PMI-80 jako informační panel

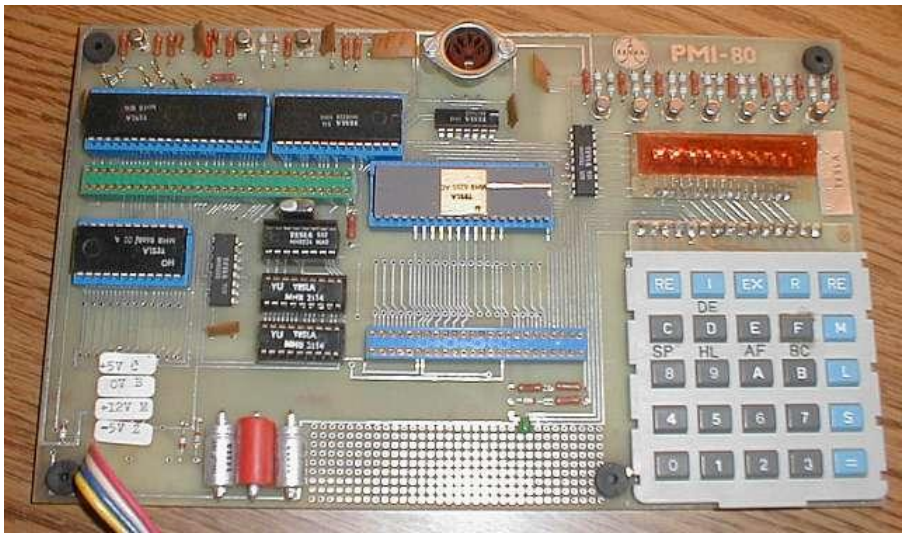
Tento dokument popisuje základy práce s mikropočítačem PMI-80, jeho funkce a ve zkratce i program „Běžící text“.

Verze dokumentu: 1.1
Autor: Blackhead
Datum: 30.3.2004, 7.4.2004

Úvod

Mikropočítač PMI-80 je jedním z vůbec nejjednodušších osmibitových počítačů, které jsem měl možnost si vyzkoušet. Byl vytvořen jako učební pomůcka v době rozvoje výpočetní techniky, kdy byly takové zařízení poměrně na úrovni. Dnes, kdy světu počítačů ve školách vládnou počítače třídy PC, je ale situace s podobnými stroji horší. Na odborné, technické úrovni, se dnes výpočetní technice věnuje jen málo kdo. A to i přes to, že její význam neklesá. Pochopit, jak to vlastně uvnitř počítače funguje, je důležité i dnes. A moderní počítací stroje zase o tolik složitější nejsou. Vše je dnes už jen otázkou programového vybavení. A právě pro Vás je určen tento návod, s nímž se pokusíme společně poodhalit taje této problematiky.

Popis



Kromě základního procesorového bloku s procesorem TESLA MHB 8080, s rychlostí asi 1MHz, obsahuje v základní výbavě paměť RAM 1KB, ROM 1KB se základními obslužnými programy a jeden obvod pro komunikaci – v podstatě paralelní port, jaký známe i z PC. Pro ovládání slouží klávesnice s pětadvaceti tlačítky a devíti místný sedmisegmentový LED displej. Navíc má vstup i výstup na magnetofon, pro ukládání a nahrávání programů na pásku. V první části se budeme zabývat jednoduchým programem, jako příklad jsem vybral program „běžící text“, na němž si ukážeme, jak počítač používat, programovat, zkrátka ovládat. V části druhé se pokusím osvětlit funkci počítače jako takového a podrobně rozebrat kód programu.

Program „Běžící text“

Jediná funkce tohoto programu spočívá v zobrazování jedné textové zprávy, stále dokola, pokud není stisknuta jistá klávesa. Jak ale počítač tomuto „výkonu“ naučit?

Zapnutí počítače

Po připojení počítače k napájení a zapnutí napájecího zroje, se na displeji zobrazí zpráva:

“PMI-80“

Tím nám dává počítač najevo, že je v pořádku a připraven. Nyní ho můžeme začít používat. Než se ale dáme do práce, je tu pro Vás jedna důležitá informace. Na klávesnici je jedno tlačítko, jehož stisknutí neumí počítač zpracovat. Je to tlačítko „RE“. Toto tlačíko funguje jako tlačítko RESET na Vašem PC. Odtud také to označení (REset). Jakmile ho zmáčknete, vše se uvede do stavu po zapnutí. Proto si na něj dávejte pozor. Data v paměti však zůstanou.

