Mikrořadiče řady 8051.

Řada obvodů 8051 obsahuje typy 8051AH, 8031AH, 8751H, 80C51, 80C31, 8052 a 8032. Jednotlivé obvody se od sebe liší technologií výroby a svojí konstrukcí. Způsob programování je však u všech typů obdobný až na některé drobné výjimky vyplývající z konstrukce mikropočítače. těchto mikropočítačů je vývojovým pokračováním řady 8048. Mikropočítače řady 8051 je však rozšířena a obohacena o nové funkce a představuje kvalitativní skok. Je určena především pro tzv. malé aplikace, kde může plně nahradit 8bitové univerzální mikroprocesory a celou skupinu jejich podpůrných obvodů.

Instrukční soubor monolitockého mikropočítače řady 8051 je nadmnožinou instrukcí mikropočítače 8048, neboť obsahuje takřka všechny jeho instrukce a je rozšířen o další instrukce, typy dat a způsoby adresování. Mikropočítače řady 8051 jsou konstruovány podle tzv. Harwardského principu, který se od klasického Von Neumannovského liší tím, že paměť dat a paměť programu je oddělena. Konstrukční rozdíly mezi jednotlivými obvody řady 8051 ukazuje následující tabulka:

Typ	**ROM/EPROM**	**RAM**
8031AH	nemá	128 B
8051AH	4 KB ROM	128 B
8751H	4 KB EPROM	128 B
80C31	nemá	128 B
80C51	4 KB ROM	128 B
8032AH	nemá	256 B
8052AH	8 KB ROM	256 B

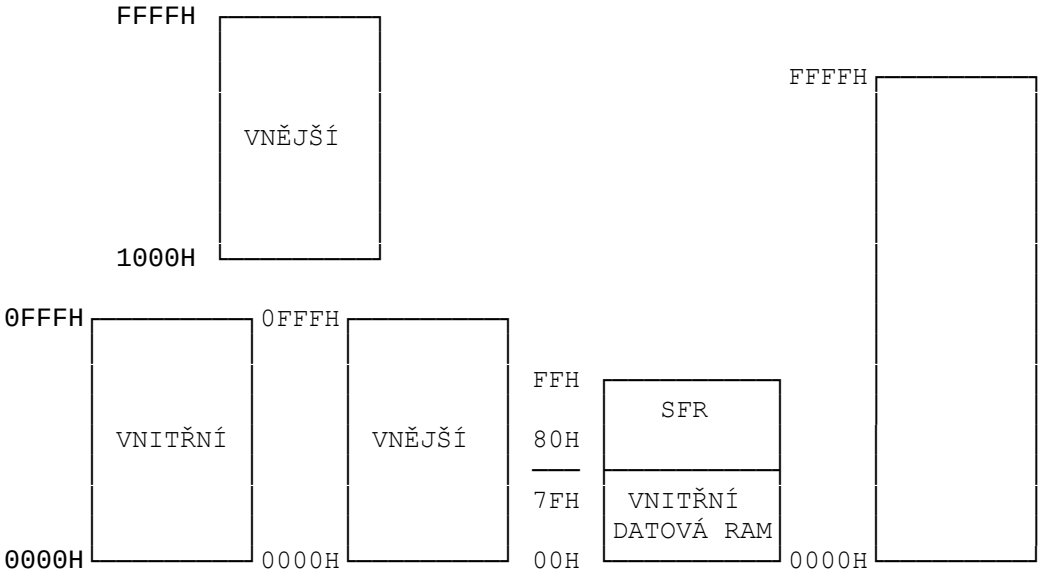
Kromě těchto vnitřních pamětí dat a programu je možno k mikropočítačům připojit vnější paměť dat i vnější paměť programu (obě mohou mít až 64KB).
Obvody řady 8051 jsou mikropočítače, to znamená, že na rozdíl od klasických mikroprocesorů je na čipu obsažena - dalších obvodů, které zjednodušují jejich použití v malých aplikacích, kde není obvod třeba doplňovat dalšími podpůrnými obvody. Hlavní vlastnosti obvodů řady 8051 lze stručně shrnout v následujícím přehledu:

- osmibitová centrální procesorová jednotka (CPU)
- oscilátor a obvody hodin na čipu
- 32 linek V/V (4 osmibitové porty)
- 64 KB adresový prostor pro vnější paměť programu
- 64 KB adresový prostor pro vnější paměť dat
- dva šestnáctibitové časovače/čítače (tři u 8032/8052)
- pět zdrojů přerušení (šest u 8032/8052) se dvěma úrovněmi priorit
- plně duplexní sériový port
- Booleovský procesor

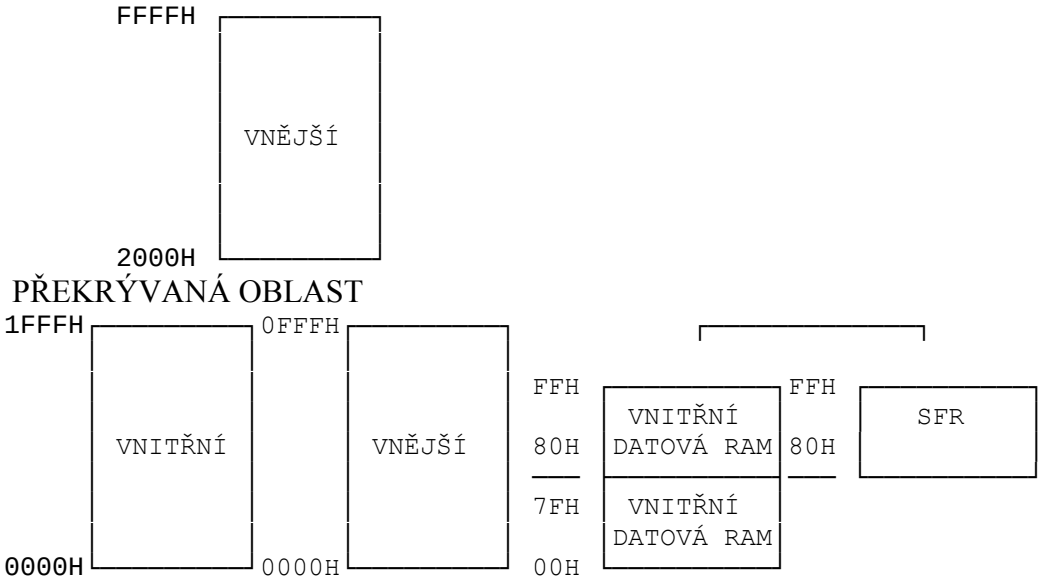
Organizace paměti.

Mikrořadiče 8051 a 8052 mají oddělený prostor pro paměť programu a paměť dat. Paměť programu může mít rozsah až do 64 KB. Nejnižší 4 KB (8 KB pro 8052) mohou být umístěny na čipu. Paměť dat je typu RWM - RAM a její rozsah vně čipu může být až 64 KB. Kromě vnější paměti dat je přímo na čipu 128 B (256 B pro 8052) vnitřní paměti dat a navíc určitý počet registrů speciálních funkcí (SFR - Special Function Registers).

