov do zhusteného dekadického tvaru. Najväčšie zobraziteľné číslo (v 16 bitovom rozsahu) je 9999, preto najväčšie prevádzané číslo môže byť $270F_H$=9999. Pri väčších vstupných číslach treba signalizovať preplnenie výsledku CY príznakom. Vstupné číslo je v HL dvojici a tam treba uložiť aj výsledok prevodu. Použitý algoritmus vysvetlíme na príklade prevodu čísla 1234_H, ktoré vyjadríme súčtom:

$$1234_H = 4.1 + 3.16 + 2.256 + 1.4096 .$$

Je zrejmé, že treba jednotlivými číslicami prevádzaného čísla násobiť dekadické ekvivalenty šestnástkových rádov a tieto násobky dekadicky sčítať.

Prevodový program bude obsahovať cyklus, ktorý prebehne 4 krát. V ňom sa budú sčítavať násobené konštanty 1, 16, 256 a 4096. Tieto budú trvale uložené v tabuľke prevodového podprogramu. V dvojici BC budeme udržiavať

aktuálnu adresu prístupu k tejto tabuľke, dvojica DE bude obsahovať medzisúčty výsledku a z dvojice HL sa budú vyberať jednotlivé šestnástkové číslice. Normálny prevod sa má indikovať hodnotou CY=0. Prevodový modul nazveme PREV a umiestnime ho mimo oblastí, kde sú uložené aritmetické programy z cvičných programov CP40 až CP42. Použijeme ukladaciu adresu 8000_H. Bolo by možné využiť viaceré podprogramy z už uvedených (sčítanie, násobenie), kvôli kontinuite v štúdiu však zostavíme samostatný modul.

```
                    ;prevod hexa-čísel na dekadické (16 bi-
                    ;tov)
                    ORG  8000H   ;ukladacia adresa
8000 3E04   PREV:   MVI  A,4     ;počet cyklov (číslic)
8002 014C80         LXI  B,TAB   ;adresa tabuľky ekvivalen-
                                 ;tov
8005 110000         LXI  D,0     ;nuluj medzisúčet
8008 F5     CYKL:   PUSH PSW     ;schovaj parameter cyklov
8009 CD2580         CALL CIS     ;daj číslicu vstupu z HL
800C CA1580         JZ   POS     ;ak bola nulová, obskoč
800F CD2980         CALL NAS     ;násob číslicou ekvivalent
8012 DA2280         JC   ERR     ;preplnenie
8015 CD4080 POS:    CALL POSH    ;posuň HL o jednu hexa-čís-
                                 ;licu
8018 03             INX  B       ;presuň BC na ďalšiu polož-
                                 ;ku
8019 03             INX  B       ;tabuľky ekvivalentov
801A F1             POP  PSW     ;obnov parameter cyklov
801B 3D             DCR  A       ;zníž parameter cyklov
801C C20880         JNZ  CYKL    ;opakuj cyklus
801F EB             XCHG         ;daj výsledok do HL
8020 B7             ORA  A       ;nuluj CY, normálny koniec
8021 C9             RET          ;návrat z PREV (normálny)
8022 33     ERR:    INX  SP      ;nebolo pop-nuté PSW!
```



```
8023 33             INX  SP      ;nebolo popnuté PSW!
8024 C9             RET          ;návrat pri preplnení
                    ;prenos ďalšej hexa-číslice z HL do
                    ; cum.
8025 7D     CIS:    MOV  A,L     ;daj najnižšiu číslicu z HL
8026 E60F           ANI  0FH     ;a zamaskuj vyššie bity
8028 C9             RET          ;návrat z CIS
                    ;násobenie ekvivalentu z tabuľky čísli-
                    ;cou v akumulátore
8029 F5     NAS:    PUSH PSW     ;schovaj parameter cyklov
                                 ;násobenia
802A 0A             LDAX B       ;vezmi LSB ekvivalentu
802B 83             ADD  E       ;pričítaj LSB medzisúčtov
802C 27             DAA          ;uprav dekadický tvar
802D 5F             MOV  E,A     ;vráť LSB výsledku
802E 03             INX  B       ;MSB ekvivalentu
802F 0A             LDAX B       ;vezmi MSB ekvivalentu
8030 8A             ADC  D       ;pričítaj MSB medzisúčtov +
                                 ; + CY
8031 27             DAA          ;uprav dekadický tvar
8032 57             MOV  D,A     ;vráť MSB výsledku
8033 DA3D80         JC   PREP    ;preplnenie
8036 0B             DCX  B       ;vráť BC na LSB ekvivalentu
8037 F1             POP  PSW     ;obnov parameter cyklov ná-
                                 ;sob.
8038 3D             DCR  A       ;zníž parameter cyklov
8039 C22980         JNZ  NAS     ;opakuj cyklus
803C C9             RET          ;normálny návrat
803D 33     PREP:   INX  SP      ;nebolo POP-nuté PSW
803E 33             INX  SP      ;nebolo POP-nuté PSW
803F C9             RET          :návrat pri preplnení
                    ;posuv HL o štyri bity doprava
8040 3E04   POSH:   MVI  A,4     ;parameter cyklov posuvu
```

```
8042 F5       P1:  PUSH PSW    ;schovaj ho
8043 CD2E02        CALL SHLRZ  ;posuv HL o bit doprava
8046 F1            POP  PSW    ;obnov parameter cyklov
8047 3D            DCR  A      ;zníž ho
8048 C24280        JNZ  P1     ;opakuj cyklus
804B C9            RET         ;návrat z POSH
804C 01001600 TAB: DW   1,16H  ;ekvivalenty šestnástkových
8050 56029640      DW   256H, 4096H ;rádov
```