Nyní můžeme počítač zapnout. Stiskem libovolné klávesy se potvrdí „úvodní zpráva“ a na displeji se objeví znak podobný otazníku „?“. „Libovolnou klávesou“ se rozumí jakákoli jiná klávesa, kromě „RE“ a „I“. Klávesa „I“ sice Vašemu programu neublíží, tak jako „RE“, ale počítač na ní nebude reagovat. Tedy, stiskněte nějkakou klávesu, jinou než „RE“ a „I“. Klávesou „RE“ by se v tomto případě počítač jen restartoval a ocitl by se ve stejném stavu, tudíž u zprávy “PMI-80“.

Základní funkce

Počítač se nyní nachází v režimu tzv. MONITORu. Je to základní program, dodávaný už s počítačem a je umístěn v ROM paměti. Má několik funkcí, které se vyvolávají těmi modrými tlačítky. Odspodu jsou to tlačítka „=“, pro potvrzení zadávaných hodnot, podobně, jako u PC funguje tlačítko Enter. Tlačítka „S“ a „L“ mají na starost spouštění funkce nahrávání na magnetofon a z magnetofonu do paměti. Jmenují se podle anglických výrazů Save a Load. Save = zachránit, Load = naplnit. Dále je tu tlačítko „M“. To je jedno z těch „důležitejších“ pro Vaši práci. „M“, jako Memory, čili pamět. Tímto tlačítkem se vyvolá funkce pro zadávání bajtů do paměti. Tlačítko „BR“ asi moc používat nebudete, má jméno od „BReak“ a nastavuje se s jeho pomocí tzv. breakpoint – bod zastavení programu. Tlačítkem „R“ se spustí funkce pro zobrazení registrů procesoru, odtud R, jako „Registers“. No, a konečně druhé nejdůležitější tlačítko, tlačítko

„EX“ má za úkol spouštět Vámi napsané programy. „EX“ je zkratka od „EXecute“, čili spustit, nebo provést.

Zadáváme program do paměti

Nebudeme dlouho chodit okolo horké kaše, ukážeme si, jak donutit PMI-80, aby Vás opravdu poslouchalo. Vnutíme mu program a následně ho spustíme. Stiskněte nyní klávesu „M“, aby jsme mohli program zadat do paměti. Na displeji se objeví znak „M“ úplně vlevo a vedle něj náhodná kombinace znaků. Bude to vypadat asi takto: „M 44F8“

Program je třeba zadávat ve strojovém kódu a ještě po jednotlivých bajtech. Ale není se čeho bát, tenhle program je docela krátký. Nejprve musíme vyplnit adresu paměti, na které program bude začínat. Protože s námi tento počítač komunikuje pouze v šestnáctkové soustavě, musíme i tuto adresu zadávat hexadecimálně (šestnáctkově). Budeme používat adresu 1C00. Takže nyní zmáčkněte klávesy “1”, “C”, “0” a ještě jednou “0”. Znaky se budou postupně z pravé strany objevovat na displeji a budou jakoby vytlačovat znaky které tam už byly vlevo, mimo displej. Až na displeji uvidíte “M 1C00”, potvrďte operaci klávesou “=”. Následně se na displeji objeví úplně vpravo dva náhodné znaky, které momentálně obsahuje paměťová buňka na adrese 1C00. Dohromady bude na displeji asi toto: “M 1C00 F9”.

Nyní je čas zadat první bajt našeho programu. Tím bajtem bude číslo 62, čili “3E” šestnáctkově. Stiskněte po sobě tlačítka “3” a “E”, až se na displeji objeví toto: “M 1C00 3E”. Operaci potvrdíme opět tlačítkem “=”, takže se tento bajt skutečně zapíše do paměti počítače. Na displeji se změní adresa paměti na 1C01, protože nyní budeme zadávat další bajt, a také se asi změní dva znaky vpravo na displeji, zobrazující bajt, který je na adrese 1C01. Kromě toho se mezi adresou a bajtem v paměti objeví “=”. Uvidíte asi neco jako: “M 10C1=4E”. Stejným způsobem zadáme i další bajt – “19”. Tedy pomocí tlačítek “1”, “9” a “=”. Ukazatel paměti se posune na další paměťovou buňku. Tj. na 1C02. Tímto způsobem opíšeme do paměti celý program, tak, jak je zapsán níže.