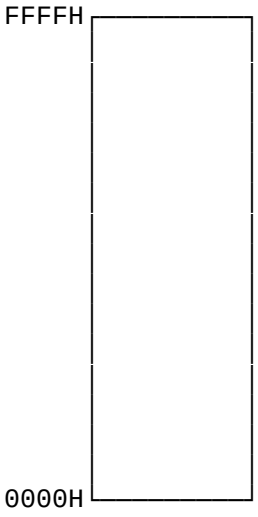
Struktura paměťových prostorů u obvodů 8051.



Struktura paměťových prostorů u obvodů 8052.



PAMĚŤ PROGRAMU VNITŘNÍ PAMĚŤ DAT



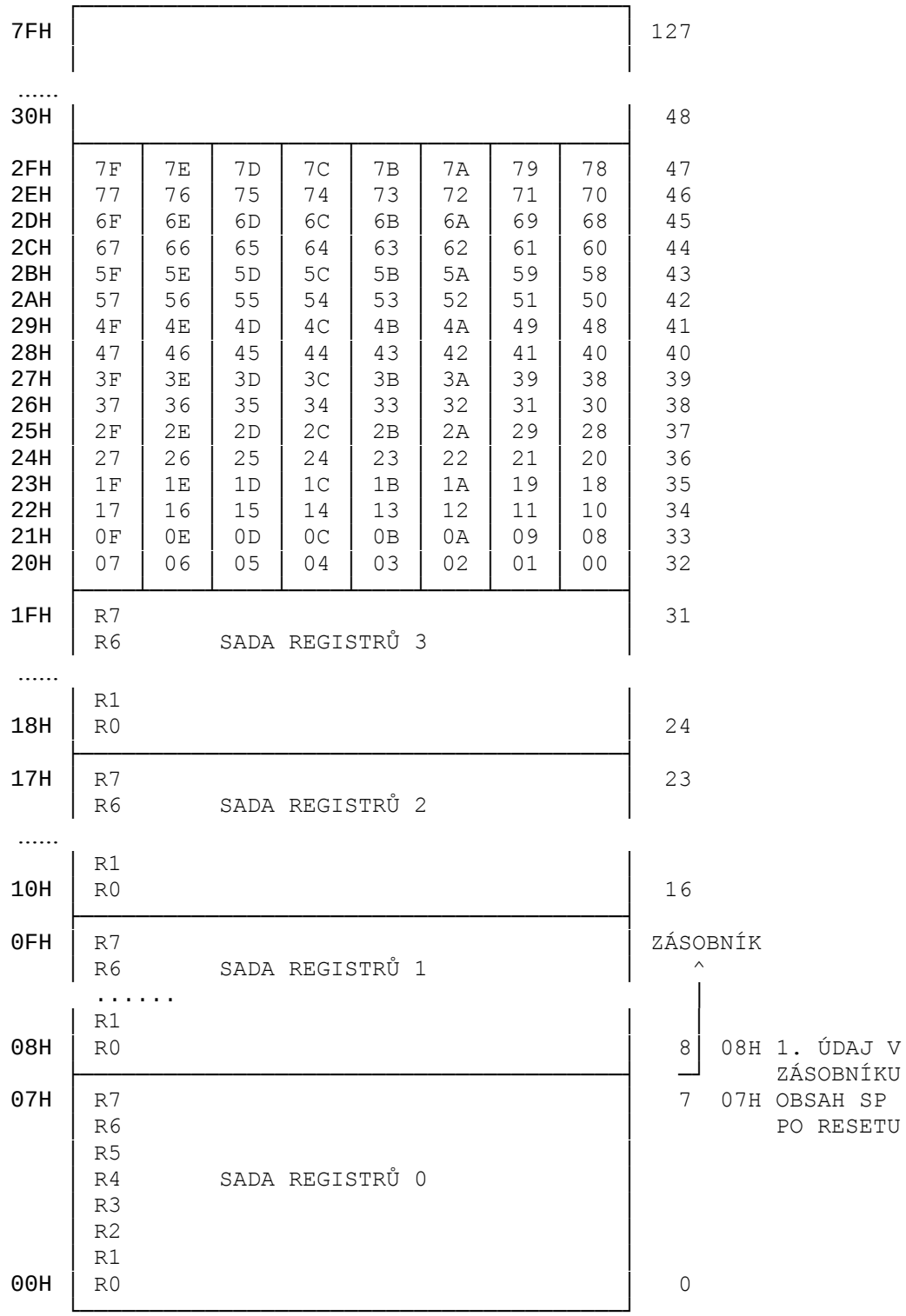
VNĚJŠÍ PAMĚŤ DAT Z uvedených schemat je zřejmé, že obvody 8051 a 8052 mají tři základní paměťové prostory: - 64 KB paměti programu - 64 KB vnější paměti dat - 256 B (384 B pro obvody 8052) vnitřní paměti dat **Adresový prostor paměti programu** Adresový prostor paměti programu má velikost 64 KB a skládá se z vnitřní a vnější části paměti. Odkud se program zavádí určuje signál -EA (negace EA). Je-li na vstup mikroprocesoru -EA přivedena logická hodnota 1, provádí se program z vnitřní paměti programu, pokud adresa nepřekročí hodnotu 0FFFH (1FFFH pro obvody 8052). Obsahy adres 1000H až 0FFFFH (2000H až 0FFFFH pro obvody 8052) se v tomto případě vybírají z vnější paměti programu. V případě, že na vstup -EA je přivedena logická hodnota 0, vybírají se všechny instrukce z vnější paměti programu. Logická hodnota 0 musí být na tento vstup přivedena i u obvodů 8031 a 8032, které vnitřní paměť nemají. V každém případě se pro adresování používá šestnáctibitový programový čítač.
Paměťová místa o adresách 00H až 23H (00H až 2BH pro obvody 8052) v paměti programu jsou využívána pro přechod do obslužných programů přerušení tak, jak ukazuje následující tabulka:

Zdroj přerušení Adresa počátku obsluhy přerušení
Vnější přerušení 0 0003H
Přetečení časovače 0 000BH
Vnější přerušení 1 0013H
Přetečení časovače 1 001BH
Sériový port 0023H
Přetečení časovače 2 002BH (jen u obvodů 8052)

Adresový prostor paměti dat.

Adresový prostor paměti dat se skládá z vnitřního a vnějšího adresového prostoru. Vnější paměť dat se používá při provádění instrukce MOVX.
Vnitřní paměť dat se dělí na tři fyzicky oddělené a vyznačené bloky:
- nižších 128 bytů RAM
- vyšších 128 bytů RAM (dosažitelných pouze u obvodů 8052)
- 128 bytů oblasti registrů speciálních funkcí SFR

I když horní oblast paměti RAM a oblast registrů speciálních funkcí mají stejný adresový prostor, jsou dosažitelné rozdílnými způsoby adresování, které budou probrány později.
Na následujícím obrázku je znázorněna struktura paměťového prostoru nižší oblasti vnitřní paměti dat.



Adresy 00H (0D) až 1FH (31D) Mikropočítač může pracovat celkem se čtyřmi sadami (bankami) registrů pro obecné použití. Každá sada obsahuje osm osmibitových registrů. Tyto sady registrů, z nichž každá obsahuje osm osmibitových registrů, zabírají v nižší oblasti paměti RAM adresy 0 až 31. V každém okamžiku může být aktivní pouze jediná sada registrů (výběr se řídí dvoubitovým polem v PSW). **Adresy 20H (32D) až 2FH (47D)** Dalších šestnáct bytů na adresách 32 až 47 je možno adresovat dvojím způsobem. Buď jako jednotlivé byty, nebo po jednotlivých bitech. Celkem tato oblast tedy představuje 16 x 8 = 128 pozic adresovatelných po jednotlivých bitech.