V tabuľke TAB sú štyri ekvivalenty šestnástkových rádov, ktoré sú vyjadrené v desiatkovej sústave. Tieto sa pripočítavajú podľa hexa-číslic k medzisúčtom v DE podprogramom NAS, ktorý sčítava dekadicky. Dvojica HL sa posúva doprava o 4 bity, takže v jej najnižších 4 bitoch je stále aktuálna číslica prevádzaného čísla. Hlavný program PREV aj jeho podprogramy pracujú podobne ako násobiaci modul NA10. Jeden rozdiel je v tom, že sa pričítavajú hodnoty uložené v pamäti. Sprístupňujú sa ukazovateľom v dvojici BC. Preplnenie sa indikuje vtedy, keď nie je možné zobraziť výsledok v 16 bitoch. To sa zistí pri sčítavaní inštrukciou ADC D resp. DAA na adrese 8031_H.

Na overenie funkcie prevodníka PREV zostavíme cvičný podprogram CP43, ktorý pracuje jednoducho: z klávesnice vstúpi 16 bitové číslo. Toto sa v podprograme PREV interpretuje ako hexadecimálna hodnota a prevedie sa na dekadické číslo, ktoré sa ako výsledok zobrazí na displeji. Ak pri prevode vznikne preplnenie výsledku, zobrazí sa text "Err" :

```
                ;cvičný program na overenie funkcie pod-
                ;programu PREV
                ORG 8200H  ;začiatok
8200 CD4603 CP43: CALL ENTWD ;vstup dvoch byte a zobraze-
                             ;nie
```



```
8203 CD0080        CALL PREV   ;urob prevod
8206 DA1282        JC   ERR    ;pri prevode bolo preplnenie
8209 11FF83        LXI  D,83FFH ;pravý displej
820C CDD402        CALL DWD2   ;zobraz výsledok vpravo
820F C30082        JMP  CP43   ;cyklicky znova
8212 CDBC00 ERR:   CALL ERROR  ;zobraz text ERR
8215 C30082        JMP  CP43   ;cyklicky znova
```

Funkciu možno overiť zadaním až štvor-miestneho hexadecimálneho čísla a stlačením niektorého červeného klávesu okrem RESET (vyžaduje ENTWD). S výhodou možno použiť obojsmernú tabuľku hexadecimálnych a desiatkových kódov, ktorá je uvedená v lit. [3] na str. 109 .

Opačný prevodník z desiatkových čísel na hexadecimálne možno získať z prevodníka PREV pomerne jednoducho. Stačí vykonávať sčítanie v podprograme NAS dvojkovo a vymeniť ekvivalenty v tabuľke TAB takto:

```
                ;tabuľka pre desiatkovo-hexadecimálny
                ;prevod
                ORG  804CH  ;namiesto predošlej tabuľky
804C 01000A00 TAB: DW  1,10    ;ekvivalenty desiatkových
8050 6400E803      DW  100,1000 ;rádov
```

Dvojkové sčítanie dosiahneme vynechaním inštrukcií DAA v podprograme. Pri laborovaní ich možno nahradiť inštrukciami NOP na adresách $802C_H$ a 8031_H. Potom cvičný program CP43 funguje tak, že treba zadať desiatkové štvormiestne číslo a prevodom sa získa a zobrazí hexadecimálny ekvivalent. Pri tomto smere prevodu nemôže vzniknúť preplnenie ($9999_D = 270F_H$).