Při zápisu programu dávejte pozor, některé klávesy mohou vlivem “zubu času” reagovat zmateně, jako kdyby jste je stiskli několikrát opakovaně. Proto vždy kontrolujte stav počítače na displeji, před každým zmáčknutím tlačítka “=”.

Po přepsání programu, tedy když zadáte poslední bajt a potvrdíte ho “=”, na displeji se ukáže adresa 1C54. Nyní máme celý program v paměti, ale ještě nemáme nic, co by tento program mohl na displeji zobrazovat. Nyní tedy musíme zadat

nějaká data. Prosím, opište data z odstavce “Výpis dat” pod výpisem programu stejným způsobem, jako program, ale od počáteční adresy 1D00. Tuto akci zahájíte stiskem klávesy “M”. Objeví se chybová hláška. Stiskem kterékoli klávesy hlášku zrušíte, následně opětovným stiskem “M” vstoupíte opět do módu zadávání bajtů. Vyberete adresu 1D00 a zadáte požadovaná data, tak jako jste zadávali program: “=”, “1”, “9”, “=”, atd.

Spuštění programu

Následuje “operace” rozběhnutí programu, okamžik pravdy, kdy se uvidí, jak dobře jste program opsali. Spouštění programu zahájíme stiskem klávesy “EX”, zadáním počáteční adresy (1C00) a potvrzením klávesou “=”. V ten moment se program rozjede a za chvilku se Vámi zadaný text začne posouvat po displeji. Spustíme tedy program: “EX”, “1”, “C”, “0”, “0”, “=”. Program nyní běží. Vidíte text? Ne? Tak to jste někde udělali chybu. Musíte program přepsat z výpisu znovu. Jinak je vše v pořádku.

Ovládání

Ovládáním se v tomto případě myslí změna rychlosti zobrazování, případně změna zobrazovaného textu, pomocí změny dat v paměti počítače. Rychlost se dá měnit změnou hodnoty 40 na adrese 1C38. Menší hodnotou (01 - 3F) se zobrazování zrychlí, větší hodnotou (41 - FF) se zobrazování zpomalí. Změnu textu si můžete vyzkoušet změnou dat pro text. Tyto data jsou uložena v paměti od adresy 1D00. Tak jak je program napsán teď, jsou data uvozena i ukončena několika mezerami - znakem 19. Celková délka dat je i s mezerami 2D = 45 bajtů. Budete-li chtít zadat text delší (max. 256 bajtů), musíte změnit hodnotu 24 na adrese 1C18. Musíte ale vždy zadat ještě o devět bajtů více, protože displej je dlouhý devět znaků.

Podrobný popis funkce

Program funguje jako uzavřená smyčka. Přerušit se dá jen stiskem klávesy “BR”. Celkem jsou v tomto programu tři smyčky, vzájemně v sobě vnořené. V hlavní smyčce se nastavuje (posouvá) text pro zobrazení a volá se druhá smyčka. Ta spolu se smyčkou třetí funguje jako spožďovací smyčka, která zároveň volá funkci pro zobrazení textu a kontrolu klávesnice. V případě stisku klávesy “BR” je program ukončen skokem na adresu 0000, takže se počítač uvede do původního stavu.