Bitové adresy lze jako operand instrukce vyjádřit dvojím způsobem:

- Číslo adresovaného bitu připojíme za adresu slabiky. Jako oddělovač slouží tečka (.). Například operand 40.5 označuje bit číslo 5 bytu s dekadickou adresou 40. Pro zadání adresy bytu můžeme použít i hexadecimální adresu - tentýž bit můžeme adresovat jako 28H.5. Takto vyjádřené operandy se nazývají bitové selektory.

- Bitovou adresu lze specifikovat explicitně. Operand v tomto případě vyjadřuje adresu bitu v prostoru bitových adres. Bitové adresy v dekadickém vyjádření 0 - 127, což odpovídá hexadecimálním adresám 00H - 7FH jsou uloženy na adresách 20H až 2FH v paměti RAM a bitové adresy v dekadickém vyjádření 128 až 255 (80H - FFH) se vztahují k prostoru hardwarových registrů (viz následující oddíl).

Adresy 30H (48D) až 7FH (127D)

Zbývající část paměti dat na čipu je adresovatelná po jednotlivých bytech (osmibitových slabikách).

Umístění zásobníku po provedení funkce RESET

Po provedení funkce RESET se nastaví obsah ukazatele zásobníku SP (Stack Pointer) na 07H. Díky mechanismu ukládání pak první údaj vložený do zásobníku se uloží na adresu 08H, další data se pak postupně ukládají na další adresy (přímý zásobník).

Registry speciálních funkcí (hardwarové registry) - přehled.

Označení	**Pozn.**	**Název**	**Adresa**
ACC	*	Střadač (Accumulator)	0E0H
B	*	B registr	0F0H
PSW	*	Stavové slovo programu (Program Status Word)	0D0H
SP		Ukazatel zásobníku (Stack Pointer)	81H
DPTR		Ukazatel dat (Data Pointer) - skládá se z DPH a DPL	83H 82H
P0	*	Port 0	80H
P1	*	Port 1	90H
P2	*	Port 2	0A0H
P3	*	Port 3	0B0H
IP	*	Řízení priority přerušení (Interrupt Priority Control	0B8H
IE	*	Řízení povolení přerušení (Interrupt Enable Control)	0A8H
TMOD		Řízení režimu časovače/čítače (Timer/Counter Mode Control)	89H
TCON	*	Řízení časovače/čítače (Timer/Counter Control)	88H
T2CON	* +	Řízení časovače/čítače 2 (Timer/Counter 2 Control)	0C8H
TH0		Časovač/čítač 0 - vyšší byte (Timer/Counter 0 - high byte)	8CH

Označení	**Pozn.**	**Název**	**Adresa**
TL0		Časovač/čítač 0 - nižší byte (Timer/Counter 0 - low byte)	8AH
TH1		Časovač/čítač 1 - vyšší byte (Timer/Counter 1 - high byte)	8DH
TL1		Časovač/čítač 1 - nižší byte (Timer/Counter 1 - low byte)	8BH
TH2	+	Časovač/čítač 2 - vyšší byte (Timer/Counter 2 - high byte)	0CDH
TL2	+	Časovač/čítač 2 - nižší byte (Timer/Counter 2 - low byte)	0CCH
RCAP2H	+	Časovač/čítač 2 záchytný registr - vyšší byte (Timer/Counter 2 Capture Register - high byte)	0CBH
RCAP2L	+	Časovač/čítač 2 záchytný registr	0CAH

vyšší byte (Timer/Counter 2 Capture
Register - low byte)
SCON * Sériové řízení (Serial Control) 98H
SBUF Vyrovnávací paměť sériových dat 99H
(Serial Data Buffer)
PCON Řízení napájení (Power Control) 87H

Poznámka:
Registry speciálních funkcí označené * jsou adresovatelné jako celý byte nebo po jednotlivých bitech.
Registry speciálních funkcí označené + jsou pouze u obvodů 8052.

Registry speciálních funkcí - použití.
Střadač (ACC)
ACC je registr, který má obvyklé funkce střadače. V operandových částech instrukcí pro práci se střadačem se mnemonicky označuje písmenem A.
B registr
Tento registr se užívá při aritmetických operacích násobení a dělení. Pro ostatní instrukce se může využít jako jiný registr tzv. zápisníkové paměti.
Ukazatel zásobníku (SP)
Registr ukazatele zásobníku je osmibitový. jeho obsah se inkrementuje dříve než se uloží data instrukcemi PUSH a CALL. Zásobník může být umístěn kdekoliv ve vnitřní paměti RWM - RAM na čipu. Provedením funkce RESET se nastaví obsah SP na hodnotu 07H, takže data budou do zásobníku ukládána od adresy 08H.
Ukazatel dat (DPTR)
Ukazatel dat tvoří vyšší byte (DPH) a nižší byte (DPL). Registr ukazatele uchovává šestnáctibitovou adresu. Může být využíván jako šestnáctibitový registr nebo jako dva nezávislé osmibitové registry.
Porty 0 až 3
P0, P1, P2 a P3 jsou registry speciálních funkcí obsahující záchytné klopné obvody (latche) portů 0, 1, 2 a 3. Při použití mikropočítače s vnější pamětí jsou porty 0 a 2 využity pro styk s vnější pamětí (port 2 pro vysílání vyššího bytu šestnáctibitové adresy, port 0 je multiplexován pro adresovou sběrnici (nižší byte) nebo datovou sběrnici). Porty 1 a 3 je možno využít jako klasické obousměrné V/V porty.
Stavové slovo programu (PSW)
Registr stavového slova programu má osm bitů a jeho struktura je podrobně uvedena na následujícím obrázku. Významy jednotlivých bitů stavového slova programu jsou v dále uvedené tabulce.

P S W

CY	AC	F0	RS1	RS0	OV	-	P

7 6 5 4 3 2 1 0 číslo bitu
Význam bitů stavového slova programu
Bit
Symbol v PSW Název a význam

CY 7 Příznak přenosu (Carry Flag)
AC 6 Příznak pomocného přenosy (Auxiliary Carry
flag). Užívá se při operacích s čísly v kódu
BCD.
F0 5 Uživatelský příznak (Flag 0). Jeho využití je ponecháno na úvaze uživatele.
RS1 4 Řídicí bity 1 a 0 pro výběr sady registrů
RS0 3 (Register bank Select). Nastavují a nulijí

se softwarově a určují sadu registrů takto:
RS1 RS0 Sada Adresy

0 0 0 00H - 07H
0 1 1 08H - 0FH
1 0 2 10H - 17H
1 1 3 18H - 1FH
OV 2 Příznak přetečení (Overflow flag)
- Rezerva
P 0 Příznak parity (Parity flag). Nastavuje/nulu§
je se hardwarově při každém instrukčním cyklu
a indikuje lichý/sudý počet bitů ve střada§
dači, které obsahují jedničky. Jinak řečeno -
příznak parity doplňuje obsah střadače sudou
paritou.