Ďalšou potrebnou funkciou pri programovaní číslicových vstupov a výstupov s človekom je prevod zo znakového

zobrazenia informácie do číselného a opačne. Napríklad pri vstupe znaku "A" s kódovým slovom (v kóde ASCII) 41 treba interpretovať číslicu A_H, t.j. desať. Prakticky znaky vstupujú zo štítkov, klávesnice displeja, ďalekopisu atď., a vystupujú napr. do tlačiarne, ďalekopisu, displeja atď. Najčastejšie sa v triede mikropočítačov používa kódovanie alfanumerických znakov kódom ASCII (osembitový kód, lit. [3], str. 107). Použitý kód závisí od použitých prídavných zariadení. Pri výmene číselných údajov s takýmito prídavnými zariadeniami treba použiť programové prevodníky medzi vnútorným číselným vyjadrením a vonkajším textovým reťazcom. V nich sa zrejme uplatnia dva podprogramy:

- prevod jedného textového znaku (0 až F) na jednu hexadecimálnu číslicu
- prevod jednej hexadecimálnej číslice na zodpovedajúci textový znak

Zostavíme tri programové moduly, ktoré možno použiť pri tvorení programových prevodníkov znaková reprezentácia - číselná reprezentácia a opačne.

Podprogram TEST má rozhodnúť o tom, či textový znak patrí do súboru hexadecimálnych číslic alebo nie. Výsledok nech sa indikuje v príznaku CY. Prípustné sú numerické znaky Ø až 9 a textové znaky A až F. Pri ostatných znakoch nech sa nastaví CY=1. V kóde ASCII sú preto prípustné kódové slová 3Ø až 39 a 41 až 46. Zrejme treba testovať, či daný kód leží v niektorom z dvoch uvedených intervalov. Testovať budeme inštrukciou CPI s hranicami oboch intervalov:

```
                ;test znaku, či je hexadecimálnou čísli-
                ;cou
                ORG  8054H  ;hneď za PREV
```



```
8054 FE30   TEST:  CPI   30H  ;nie je kód menší ako 30H?
8056 D8            RC         ;ak je menší, CY=1 a ná-
                              ;vrat
8057 FE47          CPI 47H    ;nie je kód väčší ako 47H?
8059 3F            CMC        ;ak je väčší, CY=1
805A               RC         ;a návrat
805B FE3A          CPI  3AH   ;je kód menší ako 3AH?
                              ;(takže leží v intervale
                              ;3Ø až 39)
805D 3F            CMC        ;ak áno, nastav CY=0
805E DO            RNC        ;a návrat
805F FE41          CPI  41H   ;je väčší ako 40H? (takže
                              ;leží v intervale 41 až 46)
8061 C9            RET        ;ak je väčší, nastav CY=0,
                              ;inak CY=1 a návrat
```

V programe TEST sa najprv detekujú kódy pod 30 a nad 46 a potom kódy vnútri dovolených intervalov. Obsah akumulátora, v ktorom sa predpokladá testovaný kód, sa pritom nemení. Podprogram TEST možno využiť na prvú kontrolu textového reťazca reprezentujúceho číslo, či neobsahuje nedovolené znaky.

Podprogramom CSL možno previesť kód jedného znaku na zodpovedajúcu hexadecimálnu číslicu, pokiaľ znak patrí do prípustného súboru znakov Ø až F. Využíva sa pritom zákonitosť kódu ASCII, že číslice Ø až 9 sa líšia od znakovej reprezentácie iba v bitoch č. 4 a 5. Túto súvislosť môžeme interpretovať aj tak, že vzdialenosť kódov a čísel v hodnotovom vyjadrení oboch byte (na číselnej hexadecimálnej osi) je stála a prestavuje hodnotu 30_H. Preto stačí od kódového slova odčítať túto hodnotu a získame jeho číselné vyjadrenie. Napr. kód 35_H patrí znaku "5" a $35_H - 30_H = 5$. Pre znaky A až F platí tá istá zákonitosť s tým rozdielom, že vzdialenosť je až 37_H (v kóde ASCII

je medzi znakmi 9 a A ešte ďalších 7 znakov). Pre tieto znaky treba odčítať okrem uvedených 30_H ešte 7_H :

```
                        ;prevod znakovej na číselnú reprezen-
                        ;táciu
                        ORG  8062H  ;hneď za TEST
8062 D630     CSL:      SUI  30H    ;odčítaj konštantu
8064 FE0A               CPI  0AH    ;bola to číslica 0 až 9?
8066 F8                 RM          ;ak áno, návrat - hotovo
8067 D607               SUI  7      ;inak odčítaj ešte 7
8069 C9                 RET         ;návrat pri znakoch A až
                                    ; F
```

Podprogram CSL tiež očakáva prevádzané kódové slovo v akumulátore, kde necháva aj prevedenú číslicu 00 ÷ 0F. Prípustnosť znakov sa nekontroluje.