Nejprve se do A vloží hodnota 19. Pomocí volání funkce CLEAR (adresa 00AB) vyčistí displej. Funkce CLEAR nejen smaže displej, ale po té na jeho první pozici též umístí znak, jehož kód je při volání funkce v registru A. Kód 19 odpovídá mezeře, tedy prázdnému znaku. Pak se zahájí první smyčka. Nejdřív se naplní registr A nulou, poté se nulový bajt uloží z A do paměti na adresu 1C60. Na této adrese se bude měnit hodnota podle počtu průběhů smyčkou. Následně se zavolá podprogram s dalším cyklem a funkcí pro zobrazení textu. Po návratu do hlavního programu se načte do A hodnota 1C60, kam se obsah A předtím uložil a přesune se do L. Dál se do H přesune 1D. Následuje uložení dat v obou registrech L i H, jako dvojregistr do paměti, na adresu 1FFC (1FFC/1FFD). Na adresách 1FFC a 1FFD je uložena adresa začátku paměti textu pro displej. Jedná se o tzv. výstupní registr zobrazovače (VRZ). Standartně je text displeje uložen v paměti od adresy 1FEF do 1FF7. Takže v 1FFC/1FFD je adresa 1FEF (1FFC=EF, 1FFD= 1F – bajty jsou uloženy v obráceném pořadí). Změna hodnoty v A bude sloužit pro změnu adresy paměti znaků. Když se změní ukazatel na text, bude se na displeji zobrazovat text jiný. Takže posun textu je způsoben posunem ukazatele na adrese 1FFC/1FFD po bloku textu. Dál se k A přičte 01 a zkontroluje se zda už hodnota v A dosáhla 24, což je délka textu + devět znaků délka displeje + nějaký volný prostor (prázdné znaky – mezery) pro vyčištění displeje před dokončením cyklu. Následuje skok s testem na příznaky stavu procesoru JNZ (Jump if Not Zero). Příznaky jsou popsány níže, v dodatcích. Jen ve zkratce – není-li nastaven příznak Z (předchozí operace neměla úspěch) provede se skok na adresu 1C1E, tj. přesočí se nulování registru A. Jestliže je příznak Z zapnut, znamená to, že při předchozí operaci jsme uspěli, A se rovná hodnotě 24, a proto se skok pomocí instrukce JNZ neprovede. Bude tedy následovat vynulování registru A, ukazatel textu se tedy přesune na adresu 1D00 (hodnota v A je spodní bajt adresy 1D00-1D24). Poté se pomocí skokové instrukce vrátí program na začátek smyčky.

Následuje popis druhé, vnitřní smyčky. V úvodu se vynuluje registr A a jeho hodnota se uloží na adresu 1C61. Následně se opět načte do A. Tato zdánlivě zbytečná operace zde má své místo. Zde se totiž začíná program ve smyčce opakovat. Obsah A se přesune do registru B, A se vynuluje a přičte se k němu 01. Tím se zahájí třetí, vnitřní smyčka. V ní se porovná obsah A s nulou (00). V případě, že A není nulové, se provede skok na instrukci INR A (INcRement), která registr A opět navýší o 01. Teprve když touto operací změní registr A stav z FF na 00 se smyčka ukončí, protože při kontrole pomocí CPI 00 (ComPare Immediate), se následný skok neprovede, protože se provádí jen když je příznak Z vypnut. Nyní ale instrukce CPI 00 příznak Z nastavila, takže dojde k výstupu ze smyčky. Následuje přesunutí obsahu registru B zpět do A, tím se v A oěví kontrolní hodnota