Vyrovnávací paměť sériových dat (SBUF)
Vyrovnávací paměť sériových dat tvoří ve skutečnosti dva oddělené registry a to vyrovnávací registr pro vysílání a vyrovnávací registr pro příjem. Data přesouvaná do SBUF se ukládají do vysílacího vyrovnávacího registru (přesun bytu do SBUF zahajuje sériový přenos dat). obdobně data čtená s SBUF jsou odebírána z přijímacího vyrovnávacího registru.
Časovače
Registrové páry (TH0, TL0), (TH1, TL1) a (TH2, TL2) jsou šestnácti bitové čítací registry pro časovače/čítače 0, 1, 2.
Záchytné registry
Registrový pár (RCAP2H, RCAP2L) představuje záchytné registry (capture registers) pro časovač 2 pracující v tzv. "záchytném režimu". V tomto režimu se v závislosti na změně na vstupu s označením T2EX obvodů 8052 zkopírují obsahy TH2 a TL2 do RCAP2H a RCAP2L.
Časovač 2 může také pracovat v šestnáctibitovém samoplnícím režimu, kdy RCAP2H a RCAP2L obsahují hodnotu pro nové naplnění. Podrobněji budou vlastnosti časovačů popsány později.
Řídicí registry
Registry speciálních funkcí IP, IE, TMOD, TCON, T2CON, SCON a PCON obsahují řídicí a stavové bity pro přerušovací systém, časovače/čítače a sériový port. Budou popsány v dalším textu.
Adresy bytů v registrech speciálních funkcí ve vnitřní paměti dat

F	B														
E	ACC														
D	PSW														
C								T2CON		RCAP2L	RCAP2H	TL2	TH2		
B	P3							IP							
A	P2							IE							
9	P1							SCON	SBUF						
8	P0	SP	DPL	DPH			PCON	TCON	TMOD	TL0	TL1	TH0	TH1		

0 1 2 3 4 5 6 7 8 9 A B C D E F Byty, u kterých není vyplněn název registru speciálních funkcí představují tzv. rezervované adresy. Při pokusu o čtení z rezervovaných adres budou načtena nedefinovaná data. Při pokusu o zápis na rezervované adresy dojde ke ztrátě datové slabiky. Vzhledem k tomu, že oblast rezervovaných adres v rámci hardwarových registrů je pro různé typy čipů různá, může dojít při přenášení programů k nekompatibilitě. Je třeba na to dát pozor při používání nepřímého adresování u adres vyšších než 127.

Bitově adresovatelné slabiky v oblasti hardwarových registrů

	F8H	FF	FE	FD	FC	FB	FA	F9	F8
B	F0H	F7	F6	F5	F4	F3	F2	F1	F0
	E8H	EF	EE	ED	EC	EB	EA	E9	E8
ACC	E0H	E7	E6	E5	E4	E3	E2	E1	E0
	D8H	DF	DE	DD	DC	DB	DA	D9	D8
PSW	D0H	D7	D6	D5	D4	D3	D2	D1	90
T2CON	C8H	CF	CE	CD	CC	CB	CA	C9	98
	C0H	C7	C6	C5	C4	C3	C2	C1	C0
IP	B8H	BF	BE	BD	BC	BB	BA	B9	B8
P3	B0H	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	AF	AE	AD	AC	AB	AA	A9	A8
P2	A0H	A7	A6	A5	A4	A3	A2	A1	A0
SCON	98H	9F	9E	9D	9C	9B	9A	99	98
P1	90H	97	96	95	94	93	92	91	90
TCON	88H	8F	8E	8D	8C	8B	8A	89	88
P0	80H	87	86	85	84	83	82	81	80

Znovu připomínáme, že v oblasti hardwarových registrů nejsou definovány všechny adresy (viz byte s adresou F8H, E8H, D8H a C0H). I v tomto případě při zápisu bitu na takovou adresu dojde k jeho ztrátě, při čtení bitu je vrácena nedefinovaná hodnota.

Programování mikropočítačů 8051.

Pro programování jednočipových mikropočítačů řady 8051 budeme používat jazyk symbolických instrukcí - asembler, strojově orientovaný jazyk, který umožňuje pružně a beze zbytku využívat technické prostředky mikrořadiče. Zdrojový text programu, zapsaný v tomto jazyce je nutné zpracovat překladačem (kompilátorem) CASS51, který generuje strojový kód mikroprocesoru, který je uložen ve zvláštním souboru s příponou .HEX ve formátu INTEL-HEX. Tento formát je vhodný pro zavedení buď do paměti ROM prostřednictvím programátoru těchto pamětí, nebo je využíván programy vývojového prostředí k zavedení programu do simulátoru paměti EPROM.

V průběhu překladu je možno generovat i protokol o překladu programu. Tento protokol je uložen ve zvláštním souboru s příponou .PRN. Protokol o překladu programu může obsahovat tyto části:

- opis zdrojového textu symbolického programu tak, jak byl zadán k překladu,
- hlášení o chybách,
- tabulku symbolů a křížových odkazů.

Překladač CASS51 je možno spouštět dvěma způsoby - jednak přímo z integrovaného vývojového prostředí volbou Compile nebo z DOSu příkazem:

CASS51 <jméno>[.ASM] [/c][/s][/n]

jméno : jméno souboru se zdrojovým textem programu
/c : parametr pro generování tabulky křížových odkazů
/s : parametr pro generování tabulky symbolů
/n : parametr pro potlačení generování protokolu o překladu

Popis formátu instrukce

Každý řádek programu musí splňovat následující formát:

[návěští:] <pseudo>instrukce [operand<y>] [:komentář]

Přesná délka jednotlivých polí není určena. Pro větší přehlednost a pohodlnější práci se však doporučuje vyhradit pro každé pole 8 znaků (většina editorů včetně SIMED51 má tabelátory nastaveny právě na tuto velikost).

Návěští

Návěští je symbolické jméno, které je ukončeno dvojtečkou. Jménu je při překladu přiřazena konkrétní adresa (symbolická adresa je nahrazena adresou skutečnou). Dané návěští smí být deklarováno pouze jednou. Při vícenásobném výskytu téhož jména v poli návěští hlásí překladač chybu.

Pro vytváření symbolických jmen je možno používat písmena abecedy, číslice 0 až 9 a znaky _(podtrhovátko), ?, $, @. Prvním znakem však musí být vždy písmeno. Překladač rozlišuje velká a malá písmena. Jméno může být dlouhé 8 znaků.

Při vytváření symbolických jmen nelze použít vyhrazené symboly překladače:

- názvy instrukcí a pseudoinstrukcí
- názvy registrů
- názvy bitů SFR

Instrukce (pseudoinstrukce)

V tomto poli je uveden symbolický název instrukce (pseudoinstrukce). Překladem instrukce se vytváří cílový kód (strojová instrukce mikroprocesoru. Překladem pseudoinstrukce cílový kód nevzniká, slouží jako pokyn pro překladač.

Operand

Pokud instrukce vyžaduje jeden nebo dva operandy, uvádí se za symbolický název instrukce a je od ní oddělen nejméně jednou mezerou. Operandy se vzájmně oddělují čárkou.