Opačný podprogram ZNAK prevádza jednu hexadecimálnu číslicu v akumulátore na zodpovedajúci znak. Vstupná číslica má byť v akumulátore v nižších 4 bitoch, vyššie 4 bity musia byť nulové. Výsledný znak ostáva v akumulátore. Program ZNAK pracuje opačným spôsobom ako podprogram CSL.

```
                        ;prevod jednej číslice na znak kódu
                        ;ASCII
                        ORG  806AH  ;hneď za CSL
806A C630     ZNAK:     ADI  30H    ;pričítaj konštantu
806C FE3A               CPI  3AH    ;bola číslica 0 až 9?
806E F8                 RM          ;ak áno, hotovo a návrat
806F C607               ADI  7      ;inak pričítaj ešte 7
8071 C9                 RET         ;návrat pri číslicíach
                                    ;A až F
```

Funkciu zostavených modulov overíme cvičným príkladom CP44, ktorý dostane z klávesnice kódové slovo ASCII kódu. Kód sa otestuje na prípustnosť modulom TEST. Ak je neprípustný, zobrazí sa text "Err". Ak je prípustný, prevedie sa na hexadecimálnu číslicu a tá sa zobrazí na displeji. Potom sa vykoná spätný prevod na textový znak podprogramom ZNAK a jeho kód sa tiež zobrazí na displeji. Musí byť zhodný so vstupným kódom. Vstup z klávesnice vykonáme podprogramom ENTBY, zobrazenie číslice podprogramom DISPR a zobrazenie kódu znaku podprogramom DBY2. Podprogram SCAN využijeme na testovanie stlačenia klávesu (=vstup ďalšieho prevádzaného kódu).


```
                        ;overenie funkcie podprogramov TEST,
                        ; CSL a ZNAK
                        ORG  8200H  ;ukladacia adresa
8200 CD8702   CP44:     CALL CLEAR  ;zmaž displej
8203 CD3603             CALL ENTBY  ;vstup kódu znaku
8206 7D                 MOV  A,L    ;daj zadaný kód do akumu-
                                    ;látora
8207 CD5480             CALL TEST   ;je to prípustný kód?
820A DA2682             JC   ERR    ;ak nie, zobraz "Err"
820D CD6280             CALL CSL    ;preveď kód na číslicu
8210 11FC83             LXI D,83FCH ;adresa 4. displeja spra-
                                    ;va
8213 CDA602             CALL DISPR  ;zobraz prevedenú číslicu
8216 1B                 DCX  D      ;vynechaj jeden displej
8217 CD6A80             CALL ZNAK   ;preveď späť na znak
821A CD9802             CALL DBY2   ;zobraz prevedený znak
821D CD5702   T:        CALL SCAN   ;je stlačený kláves?
8220 D21D82             JNC  T      ;ak nie, testuj znova
8223 C30082             JMP  CP44   ;cyklicky znova
8226 CDBC00   ERR:      CALL ERROR  ;zobraz text "Err" a
8229 C31D82             JMP  T      ;čakaj na stlačenie kláve-
                                    ;su
```

Program CP44 sa používa tak, že sa zadá jeden byte ukončený červeným klávesom (vlastnosť ENTBY). Program zobrazí príslušnú hexadecimálnu číslicu a kód znaku po spätnom prevode. Nesprávny vstupný kód sa indikuje textom "Err".

14. Doplnenie sortimentu cvičných periférií a technických prostriedkov Školského mikropočítača VÚVT

ŠMS VÚVT je výborným výukovým prostriedkom v oblasti mikropočítačovej techniky. Prax školení s ŠMS VÚVT však ukazuje, že je možné doplniť súbor cvičných periférií a upraviť niektoré obvody ŠMS s cieľom zlepšiť výukové parametre ŠMS.

Výuka so ŠMS je charakterizovaná organizovaním po skupinách (školenia, intenzívne kurzy), ale aj samotným laborovaním (cvičenie, opakovanie). V každom prípade nie sú k dispozícii (okrem výbavy ŠMS) ďalšie technické pomôcky (generátory, meracie prístroje). Súbor cvičných periférií musí mať určitý sortiment z dôvodov metodických (predpokladáme uplatnenie mikropočítača pri riadení alebo sledovaní procesu - riadenie technológie, mnohofunkčné prístroje) a tiež z dôvodov didaktických (práca so ŠMS musí byť pre poslucháča zaujímavá a vnímateľná - prenos informácií medzi mikropočítačom a okolím musí byť pre poslucháča pozorovateľný jeho zmyslami). Takto je totiž možné dosiahnuť, aby bola výuka cielená a efektívna.

Výmena informácií medzi mikropočítačom a jeho okolím je komplikovaná tým, že treba prenášané informácie zmeniť na číslicovú formu (len takú možno preniesť do mikroprocesora). Preto treba riešiť otázku zmeny formy informácie medzi formou číslicovou (kombinácia stavov 0,1) a ostatnými formami.