druhé smyčky. Přičte se k ní 01 a porovná se s 40 (CPI 40). Registr A se následně uloží na adresu 1C61 a pak se teprve provede podmíněný skok, při Z=0. Jestliže je příznak Z nulový, znamená to že při operaci CPI 40 ještě v registru A nebyla hodnota 40. Proto bude smyčka dál pokračovat. V případě, že v A už je 40, skok se neprovede a následuje další instrukce přímo. Je to instrukce RET (RETurn) pro návrat z podprogramu. Vzhledem k tomu, že byla tato smyčka volána jako podprogram, dojde k návratu do nadřazené, primární smyčky. V případě pokračování vnitřní smyčky následuje volání podprogramu DISP (adresa 0140). V dokumentaci MONITORu při uživatelské příručce, tato funkce není uvedena. Je však v další literatuře uvedena ve výpisu paměti MONITORu. Tato funkce je v podstatě součástí dokumentované funkce OUTKE (adresa 0116). Má za úkol zobrazit na displeji znaky z paměti výstupního registru zobrazovče (VRZ). Standartně je VRZ umístěn od 1FEF do 1FF7. V naší aplikaci se posouvá mezi adresami 1D00 – 1D08 az 1D24 – 1D32. Kromě zobrazování znaků tato funkce zkontroluje zda byla stisknuta nějaká klávesa. Nesnaží se však přečíst její hodnotu do A, tak jak to dělá funkce OUTKE, pouze v případě stisku klávesy nastaví příznak C. Proto v našem programu následuje po volání funkce DISP (0140) instrukce podmíněného skoku při C=1, instrukce JC (Jump if Carry). Tento skok přeskočí návrat na začátek smyčky v případě že je klávesa stisknuta. Není-li, program pokračuje skokem JMP (JuMP) na začátek této druhé smyčky. Při stisku klávesy se ale neprovede, místo toho se provede volání CALL 011C, což je adresa uvnitř procedury OUTKE. OUTKE začíná na adrese 0116, ale hned na začátku má krátkou smyčku, ve které volá pouze funkci DISP a čeká tak až na stisk klávesy. To znamená, že by se náš text posouval po displeji pouze o jeden znak po každém stisknutí nějaké klávesy. Vzhledem k tomu, že my už jsme funkci DISP volali a podle příznaku C také víme zda byla stisknuta nějaká klávesa, můžeme tuto úvodní smyčku přeskočit. Voláme tedy přímo zbytek funkce OUTKE, která nám vrátí kód klávesy v registru A. Následuje porovnání A, jestli je rovno 97, což by znamenalo že byla stisknuta klávesa “BR”. Při následném podmíněném skoku JNZ buď dojde k návratu na začátek smyčky, v případě že byla stisknuta jiná klávesa, nebo se provede, v případě stisku klávesy “BR”, skok JMP na adresu 0000. Tím se program zastaví a počítač se uvede do původního stavu. Na adrese 0000 je umístěn hlavní blok MONITORu. Toto volání funguje skoro stejně jako reset počítače. Po takovémto ukončení běhu programu můžete případně změnit rychlost opakování druhé smyčky, případně měnit zobrazovaný text, jak je to popsáno výše.

Něco málo o technice

Jen zběžně nahlédneme na fungování počítače jako takového. Procesor neumí v podstatě nic víc, než přesouvat bajty mezi pamětí a svými registry, posouvat bajty o bit vlevo, či vpravo, přičítat či odečítat jedničku a nebo posílat bajty na sběrnici vstupů a výstupů. Ale protože je to schopen dělat HODNĚ rychle, dokáže z těchto dílčích operací sestavit i složitější děje. Procesor má celkem šest šestnáctibitových dvojregistrů. Čtyři z nich se používají i jako samostatné osmibitové registry. Jsou to A, to je hlavní registr, tzv. střadač (Acumulator), registry B, C,D, E, H, L a zvláštní registr F, který udržuje údaje o příznacích procesoru (Flags). Registry jsou sdruženy po dvou následovně: BC, DE, HL a AF. Střadač je spojen s příznakovým registrem. Další dva dvojregistry mají zvláštní význam. Jsou to SP a PC. PC je programový čítač (Program Counter).V něm je uložena adresa právě vykonávané instrukce. Po přečtení instrukce a jejích případných parametrů se hodnota (adresa) v PC zvýší o patřičný počet bajtů. SP je ukazatel na zásobník procesoru, uchovává se v něm adresa zásobníku. Ten obsahuje adresy všech podprogramových volání a případně hodnoty dvojregistrů vložené do zásobníku přímo. Mezi instrukcemi JMP, JNZ, JC, JNC, JM, JPO a instrukcemi CALL, CNZ, CC, CNC, CM, CPO je velký rozdíl. Všechny páry (JMP-CALL, JNC-CNC) mají stejné požadavky. JMP stejně jako CALL, nepotřebují k provedení žádné informace. JNC i CNC se provedou jen když se při minulé operaci nenastavil příznak C (Carry). Ale rozdíl je právě mezi JMP a CALL, nebo mezi JNC a CNC v tom co dělají. JMP a JNC provedou pouze změnu PC. Např. když je na adrese 1C21 instrukce JMP 1C0A (PC je nastaven na 1C21), po provedení se do PC přenese hodnota 1C0A a procesor pokračuje instrukcí na 1C0A. Při instrukci CALL se ale kromě změny PC uloží adresa “za instrukcí” do zásobníku procesoru (STACK – odtud SP – Stack pointer – ukazatel na zásobník) a v jeho ukazateli SP se sníží hodnota o dvě. To proto, že v zásobníku jsou data ukládána po dvou bajtech, buď se ukládá do zásobníku adresa, nebo hodnota dvojregistru a jsou ukládána odzhora dolů. Standartně je vrchol zásobníku nastaven na 1FD8. Když potom dojde k ukončení podprogramu, pomocí instrukce RET, vybere se hodnota posledního skoku do podprogramu ze zásobníku, posune se ukazatel SP o dvě hodnoty výše a adresa odkud se pod program spouštěl se přesune do PC (adresa která je až za instrukcí která volala podprogram).

