Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

2 INSTRUKČNÍ SOUBOR MCS51

Dříve než se začneme zabývat konkrétním popisem jednotlivých instrukcí mikroprocesoru, je nutné nejprve vysvětlit, co to vlastně instrukce je, jak je tvořena a jak ji v programu zapisujeme.

☞ *V této části knihy se nejdříve seznámíme se způsobem zápisu jednotlivých instrukcí jazyka ASSEMBLER a jejich rozdělením z hlediska funkce v programu. Dále si vysvětlíme pojem adresový prostor a popíšu jednotlivé adresové prostory, které u MCS51 z hlediska programování existují. Na závěr uvedu význam symbolů použitých v detailním popisu instrukcí včetně objasnění významu jednotlivých bitů stavového registru **PSW** a vzápětí bude následovat detailní popis jednotlivých instrukcí jazyka ASSEMBLER procesoru MCS51.*

Detailní popis instrukcí je systematicky uspořádán tak, že jednomu kódu instrukce je v knize přidělena jedna stránka, kde bude uveden její úplný popis včetně příkladů jejího typického použití v programu. Konkrétní popis bude obsahovat kód instrukce, varianty operandu, varianty schematického popisu operace instrukce v závislosti na operandu, výčet ovlivněných příznaků registru ***PSW*** a varianty kódu instrukce. Dále bude následovat slovní popis instrukce a na závěr uvedu počet kódových slabik, kterými je instrukce tvořena, a počet strojových cyklů potřebných k vykonání popisované instrukce. Existují-li pro daný kód instrukce různé varianty operandu, budou pak seřazeny tabulkovým způsobem pod sebe.

Instrukční soubor mikroprocesoru je nezbytně nutné dokonale znát, chceme-li se s mikroprocesorem úspěšně domluvit. Je to podobné jako když se chcete domluvit s druhým člověkem v cizím jazyce, také musíte především ovládat slovní zásobu.

2.1 Zápis a kategorie instrukcí MCS51

Nyní již přistoupím k vysvětlení konstrukce popisu obecné instrukce v jazyce ASSEMBLER. Zápis instrukce se skládá ze dvou částí, a to z kódového slova a operandu. Kódové slovo instrukce určuje, jaká operace se bude vykonávat a v operandu se říká, kde a s jakými daty se daná operace vykoná, popř. kam se její výsledek uloží. Příkladem zápisu instrukce je

MOVC A,@A+DPTR

kde MOVC je kódové slovo instrukce a v tomto případě nám říká, že půjde o operaci přesunu dat z kódové paměti. Druhá část instrukce je tvořena zápisem A,@A+DPTR, který nám dává informaci o tom, že do střadače (registr ***A***) bude uložena hodnota z kódové paměti, jejíž adresa je dána součtem hodnoty registru ***DPTR*** a střadače před vykonáním této instrukce.

V našem případě výraz **@A+DPTR** je **parametr operandu** a ***A*** je **cíl operandu**. Parametr operandu nám říká s jakými daty se bude operace provádět a cíl operandu nám říká, kam se výsledek operace uloží. Toto je klasický příklad nepřímého adresování při přesunu dat mezi procesorem a pamětí.

Jednotlivé instrukce můžeme z hlediska jejich funkce rozdělit do čtyř funkčních skupin:

- Přesuny dat.
- Aritmetické operace.
- Logické operace.
- Operace pro řízení běhu programu (skoky).

Přehledný seznam kódových slov instrukcí seřazených podle funkčnosti nalezneme v *tab. 1*.

Pojmem **přesuny dat** máme na mysli obecné přesuny dat mezi libovolnými buňkami procesoru, přesuny které se týkají střadače a přesuny adresy do cílového operandu. Většina těchto instrukcí neovlivňuje příznaky ve stavovém slově /**PSW**/.

Aritmetickými operacemi máme na mysli čtyři základní matematické operace. Jsou to sčítání, odečítání, násobení a dělení. Přímo instrukcemi se provádí pouze 8bito-

vé operace, a to ještě bez znaménka. Tato kategorie instrukcí většinou ovlivní po svém vykonání příznaky v registru ***PSW***. Je to logické, protože většina těchto operací se provádí ve střadači (registr ***A***) za spolupráce pomocného registru ***B***. Stav registru ***PSW*** je přímo svázán s událostmi a stavy ve střadači.