Komentář

Na konci každého řádku (ať už je tvořen instrukcí s operandem nebo bez) je možno uvést poznámku, která musí být uvozena středníkem. Poznámka se může rovněž nacházet na samostatném řádku.

Příklad:

```
START: MOV A,B
 MOV DPTR,#124 ;toto je poznámka
 SETB C ;toto je rovněž poznámka
;poznámka na samostatném řádku
```

Způsob zápisu přímých operandů

Většina instrukcí umožňuje použít jako operandy i tzv. přímé hodnoty, tj. čísla v rozsahu 0-255 nebo 0-65535 (podle typu instrukce). Kompilátor CASS51 umožňuje zapsat přímé operandy v zásadě čtyřmi způsoby:

1. Dekadický zápis

číslo obsahuje znaky 0-9, může být ukončeno znakem D

```
Příklad: MOV A,#123
 ANL A,#34D
```

2. Binární zápis

číslo obsahuje znaky 0 a 1, musí končit znakem B

```
Příklad: MOV A,#11110110B
 ANL A,#101B
```

3. Oktalový zápis

číslo obsahuje znaky 0-7, musí končit znakem O nebo Q

```
Příklad: MOV A,#173O
 ANL A,#42Q
```

4. Hexadecimální zápis

číslo obsahuje znaky 0-9 a A-F, musí končit znakem H

```
Příklad: MOV A,#7BH
 ANL A,#0AAH
```

V případě, že hexadecimální operand začíná písmenem (A-F), musí být první znak číslice 0.

Výrazy

Přímé operandy, popisované v předcházejícím textu, je možno spojovat pomocí operátorů do výrazů. Tyto výrazy mohou být použity v instrukcích na místě přímých operandů. Překladač nejprve vyhodnotí (vypočte) uvedený výraz a tuto vypočtenou hodnotu použije jako přímý operand.

Kompilátor CASS51 umožňuje používání čtyř skupin operandů:

- aritmetické operátory
- operátory posunu
- logické operátory
- operátory separace částí slova

Následuje souhrnný přehled operátorů v jednotlivých skupinách:

Atitmetické operátory

+ unární + nebo binární sčítání
- unární - nebo binární odečítání
***** násobení
/ dělení (celočíselné)

Operátory posunů

y **SHR** x posun operandu y doprava o x bitových pozic
y **SHL** x posun operandu y doleva o x bitových pozic

Logické operátory

NOT logická negace
AND logický součin
OR logický součet

Operátory separace částí slova

HIGH vyšší bajt 16-bitového slova
LOW nižší bajt 16-bitového slova

Výrazy jsou vyhodnocovány zleva doprava. Operace s operátory s vyšší prioritou jsou vyhodnoceny dříve než u operátorů s nižší prioritou. Pořadí vyhodnocení je samozřejmě možno změnit pomocí závorek. Pořadí priority operátorů je uvedeno v následující tabulce:

HIGH, LOW nejvyšší priorita
NOT
SHR, SHL
AND
OR
*, /
+, - nejnižší priorita

Pseudoinstrukce překladače

Kromě řádků s instrukcemi programu může překládaný zdrojový text obsahovat příkazy pro řízení práce překladače - pseudoinstrukce (někdy se jim též říká direktivy). Pseudoinstrukce můžeme rozdělit do několika základních skupin, jejich přehled ukazují následující odstavce:

Definice symbolických jmen

Tato skupina pseudoinstrukcí dovoluje definování uživatelských jmen. Obsahuje pseudoinstrukce EQU, SET, DATA. Jejich syntaxe a stručný popis následuje.

EQU

Syntaxe: <jméno> EQU <výraz>
<jméno> EQU <rezervovaný symbol Rx>
Symbolické jméno je přiřazena hodnotě, získané vyčíslením výrazu nebo některému z registrů R0 až R7.

Příklad: POCET EQU 10 ;symbol POCET je ekvivalentní hodnotě 10
ZDE EQU $;symbolu ZDE bude přiřazena současná
;hodnota počítadla adres
BASE EQU R0 ;symbol BASE je přiřazen registru R0

SET

Syntaxe: <jméno> SET <výraz>
Účinek pseudoinstrukce SET je stejný jako u EQU. Rozdíl spočívá v tom, že symbolické jméno definované pseudoinstrukcí SET lze pomocí další pseudoinstrukce SET předefinovat. Pomocí SET tedy může být jednomu jménu přiřazena vícekrát různá hodnota (jedná se tedy o přiřazení hodnoty), kdežto EQU odpovídá definici a jméno už nesmí být na jiném místě definováno. SET nelze použít pro přiřazení symbolu registrům.

Příklad: POCET SET 20 ;symbol POCET je ekvivalentní hodnotě 20

...
...
POCET SET 25 ;symbol POCET je předefinován na 25

DATA

Syntaxe: <jméno> DATA <výraz>

Pseudoinstrukce DATA je určena k definici symbolických jmen, které se odkazují na adresy vnitřní paměti dat umístěné na čipu (direct address).

Příklad: TABPOC DATA 70H ;definice začátku tabulky TABPOC
;na adrese 70H
TABKON DATA 7FH ;definice konce tabulky TABKON
;na adrese 7FH

Definice uživatelských označení bitů

BIT

Syntaxe: <jméno> BIT <hodnota>

Pseudoinstrukce BIT umožňuje přiřadit přímo adresovatelným bitům ve vnitřní paměti dat symbolická jména. Jako hodnota může být použita konstanta z intervalu 0-FF nebo symbolické označení bitu dle mnemoniky výrobce.

Příklad: PRVNI BIT 34H
DRUHY BIT TxD
TRETI BIT ACC.3

Výběr použité banky registrů

USING

Syntaxe: [návěští:] USING <výraz>

Pseudoinstrukce definuje použití příslušné banky registrů od místa svého výskytu. Výraz po vyhodnocení musí nabývat hodnot z intervalu 0 až 3. Při zpracování instrukcí, které pracují s registry R0 až R7, je při překladu zohledněna zvolená banka registrů tehdy, je-li na místě registrů v instrukci uveden operand ARx (Absolute Register), kde x je číslo registru 0 až 7. V takovém případě přeloží překladač instrukci jakoby byla použita instrukce s přímou adresou. Implicitně je zvolena banka 0.

Příklad: START: mov AR0,#1 ;vloží hodnotu 1 do R0
;implicitně nastavené banky 0
using 1 ;definuje použití banky 1
;pro instrukce ARx
mov AR0,#2 ;vloží hodnotu 2 do R0 před§
;cházejícím příkazem nastavené
;banky 1
mov R1,#9 ;vloží hodnotu 9 do R1 banky 0

Podmíněný překlad

Pseudoinstrukce IF, ELSE, ENDIF umožňují zavést podmíněný překlad.

Syntaxe: IF <výraz>
<seznam příkazů 1>
ELSE
<seznam příkazů 2>
ENDIF

Překladač vyhodnotí výraz a pokud je výsledek různý od nuly, přeloží následující sekvenci instrukcí <seznam příkazů 1> až po výskyt pseudoinstrukce ELSE nebo ENDIF (chybí-li ELSE a <seznam příkazů 2>. Při výskytu pseudoinstrukce ENDIF je podmíněný překlad ukončen. Je-li ELSE přítomno, <seznam příkazů 2> se nepřekládá. Je-li výraz roven nule, <seznam příkazů 1> se nepřekládá. Vyskytuje-li se pseudoinstrukce ELSE, provede se překlad sekvence instrukcí <seznam příkazů 2>.