Další funkcí počítače je posílání bajtů na výstupní porty, nebo čtení bajtů z portu vstupního. To se děje pomocí indtrukce OUT, nebo IN. V podstatě dojde k tomu, že počítač pošle tento bajt na nějakou adresu, ale pomocí ovládacích signálu vybere harwarově, místo obvodu paměti, obvod vstupu a výstupu. Počítač PMI-80 má standartně osazen jeden V/V (vstupně-

výstupní) obvod, v našem případě má osazen i druhý, doplňkový. Každý z obvodů má k dispozici tři osmibitové brány. Každá může být nastavena různě pro čtení i zápis. Těmi to V/V obvody je ovládán i displej a klávesnice. Klávesnice je v podstatě matice tlačítek. Funguje to tak, že z jedné strany je klávesnice „buzena“ vždy jedním z devíti vodičů, ty jsou společné pro katody LED zobrazovače. Takže je vždy jen několik tlačítek pod napětím, a záleží na tom, které tlačítko stiskněte. Podle toho se propojí tento jeden vodič s jedním ze tří výstupních vodičů. Následně je kód stisknuté klávesy rozkódován podle bitu na jedné bráně (buzení klávesnice) a aktivovaném bitu na druhé bráně (čtení stavu). Podobně je buzen i displej. Devět vodičů (jsou rozkódovány obvodem MH 1082, ze čtyř výstupů jedné brány) je připojeno na katody displeje. Dalších sedm bitů druhé brány spíná anodové napětí na všech anodách jednotlivých segmentů. Je to opět něco na způsob matice, každý z devíti vodičů je připojen na společnou katodu celé číslovky, anody jsou propojeny pro jednotlivé segmenty všech číslovek.

Z toho vyplývá, že když chceme zobrazit text na displeji musí se data pro jednotlivé segmenty (pro všechny číslovky zároveň) postupně posílat na postupně se přepínající (po jedné) číslovky. Takže když v programu chcete reagovat na klávesy, zatímco se zobrazuje text, máte jedinou možnost. V jednom cyklu neustále dokola volat funkci pro zobrazení a zároveň zjišťovat zda byla stisknuta nějaká klávesa. Tak je tomu i v tomto programu. Normálně lze zobrazit text na displeji i s čekáním na klávesu pomocí funkce OUTKE. Ale text je zobrazen pouze staticky. V tomto případě jsem byl nucen hledat jinou cestu. Proto jsem využil oddělených částí OUTKE, jako je DISP a zbytek OUTKE, od adresy, před kterou OUTKE volá DISP ve smyčce. Samozřejmě, byla by zde možnost napsat si vlastní kód pro obsluhu klávesnice i displeje, ale po važuji takovou práci za zbytečnou.

Výpis programu

Následuje výpis programu v asembleru, tj. jazyku symbolických kódů. Každá instrukce je v paměti uložena jako bajt instrukčního kódu, který, podle typu instrukce, může (ale nemusí) následovat jeden či dva bajty dat (adresa či přímo data). Tento výpis je zapsán s ohledem na jednotlivé instrukce.