Prakticky možno konštatovať takéto formy informácií:

- analógová - najčastejšie elektrické napätie
- binárna - binárny signál má stav nula alebo jedna (napr. kontakt zopnutý, rozopnutý)
- impulzná - hodnota informácie je daná počtom impulzov
- frekvenčná - hodnota informácie je daná frekvenciou alebo periódou (impulzov, vĺn)

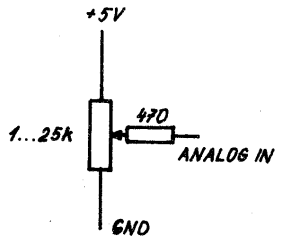
Nezávisle od tohto delenia existujú nosiče informácií (fyzikálne veličiny), ktoré nesú príslušnú formu informácie (elektrické napätie, elektrický prúd, magnetická indukcia, teplota, tlak, osvetlenie atď.). Preto treba pri prenose informácií s mikropočítačom použiť snímače, akčné členy a prevodníky informácií, pričom ani jedny z nich nemenia hodnotu informácie (až na prípadné systematické alebo technické chyby), menia len fyzikálnu veličinu alebo formu informácie. Snímačom a akčným členom prisudzujeme úlohu zmeny fyzikálnej veličiny a prevodníkom zas zmenu formy informácie.

Z uvedeného prehľadu je pochopiteľné, prečo treba vystrojiť ŠMS jednoduchými pomôckami, ktoré umožnia vstup informácie podľa manuálneho zásahu poslucháča resp. registráciu výstupu informácie zmyslami človeka (v príslušnej forme informácie a fyzikálnej veličine). Týmto pomôckam budeme hovoriť cvičné periférie (periférie).

Cvičné periférie majú byť materiálovo a mechanicky nenáročné a majú umožňovať vstup resp. výstup (ak je to možné) informácií vo vymenovaných formách. Zostavíme prehľad možností a odporúčaných cvičných periférií na prenos informácií v rozličných formách:

- vstup analógovej informácie odporúčame robiť s použitím potenciometra, fotoodporu alebo termistora. Potenciome-

ter generuje analógové napätie v závislosti od ručného nastavenia jeho bežca. Vhodné zapojenie je na obr. 14.1.



Obr.14.1

Ochranný odpor v obvode bežca chráni potenciometer pred zničením pri nesprávnom pripojení na kontaktové pole. Perifériou potenciometra možno dosiahnuť vstup analógového napätia v rozsahu 0V až +5V. Upozorňujeme, že sú vhodné lineárne typy potenciometrov. Fotoodporom tiež dosiahneme analógový vstup v závislosti od jeho osvetlenia. Odporúčame použiť typ WK 650 37 v zapojení podľa obr. 14.2. Pritom analógové napätie je (nelineárne) závislé od teploty. Odporúčame použiť niektorý dostupný typ s malou tepelnou kapacitou (t.j. malou hmotnosťou napr. perličkový) a s odporom pri izbovej teplote 1 až 5 kOhm. Pripojenie na kontaktové polia je rovnaké ako pri fotoodpore s pomocným odporočom R s takou hodnotou, ako v predošlom prípade t.j. asi 10 kOhm. Pomocný odpor R v týchto prípadoch nie je kritický. Fotoodpor a termistor však použijeme aj na pripojenie na kontakt EXT 4 alebo EXT 5. Potom žiadame, aby sa podľa okamžitého odporu (závislého od teploty resp. osvetlenia) dosiahol na týchto svorkách raz stab "1", inokedy "0". V tomto prípade je výhodnejšie takéto prepojenie kontaktov:

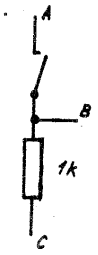
Obr. 14.2

A- GND, B- EXT 4,5, C- +5V. Odpor R preto treba zvoliť taký, aby jeho hodnota vyhovela aj pri analógovom aj diskrétnom vstupe (závisí od použitých prvkov). Žiadame, aby jednoduchým zásahom (zakrytím, dotykom ruky) vznikla vhodná zmena analógového napätia resp. aby sa zmenil logický stav diskrétneho vstupu. Takáto funkcia závisí od parametrov použitého prvku (konkrétneho) a aj od vstupu EXT (integrovaný Schmittov obvod).

- výstup analógovej informácie odporúčame robiť pomocou LED diódy v zapojení podľa obr. 14.3. Prepojenie na kontakty je takéto: A- ANALOG OUT, B- GND. Treba poznamenať, že medzi analógovým napätím a svietivosťou je veľmi nelineárny vzťah zapríčinený charakterom analógového výstupu a parametrami diódy (otváracie napätie je asi 2V). Odporúčame použiť typ LED diódy LQ 190. Inou možnosťou analógového výstupu je použiť elektromotor (súčasť výbavy ŠMS) pripojený na kontakt ANALOG OUT cez optočlánok a zosilňovač tak, ako je to uvedené v čl. 11 v cvičnom programe CP24. Aj tu je však veľmi nelineárna závislosť otáčok motora od analógového napätia. Snáď najdokonalejším riešením je použiť merací ručičkový prístroj (nákladné riešenie).