Tab. 1

přesuny dat	aritmetické	logické	skoky
MOV	ADD	ANL	ACALL
MOVC	ADDC	CLR	AJMP
MOVX	DA	CPL	CALL
POP	DEC	ORL	CJNE
PUSH	DIV	RL	DJNZ
XCH	INC	RLC	JB
XCHD	MUL	RR	JBC
	SUBB	RRC	JC
		SETB	JMP
		SWAP	JNB
		XRL	JNC
			JNZ
			JZ
			LCALL
			LJMP
			RET
			RETI
			SJMP

Logickými operacemi máme na mysli operace jak s jednotlivými bity, pro která platí obecná pravidla Booleovské algebry, tak s celými byte. Jde především o logické součty a součiny a popřípadě posuny jednotlivých bitů registru doleva či doprava nebo záměna vrchních a spodních 4 bitů v instrukci specifikovaného registru.

Poslední funkční kategorií jsou **operace pro řízení běhu programu**. Jde především o přímé nepodmíněné skoky či volání podprogramu a návraty z tohoto podprogramu. Dále jsou to podmíněné skoky, pro jejichž vykonání musí být splněna zvolená podmínka např. je-li příznak přenosu carry v úrovni 1, potom skoč na adresu … Zvláštním případem řízení programu je volání přerušení a návrat z podprogramu pro obsluhu přerušení. Jak již bylo zmíněno, přerušení je voláno hardwarovým voláním instrukce CALL (instrukce pro volání podprogramu) a tato hardwarově volaná instrukce má v adresovém prostoru kódové paměti definovánu pevnou adresu. Tuto adresu nazýváme vektorem přerušení. Skok na obsluhu přerušení je svázán s definovanou hardwarovou událostí v procesoru (např. změna logické úrovně na vstupu INT0). Podrobně se, jak již bylo řečeno dříve, budeme přerušením a jeho konfigurací zabývat při objasňování řešení konkrétního problému v následujících příkladech.

2.2 Adresový prostor MCS51

Adresový prostor je ve výpočetní technice považován za jeden ze základních pojmů. Proto je účelné se o něm zmínit i v této publikaci. Adresový prostor je vlastně z hlediska programování číselný interval (rozsah adres), kde u daného typu mikroprocesoru je hardwarově realizována paměť kódu, dat nebo řídicí registry periferií mikroprocesoru. Pro nás to vlastně znamená, že u daného typu procesoru můžeme tyto, a jen tyto, definované adresy použít pro operace zápisu a čtení dat, protože tyto adresy v hardwaru procesoru fyzicky existují. Jednotlivé adresy procesoru se mohou lišit strukturou svých dat (šířka slova) a zároveň se mohou lišit i fyzickým umístěním v hardwaru procesoru, a tím i způsobem a rychlostí přístupu k těmto datům. Toto je velmi důležité mít na zřeteli při úvahách o správném rozvržení jednotlivých proměnných v programu, aby náš program pracoval co nejrychleji. Budeme-li mít například proměnnou, která bude často používána procedurou pro obsluhu přerušení a toto přerušení bude ještě navíc časté, určitě nebude rozumné tuto proměnnou umístit do externího adresového prostoru dat, protože přístup k těmto datům je složitější a zároveň pomalejší než přístup k datům v interní paměti procesoru. U mikroprocesoru MCS51 existuje pět typů adresových prostorů.

Jsou to:

- DATA
- IDATA
- XDATA
- BIT
- CODE.

Nyní se konkrétně zmíníme o jednotlivých adresových prostorech mikroprocesoru MCS51.

Jako první zmíníme adresový prostor **DATA**. Tento adresový prostor je u MCS51 realizován ve vnitřní paměti dat mikroprocesoru a je definován na intervalu adres <0 – 7FH>. Do tohoto adresového prostoru je možno vstupovat přímo i nepřímo, a to tím nejrychlejším způsobem jaký je u MCS51 možný. V tomto adresovém prostoru doporučuji definovat lokální proměnné, parametry výpočtů, atd., zjednodušeně řečeno proměnné, do kterých se přistupuje velmi často, a to z důvodu aby nebyl příliš vyčerpáván strojový výpočetní čas pro realizaci jednotlivých přesunů dat.