Adresa	Instrukce	Bajty v paměti	Význam / funkce
1C00	MVI A,19	**3E 19**	Registr A = 19
1C02	CALL 00AB	**CD AB 00**	Zavolá podprogram na adrese 00AB (CLEAR)
1C05	MVI A,00	**3E 00**	Registr A = 00
1C07	STA 1C60	**32 60 1C**	Obsah A uloží na adresu 1C60
1C0A	CALL 1C24	**CD 24 1C**	Zavolá podprogram na adrese 1C24 (smyčka)
1C0D	LDA 1C60	**3A 60 1C**	Uloží do A hodnotu adresy 1C60
1C10	MOV L,A	**6F**	Přesune A do L
1C11	MVI H,1D	**26 1D**	Registr H = 1D
1C13	SHLD 1FFC	**22 FC 1F**	Uloží hodnotu ve dvojregistru HL na adresu 1FFC
1C16	INR A	**3C**	Zvýší hodnotu A o jedničku
1C17	CPI 24	**FE 24**	Ptá se jestli A = 24
1C19	JNZ 1C1E	**C2 1E 1C**	Jestliže příznak Z = 0 skoč na další blok
1C1C	MVI A,00	**3E 00**	Jinak: A = 00
1C1E	STA 1C60	**32 60 1C**	Uloží A na adresu 1C60
1C21	JMP 1C0A	**C3 0A 1C**	Skok na začátek cyklu
1C24	MVI A,00	**3E 00**	A = 00
1C26	STA 1C61	**32 61 1C**	1C61 = A
1C29	LDA 1C61	**3A 61 1C**	A = 1C61
1C2C	MOV B,A	**47**	B = A
1C2D	MVI A,00	**3E 00**	A = 00
1C2F	INR A	**3C**	A = A + 01
1C30	CPI 00	**FE 00**	Ptá se jestli A = 00
1C32	JNZ 1C2F	**C2 2F 1C**	Když příznak Z = 0 skoč zpět (vnitřní cyklus)
1C35	MOV A,B	**78**	A = B
1C36	INR A	**3C**	A = A + 01
1C37	CPI 40	**FE 40**	Ptá se jestli A = 40
1C39	STA 1C61	**32 61 1C**	1C61 = A
1C3C	JNZ 1C40	**C2 40 1C**	Když příznak Z = 0 skoč dál
1C3F	RET	**C9**	Jinak návrat do hlavní smyčky
1C40	CALL 0140	**CD 40 01**	Zavolá podprogram na adrese 0140 (DISP)
1C43	JC 1C49	**DA 49 1C**	Když příznak C = 1 skoč dál
1C46	JMP 1C29	**C3 29 1C**	Jinak skoč zpět (vnější cyklus)
1C49	CALL 011C	**CD 1C 01**	Zavolá podprogram na adrese 011C („OUTKE2“)
1C4C	CPI 97	**FE 97**	Ptá se jestli A = 97 (klávesa „BR“)
1C4E	JNZ 1C29	**C2 29 1C**	Když příznak Z = 0 skoč zpět (vnější cyklus)
1C51	JMP 0000	**C3 00 00**	Jinak skoč na 0000 (vstup do MONITORu = reset)

Následuje výpis programu jen „po bajtech“, snad se Vám bude takto opisovat snáze.

```
1C00 3E 19 CD AB 00 3E 00 32
1C08 60 1C CD 24 1C 3A 60 1C
1C10 6F 26 1D 22 FC 1F 3C FE
1C18 24 C2 1E 1C 3E 00 32 60
1C20 1C C3 0A 1C 3E 00 32 61
1C28 1C 3A 61 1C 47 3E 00 3C
1C30 FE 00 C2 2F 1C 78 3C FE
1C38 40 32 61 1C C2 40 1C C9
1C40 CD 40 01 DA 49 1C C3 29
1C48 1C CD 1C 01 FE 97 C2 29
1C50 1C C3 00 00
```

Výpis dat

Následuje výpis dat textu pro náš program. Opište všechna data po bajtech do paměti od adresy 1D00.

```
1D00 19 19 19 19 19 19 19 19
1D08 19 19 19 19 19 19 19 19
1D10 0A 1A 11 21 19 21 0A 19
1D18 21 05 0E 16 19 13 16 01
1D20 1F 08 00 19 19 19 19 19
1D28 19 19 19 19 19 19 19 19
```

Dodatky

Tabulka kódů kláves

„0“ – „F“ = 80 - 8F
„=“ = 90
„EX“ = 91
„M“ = 92
„L“ = 93
„S“ = 94
„BR“ = 97
„R“ = 9A
(klávesy RE a I jsou řešeny přímo na hardwarové úrovni)

Tabulka znaků pro displej

A=0A	F=0F	K=85	P=13	U=15	“=“=18	1=01	6=06
B=0B	G=20	L=14	Q=43	V=8E	“-“=1F	2=02	7=07
C=0C	H=1A	M=16	R=12	X=9B	“““=1E	3=03	8=08
D=0D	I=01	N=1B	S=05	Y=23	“,“=22	4=04	9=09
E=0E	J=21	O=11	T=10	Z=02	“?“=1D	5=05	0=00