Obr. 14.3

- vstup binárnej informácie odporúčame robiť pomocou mechanického kontaktu s tlačidlom podľa obr. 14.4. Binárny vstup použijeme pri snímaní zo vstupného portu alebo na prerušenie procesora. V obidvoch prípadoch možno použiť kontakt EXT 4 alebo EXT 5, alebo tiež kontakty iných portov napr. P1A, P2A (len čítanie bez prerušenia). Periférium tlačidla zapojíme na kontakty takto: A- +5V, B- GND. Pri stlačení sa na svorke B dosiahne log. 1.



Obr. 14.4

Upozorňujeme na výhodnú možnosť takto generovať prerušenie (v procesore je "hlavný program" a úlohou je zostaviť obslužný podprogram pre prerušenie od tlačidla, ktorý

sa nejako prejaví napr. iným binárnym výstupom). Iným binárnym vstupom môže byť napr. fotoodpor alebo termistor v zapojení podľa obr. 14.2, ktoré treba zapojiť takto: A- GND, B- EXT 4, C- +5V. Odpor R treba zvoliť s ohľadom na parametre konkrétneho prvku tak, aby sa jednoduchým zásahom (zakrytie, dotyk ruky) dali vytvoriť obidva logické stavy. Odporúčame použiť prvky uvedené pri vstupe analógovej informácie. Pri binárnom vstupe je niekedy vhodné odpor R z obr. 14.2 vynechať (pri vynechaní však hrozia oscilácie pri určitom odpore snímacieho prvku).

- výstup binárnej informácie možno urobiť použitím periférie z obr. 14.3 a pripojením takto: A- +5V, B- MOT CTL - (= bit P1C1) alebo B- T0 (tiež T1). V prvom prípade sa robí výstup informácie z výstupného portu (P1C) a dióda svieti, ak P1C1=0. V druhom prípade sa indikuje stav výstupu T0 resp. T1 (dióda svieti pri jednotkovom výstupe t.j. ak T0, T1 = log. 1). Binárny výstup možno urobiť jednoducho aj pomocou portu P1A a svietiacich diód na základnej doske ŠMS. Iným binárnym výstupom je elektromotor pripojený na P1C1 cez zosilňovač tak, ako je to uvedené v čl. 11 v cvičnom príklade CP25.

- vstup impulznej informácie možno dosiahnuť napr. použitím periférie z obr. 14.4 (dĺžka stlačenia alebo ich počet) a pripojením na niektorý vstupný port, najlepšie cez EXT 4 alebo EXT 5. Dĺžku impulzu možno merať aj pripojením na hradlovací vstup počítadiel (G0, G1) a ich počet pripojením na počítací vstup niektorého počítadla 8253 (pritom treba rozpojiť vodivé prepojky, ktoré sú naznačené v [1] v obr. 2.7 - 5 na str. 82). Vhodné prepojenie periférie z obr. 14.4 je takéto: A- +5V, B- na príslušný vstup, C- GND. Inak je možné použiť impulzný vstup z optočlena elektromotora, ktorý je zakrývaný seg-

mentovým kotúčom na jeho hriadeli. Vhodné prepojenie je takéto: A- +5V, K- GND, E- GND, KOL- príslušný vstup. Ak je v optočlene priehľadný segment, na svorke KOL je log. 0. Aj v tomto prípade je vhodné pripojiť odpor asi 10 kOhm medzi KOL a +5V (kvôli nežiadúcim osciláciám). Pri impulznom vstupe z optočlena elektromotora je vhodné sledovať dĺžku zakrytia jeho fototranzistora.

- impulzný výstup (pre dlhšie časy) možno urobiť tak, ako pri binárnom výstupe.
- frekvenčný vstup možno urobiť pomocou optočlena elektromotora tak, ako pri impulznom vstupe. Nositeľom informácie je frekvencia zakrývania fototranzistora kotúčom na hriadeli.
- frekvenčný výstup je vhodné robiť použitím periférie reproduktora, ktorý je súčasťou ŠMS. Vhodné je zapojiť ho medzi príslušný výstup (P1A, počítadlá) a GND (nižšia hlasitosť) alebo +5V (vyššia hlasitosť). Frekvencie majú byť v akustickom pásme, t.j. od asi 30 Hz do 15 kHz.

K pripojeniu periférií záverom tri poznámky:

- na kontaktovom poli vľavo sú kontakty EXT 4 a EXT 4 OUT, ktoré sú vstupom a výstupom invertora. Možno ho pri pripájaní periférií využiť napr. na inverziu niektorého signálu.
- niektoré vstupy portov (v strede dole) sú neošetrené (citlivé na vyššie napätia, výstupy sú nízkoprúdové), iné sú ošetrené (vľavo, červené), takže majú charakter asi ako TTL (presnejšie pozri v [1]).
- počítacie vstupy počítadiel 8253 sú prístupné po odokrytí plexi-krytu, asi v strede.