Vnoření pseudoinstrukcí IF, ELSE, ENDIF není povoleno.

Definice dat

DB
Syntaxe: [návěští:] DB <výraz>[,<výraz>]
Pseudoinstrukce postupně ukládá jednobajtové hodnoty, získané vyhodnocením jednotlivých výrazů, na aktuální adresy, počínaje aktuální adresou programového čítače.
Výraz musí nabývat hodnot v rozsahu 0 - 0FFH. Povoleny jsou také řetězce znaků uzavřené do apostrofů, např. posloupnost znaků '123' se uloží jako 31H,32H,33H. Počet výrazů za jedním příkazem DB je omezen pouze délkou řádku. Návěští je nepovinné.
DW
Syntaxe: [návěští:] DW <výraz>
Pseudoinstrukce ukládá dvoubajtovou hodnotu, získanou vyhodnocením výrazu, na aktuální adresu, počínaje aktuální adresou programového čítače.
Výraz musí nabývat hodnot v rozsahu 0 - 0FFFFH a ukládá se do paměti tak, že nejprve je uložen vyšší bajt (na nižší adresu) a potom nižší bajt (na vyšší adresu). Návěští je nepovinné.
Příklad: PRICHOD: DW 710 ;uložení číselného údaje ;do paměti

Rezervování paměti
DS
Syntaxe: [návěští:] DS <výraz>
Při překladu je v paměti rezervován počet bajtů daný výrazem. Návěští je nepovinné.
Hodnoty případných proměnných, použitých ve výrazu, musí být všechny definovány před vyčíslením výrazu.
Příklad: DIRA: DS 10 ;rezervuje 10 bajtů

Vkládání souborů
INCLUDE
Syntaxe: INCLUDE(soubor)
Pseudoinstrukce INCLUDE umožňuje zahrnout do programu, ve kterém je uvedena, zdrojový text programu uložený v jiném souboru. V cílovém textu dojde k nahrazení pseudoinstrukce INCLUDE obsahem souboru, který je uveden jako parametr. Soubor musí existovat, jinak je hlášena chyba. Vícenásobné vnoření není dovoleno, avšak pseudoinstrukce INCLUDE může být použita na více místech v jednom programu.

Ukončení zdrojového textu
END
Syntaxe: [návěští:] END
Pseudoinstrukce oznamuje překladači konec překládané části zdrojového textu. Jakýkoliv další text je ignorován. Návěští je nepovinné. Chybí-li tato pseudoinstrukce na konci zdrojového textu, je v protokolu o překladu uvedena varovná zpráva:
Warning *** END expected

Instrukční soubor 8051

Instrukční soubor mikropočítačů 8051 má celkem 111 instrukcí, z nichž 49 je jednoslabikových, 45 dvouslabikových a 17 třískabikových. Každá instrukce má pole operačního kódu (zkratka anglického vyjádření činnosti instrukce), za kterým následuje pole operandů, zpravidla ve tvaru "cílový operand, zdrojový operand"). Pole operandů určuje použitý typ dat a způsob (nebo způsoby) adresování.

Soubor instrukcí obvodů 8051 se rozděluje do čtyř funkčních skupin:

- přesuny dat,
- aritmetické operace,
- logické operace,
- operace pro předání řízení (skoky).

V následujícím textu se postupně seznámíme se soubory instrukcí v těchto skupinách i s elementárními programátorskými technikami a postupy, které je nutno zvládnout pro programování konkrétních aplikací.

Přesuny dat a zápis konstant do paměti

Přesuny dat v paměti procesoru provádí instrukce přesunu (anglicky move):

MOV <oper1>,<oper2>

Začněme od nejjednoduššího příkladu, a to je zápis konstanty do registru A (střadače).

```
MOV A,#12
```

Po provedení této instrukce se naplní obsah registru A číslem 12. Zde je nutno upozornit na znak # (dvojitý křížek), který je uveden před číslem 12. Tento znak je v assembleru 8051 velmi důležitý, protože odlišuje zápis konstanty (přímých dat) od zápisu hodnoty na přímé adrese (adresa v oblasti vnitřní paměti RAM nebo oblasti registrů speciálních funkcí SFR). Špatné použití znaku # (ať už jeho vynechání či nadbytečnost) má za následek vznik těžko zjistitelných chyb a způsobuje "záhadné" chování programu.

Příklad:

```
MOV R0,#10 ;zápis čísla 10 do registru R0
MOV A,#0 ;zápis 0 do A
MOV A,0 ;zápis čísla 10 do A !!!
```

V posledním řádku programu nedojde k vynulování registru A, ale k přesunu hodnoty uložené na přímé adrese 0 do registru A. Protože na adrese 0 ve vnitřní paměti RAM leží registr R0, dojde k přepsání hodnoty, která je v něm uložena (10 - viz první instrukce) do registru A.

Instrukce MOV má široké spektrum parametrů a je možno ji použít ve všech adresovacích modech obvodu 8051.

Příklad: Následující program umožňuje 8x sejmout hodnotu portu P1 a sejmutá data postupně uložit do **vnitřní** paměti RAM na adresy 20H až 27H.

```
MOV R0,#20 ;počáteční adresa ukládání
MOV B,#8 ;počet průchodů cyklem
CYKL: MOV @R0,P1 ;načtení portu P0 a uložení na adresu
;obsaženou v R0 (nepřímá adresa)
INC R0 ;zvyš ukazatel
DJNZ B,CYKL ;proveď celkem 8x
```

Uvedený příklad ilustruje použití instrukce MOV při nepřímém adresování - obsah P1 se neukládá do R0, ale na adresu, která je zapsána v R0 - což je vyjádřeno uvedením znaku @ před označením registru.

Příklad: Tento příklad je modifikací předchzího příkladu pro ukládání načtených údajů do **vnější** paměti dat.

```
MOV R0,#20 ;počáteční adresa ukládání
MOV B,#8 ;počet průchodů cyklem
CYKL: MOVX @R0,P1 ;načtení portu P0 a uložení na adresu
;obsaženou v R0 (nepřímá adresa)
;do vnější paměti dat
INC R0 ;zvyš ukazatel
```

```
DJNZ B,CYKL ;proveď celkem 8x
```

Pro adresování vnější paměti dat je nutno použít instrukci MOVX (Move External).

Uvedené příklady zatím prováděly přesuny dat v **paměti údajů** (ať už vnitřní, či vnější). Pokud budeme chtít načíst údaje z **paměti programu** (např. programové konstamnty), musíme užít instrukci MOVC (Move Code) - viz následující příklad.