Jako další existuje u MCS51 adresový prostor **IDATA**. Tento adresový prostor je taktéž realizován ve vnitřní paměti dat procesoru, ale je možné do něj přistupovat pouze nepřímo, tzn. složitěji. Adresový prostor IDATA je definován na intervalu adres <0 – 0FFH>. Z předchozího je zřejmé, že se adresové prostory DATA a IDATA překrývají, přičemž prostor IDATA je dvojnásobný (může být i větší a to podle typu použitého procesoru). Do adresového prostoru IDATA a zvláště od adresy 80H (od které existuje pouze prostor IDATA) doporučuji umísťovat proměnné typu pole. Důvod je jednoduchý, protože do proměnné typu pole, která má vždy parametr, vstupujeme většinou nepřímo. Zároveň je vhodné do adresového prostoru IDATA umístit zásobník STACK. Jak se toto provede, zmíním u konkrétního příkladu, ale IDATA zásobníku se ukládají a vybírají nepřímým způsobem. Ještě je vhodné podotknout, že budeme-li přistupovat do adresového prostoru IDATA od adresy 80H do adresy 0FFH přímým způsobem (a to rozhodne typ použité instrukce pro přístup k datům), nebudeme adresovat buňky paměti mikroprocesoru, ale řídicí registry podpůrných hardwarových periferií. Řídicí registry podpůrných periferií jsou u MCS51 mapovány od adresy 80H do adresy 0FFH, a to pro striktně přímý způsob adresování.

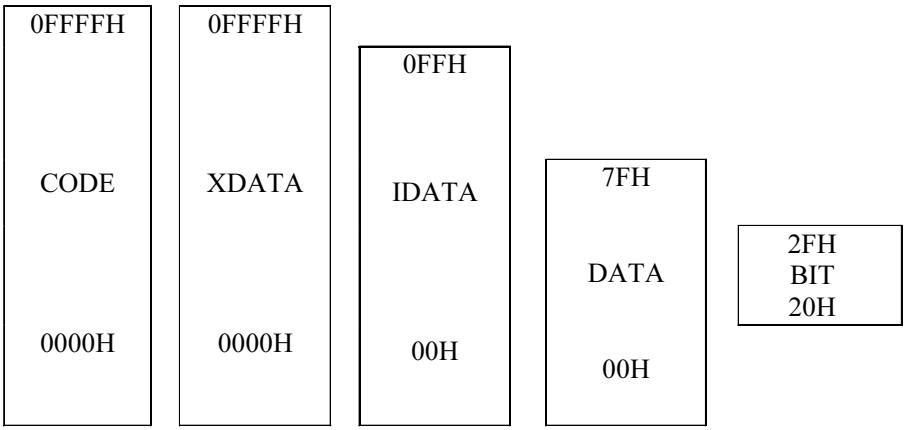
Jako třetí popíšeme adresový prostor **XDATA**. Toto je adresový prostor paměti dat, která je realizována mimo chip mikroprocesoru, tzn. je realizována externě. Tento adresový prostor je definován na adresovém intervalu <0 – 0FFFFH>. Tento interval je největším intervalem adres, které je možné pro mikroprocesory MCS51 použít. Do tohoto adresového prostoru je možné přistupovat pouze nepřímo, a to jen instrukcemi určenými pro přístup k externím datům. Z výše uvedeného vyplývá, že adresový prostor XDATA se s adresovým prostorem DATA a IDATA nepřekrývá. Tento paměťový prostor dat je největší, ale za cenu relativně složitého a tím i zároveň pomalého přístupu k datům.

Čtvrtým adresovým prostorem je adresový prostor **BIT**. Je definován na adresovém intervalu <0 – 7FH>. Tento adresový prostor je specifický tím, že na rozdíl od výše jmenovaných adresových prostorů, které mají standardní šířku slova paměťové buňky, tj. 8 bitů, má šířku slova pouze 1 bit. Adresový prostor BIT se překrývá s adresovým prostorem DATA, a to od adresy 20H až do adresy 2FH. Do tohoto adresového prostoru doporučuji umísťovat proměnné typu boolean, tj. proměnné, které nesou pouze informaci logická **1** – pravda nebo logická **0** – nepravda. Vlastnosti překrytí prostoru DATA a BIT lze též s výhodou využít pro konstrukci celého byte, protože můžeme tento byte sestavit za pomocí instrukcí pro přístup k bitovým adresám a pak celý zkonstruovaný byte přenést bytovou instrukcí jako celek.

Jako poslední zmíním adresový prostor **CODE**. Adresový prostor CODE je prostor, ve kterém je uložen kód programu. Maximální adresový interval adresového prostoru **CODE** je definován na adresovém intervalu <0 – 0FFFFH>. Velikost adreso-

vého prostoru CODE je dána velikostí použité kódové paměti daného typu procesoru, a to jak interní tak i externí.