Skúsenosť z praxe školení na ŠMS VÚVT ukazuje, že je účelné urobiť niekoľko malých úprav aj na základnej doske

mikropočítača.

Na obr. 2.10-3 v [1] na str. 95 je schéma obvodov na generovanie prerušení monitora, ktoré sa využívajú pri jeho funkciách STEP a BREAKPOINT. Úlohou týchto obvodov je generovať prerušenie po každej používateľskej inštrukcii s typom RST 7. Takto sa riadenie po každej inštrukcii dostane do monitora a tento už zariadi potrebné (napr. ak na túto inštrukciu ukazoval BREAKPOINT s nulovým počítadlom, prejde sa na teplý štart monitora). Prerušenia by však však nemali vznikať po <u>každej</u> inštrukcii, ale len po používateľskej (obslužnú rutinu monitora nechceme krokovať). Preto monitor riadi možnosť prerušenia typu RST 7 vlastným signálom ENTMON, ktorý je počas behu obsluhy prerušenia nulový a nastaví sa na log. 1 až pred opätovným vstupom do programu používateľa. Obslužná rutina prerušenia monitora končí inštrukciami EI, RET, pričom ešte predtým bolo treba nastaviť ENTMON = 1. Je zrejmé, že takto by prerušenie nastalo ešte v inštrukcii RET (vyhodnocuje sa na konci inštrukcie), takže nevykonala by sa ani jedna inštrukcia používateľa a hneď by sa prešlo na obsluhu nového prerušenia. Tento problém sa v ŠMS rieši použitím dvoch preklápacích obvodov MH 7474S (pozícia U30), ktoré sú uvedené na obr. 2.10-3. Takto sa dosiahne, že prerušenie vznikne "oneskorene" až v prvej používateľskej inštrukcii (uplat-(uplatní sa až na jej konci), nie v monitorovej inštrukcii RET (diagramy na obr. 2.10-5 v [1] sú uvedené nesprávne). Použité zapojenie obvodov (je v [1] vysvetlené na str. 94) má tú nevýhodu, že (na obr. 2.10-3 v [1]) dolný preklápací obvod vzorkuje stavový bit M1, pričom sa vyhodnocuje druhým preklápacím obvodom <u>zmena</u> jeho výstupu. Preto sa môže stať, že pri sérii krátkych inštrukcií (jednocyklových) nenastáva zmena výstupu. Dôsledkom je, že ak bol procesor v stave DI (inštrukciou alebo po prerušení) a po-

tom bol inštrukciou EI nastavený na akceptovanie prerušení, monitorové prerušenia sa nebudú generovať dovtedy, kým budú spracovávané len krátke inštrukcie. To nie je v súlade s potrebami používateľa (v stave EI by mali funkcie STEP a BREAKPOINT fungovať vždy!). Túto skutočnosť môžeme overiť krátkym programom:

```
8200 F3      ZAC:  DI    ;nastav DI
8201 3E4A          MVI   A,4AH  ;hociaká inštrukcia
8203 FB            EI           ;obnov stav EI
8204 23            INX   H      ;krátka inštrukcia
8205 80            ADD   B      ;krátka inštrukcia
8206 27            DAA          ;krátka inštrukcia
8207 EB            XCHG         ;krátka inštrukcia
8208 3F            CMC          ;krátka inštrukcia
8209 B1            ORA   C      ;krátka inštrukcia
820A C30082        JMP   ZAC    ;a stále cyklicky
```

Uvedený program nemožno ani krokovať, ani použiť v ňom bod zastavenia programu (breakpoint), hoci väčšia časť programu beží v stave EI.

Kým nemožnosť krokovať v stave DI nie je odstrániteľná (použitý princíp krokovania s využitím prerušenia), uvedený vplyv krátkych inštrukcií možno odstrániť úpravou obvodov na generovanie monitorových prerušení. Úprava je navrhnutá s ohľadom na jednoduchosť a nenáročnosť zásahu. Spočíva v doplnení prídavných obvodov tak, ako to ukazuje obr. 14.5. Preklápací obvod MH 7474S označený U30 je pôvodný, jeho vstup R bol však pripojený na log.1 (cez odpor 1k2 na +5V). Tento vstup R (v [1] na obr. 2.10-3 sú označenia R, S prehodené) treba podľa obr. 14.5 pripojiť na doplňujúce obvody (1x MH 7420, 1x KSY 21 alebo podobný a 2x 4,7 kOhm miniatúrne). Doplňujúce obvody možno umiestniť

na prázdne pole vedľa prepínača STEP/AUTO. Ich úlohou je generovať impulz na začiatku každej používateľskej inštrukcie (ktorej adresa je od 8000_H vyššie). Prepojenie signálov možno urobiť izolovanými vodičmi pod základnou doskou takto:

A_{15} - integrovaný obvod U27, vývod 12
STSTB - integrovaný obvod U29, vývod 10
M1 - integrovaný obvod U38, vývod 29 (uzemniť pájku!)
ENMON - integrovaný obvod U30, vývod 2

Pritom netreba preškrabať žiadne plošné spoje, výstup 8 obvodu MH 7420 možno pripojiť priamo na vstup 1 obvodu U30 (bude ovplyvňovaná aj druhá polovica obvodu U30 - MH 7474, čo nevadí).