Příklad: Program ukazuje, jak je možno přemístit konstanty (čísla 1 až 10), uložené v tabulce umístěné v paměti programu, do paměti dat od adresy 20H.

```
ORG 0
MOV DPTR,#TAB ;zápis adresy tabulky do DPTR
MOV R7,#TAB_LEN ;zápis délky tabulky do R7
MOV R0,#20H ;tabulka se bude ukádat
;od adresy 20H ve vnitřní RAM
CYKL: MOV A,#0 ;nulování A
MOVC A,@A+DPTR ;přesun jednoho prvku tabulky do A
MOV @R0,A ;přesun z A do RAM
INC R0 ;zvyš ukazatel do RAM
INC DPTR ;zvyš ukazatel do ROM
DJNZ R7,CYKL ;opakuj přes celou délku
ORG 300H
TAB: DB 1,2,3,4,5 ;jednotlivé prvky tabulky
DB 6,7,8,9,10
TAB_LEN EQU $-TAB ;výpočet délky tabulky provede
;překladač
```

V programu je použita instrukce MOV DPTR,#TAB, která do registru DPTR načte adresu tabulky. Instrukce s těmito parametry pracuje jako jediná se šestnácti bity, všechny ostatní parametry instrukce MOV pracují pouze s osmibitovými údaji.

Poslední řádek programu ukazuje jeden ze způsobů použití pseudoinstrukce EQU pro výpočet délky tabulky (symbol $ představuje aktuální stav čítače adres). Kdybychom v posledním řádku uvedli:

```
TAB_LEN EQU 10
```

program by pracoval stejně, ale při jakékoliv změně délky tabulky bychom tento údaj museli neustále počítat a měnit (což může zvláště u delších tabulek vést k chybám). Postupem uvedeným v předcházejícím příkladu přenecháme starost o výpočet délky tabulky překladači.

Pro přesun údajů je také možno použít instrukce, provádějící výměnu dat operandů, jejichž obecný tvar je

```
XCH A,<parametr>
```

které vymění obsah registru A s druhým parametrem. Sekvenci instrukcí

```
;výměna obsahu A, R0
MOV B,A
MOV A,R0
MOV R0,B
```

je možno nahradit jedinou instrukcí:

```
XCH A,R0
```

Z předcházejících příkladů je zřejmé, že instrukce přesunů mohou využívat různé metody adresování. Dříve, než bude uveden podrobný popis jednotlivých instrukcí MOV, uvedeme stručný přehled adresovacích metod využívaných obecně instrukcemi obvodů 8051. Mikropočítač 8051 užívá pět způsobů adresování:

- adresování s registrem
- adresování přímé
- adresování nepřímé s registrem
- adresování s přímým operandem
- adresování nepřímé s bázovým registrem a s indexovým registrem

Na základě výše uvedeného rozdělení metod adresování nyní uvedeme stručný přehled paměťových oblastí, využívaných jednotlivými adresovacími metodami.

Adresování s registrem
- registry R0 až R7
- A, B, CY (bit),AB (dva bajty), DPRT (dvojitý bajt)

Adresování přímé
- nižších 128 bajtů ve vnitřní paměti dat
- registry speciálních funkcí
- 128 bitů ve vnitřní paměti dat
- bity v bitově adresovatelných registrech SFR

Adresování nepřímé s registrem
- vnitřní paměť dat (@R0, @R1, @SP - jen PUSH a POP)
- nižší čtveřice bitů ve vnitřní paměti dat (@R0, @R1)
- vnější paměť dat (@R0, @R1, @DPTR)

Adresování s přímým operandem
- paměť programu (přímý operand - přímá data - jsou součástí instrukce, jsou obsažena ve strojovém kódu instrukce)

Adresování nepřímé s bázovým a indexovým registrem
- paměť programu (@A+DPTR, @A+PC)

Podrobný popis instrukcí MOV pro přesun dat 8 bitů

MOV A,#data Move Immediate Data to Accumulator
Přesun přímých dat do střadače
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |01110100| data |
7 0 7 0
Činnost: A <- data
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce přesune hodnotu osmibitových přímých dat do střadače

---

MOV A,Rr Move register to Accumulator
Přesun obsahu registru do střadače
Operandy: A Střadač
Rr Registr 0 <= r <= 7
Strojový kód: |11101rrr|
7 0
Činnost: A <- (Rr)
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce přesouvá obsah registru R
do střadače

MOV A,@Rr Move Indirect Address to Accumulator
Přesun obsahu nepřímé adresy do střadače
Operandy: A Střadač
Rr Registr 0 <= r <= 1
Strojový kód: |1110011r|
7 0
Činnost: A <- ((Rr))
Slabik: 1

Cyklů: 1
Příznaky: P
Popis: Instrukce přesune obsah paměťové buňky vnitřní paměti dat adresované registrem r do střadače

---

MOV A,data adr Move Memory to Accumulator
Přesun obsahu datové adresy do střadače
Operandy: A Střadač
data adr 0 <= data adr <= 255
Strojový kód: |11100101|data adr|
7 0 7 0
Činnost: A <- (data adr)
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce přesune obsah určené datové adresy do střadače

---

MOV Rr,A Move Accumulator to Register
Přesun obsahu střadače do registru
Operandy: Rr Registr 0 <= r <= 7
A Střadač
Strojový kód: |11111rrr|
7 0
Činnost: Rr <- A
Slabik: 1
Cyklů: 1
Příznaky: -
Popis: Instrukce přesune obsah střadače do registru r

MOV @Rr,A Move Accumulator to Indirect Address
Přesun obsahu střadače na nepřímou adresu
Operandy: Rr Registr 0 <= r <= 1
A Střadač
Strojový kód: |0111011r|
7 0
Činnost: (Rr) <- (A)
Slabik: 1
Cyklů: 1
Příznaky: -
Popis: Instrukce přesouvá obsah střadače do paměťové buňky ve vnitřní paměti dat adresované registrem r

---

MOV data adr,A Move Accumulator to Memory
Přesun obsahu střadače na datovou adresu
Operandy: data adr 0 <= data adr <= 255
A Střadač
Strojový kód: |11110101|data adr|
7 0 7 0
Činnost: data adr <- (A)
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce přesune obsah střadače na

určenou datovou adresu

MOV Rr,#data Move Immediate Data to Register
Přesun přímých dat do registru
Operandy: Rr Registr 0 <= r <= 7
data -256 <= data = +255
Strojový kód: |01111rrr| data |
7 0 7 0
Činnost: Rr <- data
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce přesune hodnotu osmibitových
přímých dat do registru r

MOV Rr,data adr Move Memory to Register
Přesun obsahu datové adresy do registru
Operandy: Rr Registr 0 <= r <= 7
data adr 0 <= data adr <= +255
Strojový kód: |10101rrr|data adr|
7 0 7 0
Činnost: Rr <- (data adr)
Slabik: 2
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune obsah určené datové slabiky do registru r

---

MOV @Rr,#data Move Immediate Data to Indirect Address
Přesun přímých dat na nepřímou adresu
Operandy: Rr Registr 0 <= r <= 1
data -256 <= data <= +255
Strojový kód: |0111011r| data |
7 0 7 0
Činnost: (Rr) <- data
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce přesouvá hodnotu osmibitových přímých dat do paměťové buňky ve vnitřní paměti dat adresované obsahem registru r

---

MOV @Rr,data adr Move Memory to Indirect Address
Přesun obsahu datové adresy na nepřímou adresu
Operandy: Rr Registr 0 <= r <= 1
data adr 0 <= data adr <= 255
Strojový kód: |1010011r|data adr|
7 0 7 0
Činnost: (Rr) <- (data adr)
Slabik: 2
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune obsah určené datové
adresy do paměťové buňky vnitřní paměti
adresované obsahem registru r.