Tab. 2



Jak již bylo řečeno v adresovém prostoru CODE je uložen vlastní vykonávaný program, z čehož vyplývá, že do tohoto prostoru není možné data zapisovat, ale je možné z něj data pouze číst, a to speciální instrukcí pro přístup k datům kódové paměti. Do tohoto adresového prostoru je možné umístit pouze konstanty. Tyto konstanty musí být definovány speciální direktivou ve zdrojovém kódu programu. Pro větší přehlednost uvedu mapu jednotlivých adresových prostorů v *tab. 2*.

2.3 Symboly instrukcí u MCS51 a registr PSW

Abychom byli plně připraveni k vlastnímu podrobnému popisu instrukčního souboru mikroprocesoru MCS51, je ještě nutné objasnit význam symbolů a zkratek používaných v popisu operandu jednotlivých instrukcí. Jako první uvedu symboly v předchozí kapitole popsané adresové prostory:

- **dadr** Je to 8bitová adresa v paměti RAM na chipu mikroprocesoru a zahrnuje adresový prostor DATA popř. IDATA.
- **badr** Je to adresa bitu v paměti RAM na chipu procesoru zahrnující adresový prostor BIT.

- **cadr** Adresa v adresového prostoru CODE.
- **cpos** Relativní skok v oblasti kódové paměti v rozsahu <–128..127>.

Za druhé uvedu seznam symbolů registrů používaných v podrobném popisu instrukcí:

- **A** Symbol pro registr střadače (accumulator).
- **B** Symbol pomocného registru pro aritmetické operace násobení nebo dělení.
- **AB** Symbol dvojice registrů používaných pro aritmetické operace násobení a dělení.
- **DPTR** Symbol registru ukazatele dat pro nepřímé adresování v externím datovém adresovém prostoru XDATA popř. CODE.
- **PC** Symbol registru programového čítače (čítač instrukcí).
- **SP** Symbol registru ukazatele vrcholu zásobníku (stack pointer).
- **Rr** Symbol registru ze sady registrů registrové banky (bude vysvětleno později), interval parametru
  r = 0–7 při přímém adresování,
  r = 0 nebo 1 při nepřímém adresování.

Jako poslední uvedu obecné symboly instrukcí:

- **offset** Je symbol 8bitového relativního posunutí.
- **data** Symbol přímých dat (přímého operandu).
- **high** Symbol slabiky vyššího řádu (horní osmice bitů).
- **low** Symbol slabiky nižšího řádu (dolní osmice bitů).
- **(X)** Symbol obsahu prvku X.
- **((X))** Symbol obsahu paměťové buňky adresované hodnotou obsahu prvku X (nepřímé adresování).

- ← Symbol přesunu hodnoty zprava doleva.
- # Symbol označení přímých dat v instrukci.
- @ Symbol označení nepřímé adresy v instrukci.

Na závěr této kapitoly si ještě vysvětlíme význam jednotlivých bitů stavového registru ***PSW***. Jak je již z názvu zřejmé, jde o stavový registr, který nám dává informaci o aktuálním stavu procesoru. Některé příznaky tohoto registru jsou svázány se stavem střadače.

Jednotlivé stavové bity tohoto registru jsou rozloženy takto:

C	AC	F0	RS1	RS0	OV	-	P

PSW [adr.0D0H]

Význam bitů stavového registru PSW procesoru je následující:

- **C** Příznak přenosu. Tento příznak je nastaven, dojde-li při některé aritmetické operaci k přenosu do vyššího řádu než je nejvyšší řád střadače (přenos přes hodnotu 255D).
- **AC** Příznak pomocného přenosu, užívá se při operacích s čísly v kódu BCD.
- **F0** Uživatelský příznak, jehož využití je libovolně ponecháno na úvaze programátora.
- **RS1,0** Řídicí bity pro výběr aktuální banky registrů (register bank Select). Nastavují a nulují se programově a určují aktuální banku registrů pro předávání parametrů či lokální proměnné procedur (bude vysvětleno v následujícím textu na konkrétním případě).
- **OV** Příznak přetečení. Nastavuje se tehdy, vznikne-li po aritmetické operaci chybný výsledek.
- **P** Je příznak parity. Tento bit je přímo svázán se střadačem a podává informaci o paritě hodnoty uložené ve střadači. Tento příznak vlastně doplňuje obsah střadače na sudou paritu.

Tímto jsme uzavřeli objasnění všech pojmů nezbytně nutných pro detailní popis jednotlivých instrukcí procesoru MCS51 a nyní již můžeme přistoupit k vlastnímu detailnímu popisu těchto instrukcí.