Upozorňujeme, že úpravu je možné vykonať len pri dodržaní bezpečnostných opatrení proti zničeniu iných obvodov (vybrať z pätíc všetky obvody, najmä unipolárne, uzemniť pájku a mikropočítač, použiť nízkonapäťovú pájku na integrované obvody). Zásah odporúčame urobiť len na príslušne vybavenom pracovisku a so súhlasom majiteľa (správcu) mikropočítača.

Okrem toho odporúčame vykonať ďalšiu úpravu vo funkcii LED diódy (umiestnená pri EPROM pamäti), ktorá signalizuje stav signálu ENMON. Táto signalizácia nie je pri výuke významná a odporúčame signalizovať radšej výskyt prerušovacieho signálu používateľa INTUSER z obr. 2.10-3 v [1] . Pôvodne je LED dióda pripojená cez odpor 430 Ohm na výstup 4 obvodu U27 (obr. 2.9.-1 v [1]). Odporúčame diódu cez ten istý odpor pripojiť na obvod U9, vývod 8 (pôvodné spojenie treba preškrabať). Takto bude signalizovať výskyt niektorého povoleného prerušenia (krátkotrvajúci periodický výskyt sa prejaví slabým svitom). Podmienky vy-

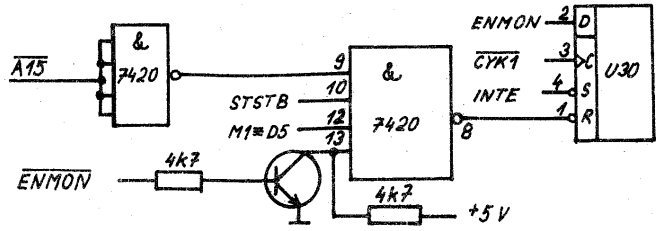
konania úpravy sú tie isté ako v predošlom prípade.

Malú nenáročnú úpravu možno vykonať aj na kontaktovom poli OKP až 6KP vľavo. Pri výuke pri použití motora treba totiž vždy podobne vytvoriť prepojenia pomocou vodičov, čo je z hľadiska výuky práca neefektívna. Odporúčame preto vytvoriť trvalé spojenia (zasunutím krátkych prepojok), ktoré sa využijú vždy. Trvalé môžu byť tieto prepojenia:

MOT SUP+ - +5V
MOT RET - GND

Motor sa potom pripojí na kontakty MOT DRV a MOT RET, riadiaci signál treba priviesť na MOT CTL+ (a P1C1 nastaviť na log. 0) alebo riadiť pomocou P1C1 a MOT CTL+ pripojiť na +5V.

Podobne aj odpor R z obr. 14.2 môže byť trvale zapojený na kontaktové pole, nemusí teda byť súčasťou ovičnej periférie. Podmienkou však je, aby bola vhodná tá istá hodnota odporu pre termistor aj fotoodpor (toto riešenie má aj výhodu - môže byť rôzny odpor pre vstup ANALOG IN a EXT 4).



Obr. 14.5

Použitá a odporúčaná literatúra

[1] Dolejší, A., Gregore P., Horváth, K., Malý, P.: Školský mikropočítačový systém VÚVT, príručka operátora a popis technického vybavenia. Datasystém 1982, Bratislava

[2] Návod na použitie Školského mikropočítačového systému ŠMS VÚVT. Vydal VÚVT Žilina

[3] Partyk, P., Machačka, I.: Základní instrukce mikroprocesoru 8080 - účelová publikácia Tesla Promes 1980

[4] Nohel, J., Jeníček, Č.: Asembler - popis jazyka, řada 8080 - účelová publikácia Tesla Promes v r. 1980

[5] Sobotka, Z., Starý, J.: Mikropočítače - textová a obrazová část. Vydalo ÚTEPS v Tesla VÚST ako internú publikáciu, 1981

[6] Sobotka, Z.; Otázky a odpovede z mikroprocesorov a mikropočítačov - Architektúra a programovanie. ALFA, 1981

[7] Sobotka, Z.: Otázky a odpovede z mikroprocesorov a mikropočítačov - Návrh mikropočítačov. ALFA, 1981

[8] Ďedina, B., Valášek, P.: Mikroprocesory a mikropočítače. SNTL, 1981

O B S A H