“ “=19 (Mezera – prázdný znak)

Příznaky stavového registru

Bity:	Význam:
0	Příznak C(Y) - přenos
1	Nastaveno na log 1
2	Lichá parita
3	Nastaveno na log 0
4	Příznak AC - pomocný přenos
5	Nastaveno na log 0
6	Příznak Z - nula
7	Příznak S - znaménka

Příznak S

Reaguje na nejvyšší (sedmý) bit střadače. Je-li např. ve střadači výsledek po operaci 01100011 převede se 0 z bitu sedm do bitu sedm stavového registru (0xxxxxxx). Je-li ve střadači 10001001 do sedmého bitu se převede 1 (1xxxxxxx). U takzvaného doplňkového kódu se sedmý bit používá k zaznamenání kladného či záporného čísla. T.j. od 0 do 127 a od -1 do -128.

Binární hodnota:	Decimální hodnota:
00000000	0
00000001	1
01111111	127
10000000	-1
11111111	-128

Odtud název bitu S jako Sign - znaménko (+/-).

Příznak Z

Reaguje na nulu ve střadači. Je-li tedy výsledek po operaci 00000000 (např. při porovnávání), pak se do šestého bitu stavového registru zapíše 1. Je-li výsledek ve střadači nenulový (t.j. např. 00101010) je bit šest nastaven na 0. T.j. Z=0. Název Z podle významu: Zero – nula.

Příznak AC

Toto je příznak čtyřbitového přenosu. V případě že dojde k přenosu z bitu tři do bitu čtyři střadače, nastaví se AC na 1. (Nezapomeň: bit tři je poslední z první čtveřice!) Tedy, je-li ve střadači 00001111 a přičteme-li ke střadači jedničku, bude AC nastaven na 1, protože došlo k přetečení hodnoty přes 15 (Max. hodnota vyjádřitelná čtyřmi bity je 15). Ve střadači bude 00010000. Název registru AC od "Auxiliary Carry" – pomocný přenos.

Příznak P

Příznak liché parity se nastavuje, jestli že je počet jedniček ve výrazu sudý. Lichá parita zaručuje že počet jedniček v osmi bitech střadače a jednom bitu stavového reg. bude lichý. Bude-li ve střadači číslo 243, t.j. 11110011 (počet jedniček je sudý), bit dva stavového registru bude mít hodnotu 1, aby tak doplnil počet jedniček na lichý. Bude-li ve střadači číslo 2, t.j. 00000010, počet jedniček je lichý, bude příznak P=0. P jako Parity - parita.

Příznak CY

Tento bit se nastaví na 1, dojde-li k přetečení ze sedmého bitu střadače. T.j.: Střadač obsahuje číslo 255, 11111111 binárně, a přičteme-li k němu 1, bude střadač obsahovat 0 a příznak CY se nastaví na 1. Jméno CY (někdy jen C), má od anglického "CarrY" – přenos, převod.

Tabulka nastavení příznaků při logických opracích

Operace:	Z	N	CY
R < O	0	0	1
R = O	1	0	1
R > O	0	1	0

Podle této tabulky se nastavují příznaky stavového registru při operacích, jako třeba porovnávání (CPI - ComPare Immediate - porovnání přímé, t.j. A se porovnává přímo se zadaným číslem, operandem; CPI 80 porovná A a $80). Rovná-li se číslo v registru R výrazu (operandu) O, pak se například příznak Z nastaví na 1. Při výrazu CPI 65 se porovná registr A (střadač) s operandem $65. Toto číslo je hexadecimální - šestnáctkové. Takže se A vlastně porovnává s číslem 101.

Závěr

Shrnutí

V případě že máte zájem se o PMI-80 dozvědět více, nahlédněte do technické dokumentace k tomuto stroji dodávané. Je sice ve slovenštině, ale to by Vám až takový problém dělat nemělo. Je v ní popsáno jak fyzické řešení celého počítače, tak jednotlivé komponenty, jako třeba procesor, řadič sběrnice, vstupně-výstupní obvody, ale i kompletní výpis paměti ROM, obsahující všechny procedury a také nepřeberné množství jiných užitečných příkladů pro programování.

Jakékoli náměty, připomínky, dotazy, nebo problémy posílejte na blackhead@blackhead.cz.

Hodně štěstí s PMI-80 přeje

Blackhead