MOV data adr,#data Move Immediate Data to Memory
Přesun přímých dat na datovou adresu

Operandy: data adr 0 <= data adr = 255
data -256 <= dat <= +255
Strojový kód: |01110101|data adr| data |
7 0 7 0 7 0
Činnost: data adr <- data
Slabik: 3
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune hodnotu smibitových přímých dat na určenou datovou adresu

---

MOV data adr,@Rr Move Indirect Address to Memory
Přesun obsahu nepřímé adresy na datovou adresu
Operandy: data adr 0 <= data adr <= 255
Rr Registr 0 <= r <= 1
Strojový kód: |1000011r|data adr|
7 0 7 0
Činnost: data adr <- ((Rr))
Slabik: 2
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune obsah paměťové buňky ve vnitřní paměti dat adresované registrem r na určenou datovou adresu

---

MOV data adr,Rr Move Register to Memory
Přesun obsahu registru na datovou adresu
Operandy: data adr 0 <= data adr <= 255
Rr Registr 0 <= r <= 7
Strojový kód: |10001rrr|data adr|
7 0 7 0
Činnost: data adr <- (Rr)
Slabik: 2
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune obsah registru r
na určenou datovou adresu

MOV data adr 1,data adr 2
Move Memory to Memory
Přesun obsahu datové adresy na datovou adresu
Operandy: data adr 1 0 <= data adr 1 <= 255
data adr 2 0 <= data adr 2 <= 255
Strojový kód: |10000101|data adr 2|data adr 1|
7 0 7 0 7 0
Činnost: data adr 1 <- (data adr 2)
Slabik: 3
Cyklů: 2
Příznaky: -
Popis: Instrukce přesune obsah zdrojové datové
adresy (data adr 2) na cílovou adresu (data adr 1)

MOVX @DPTR,A Move Accumulator to External Memory Addressed by Data Pointer Přesun obsahu střadače do vnější paměti dat adresované ukazatelem dat Operandy: DPTR Ukazatel dat A Střadač Strojový kód: |11110000| 7 0 Činnost: (DPTR) <- (A) Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce přesune obsah střadače do vnější paměti dat na adresu, kterou obsahuje ukazatel dat. Vyšší řády adresy se z ukazatele dat přesunou do portu 2, nižší řády adresy se přesunou do portu 0.

MOVX @Rr,A Move Accumulator to External Memory Addressed by Register Přesun obsahu střadače do vnější paměti dat adresované registrem Operandy: Rr Registr 0 <= r <= 1 A Střadač Strojový kód: |1111001r| 7 0 Činnost: (Rr) <- (A) Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce přesune obsah střadače do vnější paměti dat na adresu, kterou obsa huje registr r a registr speciálních funkcí P2. Slabika vyšších řádů adresy je v registru P2, slabika nižších řádů adresy je v registru r.

MOVX A,@DPTR Move External Memory Addressed by Data
Pointer to Accumullator
Přesun obsahu slabiky z vnější paměti dat
adresované ukazatelem dat do střadače
Operandy: A Střadač DPTR Ukazatel dat Strojový kód: |11100000| 7 0 Činnost: A <- ((DPTR)) Slabik: 1 Cyklů: 2 Příznaky: P Popis: Instrukce přesune obsah slabiky z vnější paměti dat adresované ukazatelem dat do střadače.
Slabika vyšších řádů ukazatele dat se přesune do portu 2, slabika nižších řádů ukazatele dat se přesune do portu 0.

MOVX A,@Rr Move External Memory Addressed by Register
to Accumulator
Přesun obsahu slabiky z vnější paměti dat
adresované registrem do střadače
Operandy: A Střadač
Rr Registr 0<= r <= 1 Strojový kód: |1110001r| 7 0 Činnost: A <- ((Rr)) Slabik: 1 Cyklů: 2 Příznaky: P Popis: Instrukce přesune obsah slabiky z vnější paměti dat adresované obsahem registru r a obsahem registru speciálních funkcí P2 do střadače. Slabika vyšších řádů adresy je v registru P2, slabika nižších řádŮ adresy je v registru r.

MOVC A,@A+DPTR Move Code Memory Offset from Data Pointer
to Accumullator
Přesun kódové slabiky ze složené adresy do střadače
Operandy: A Střadač DPTR Ukazatel dat Strojový kód: |10010011| 7 0 Činnost: A <- ((A) + (DPTR)) Slabik: 1 Cyklů: 2 Příznaky: P Popis: Instrukce sečte obsah střadače a obsah ukazatele dat. Tento součet považuje za adresu v paměti programu. Obsah této adresy - tj. kódovou slabibu - přesune do střadače. Vyšší řády adresy se přesunou do portu 2, niožší řády adresy se přesunou do portu P0.

MOVC A,@A+PC Move Code Memory Offset from Program
Counter to Accumullator
Přesun kódové slabiky ze složené adersy
do střadače
Operandy: A Střadač
PC Programový čítač
Strojový kód: |10000011| 7 0 Činnost: PC <- (PC) + 1 A <- ((A) + (PC)) Slabik: 1 Cyklů: 2 Příznaky: P Popis: Instrukce sečteobsah střadače s obsahem inkrementovaného programového čítače. Tento součet považuje za adresu paměti programu. Obsah této adresy tj. kódovou slabiku - přesune do střadače. Vyšší řády adresy se přesunou do portu 2, nižší řády adresy se přesunou do portu 0.

Podrobný popis instrukcí MOV pro přesun dat 16 bitů

MOV DPTR,#data Move Immediate Data to Data Pointer
Přesun přímých dat do ukazatele dat
Operandy: DPTR Ukazatel dat

data 0 <= data = 65535
Strojový kód: |10010000|data-v.ř.|data-n.ř.| 7 0 7 0 7 0 Činnost: DPTR <- data Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce přesune hodnotu šestnáctibitových přímých dat do ukazatele dat

Podrobný popis instrukcí MOV pro přesun dat 1 bit

MOV C,bit adr Move Bit to Carry Flag Přesun bitu do příznaku přenosu Operandy: C Příznak přenosu bit adr 0 <= bit adr <= 255 Strojový kód: |10100010| bit adr| 7 0 7 0 Činnost: C <- (bit adr) Slabik: 2 Cyklů: 1 Příznaky: C Popis: Instrukce přesune hodnotu bitu z určené bitové adresy do bitu příznaku přenosu

---

MOV bit adr,c Move Carry Flaf to Bit Přesun příznaku přenosu do bitu Operandy: bit adr 0 <= bit adr <= 255 C Příznak přenosu Strojový kód: |10010010| bit adr| 7 0 7 0 Činnost: bit adr <- C Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce přesune hodnotu příznaku přenosu na určenou bitovou adresu

Podrobný popis instrukcí pro výměnu dat

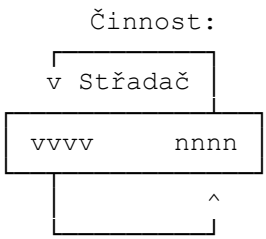
XCH A,@Rr Exchange Indirect Address with Accumulator
Výměna obsahu nepřímé adresy s obsahem střadače
Operandy: A Střadač
Rr Registr 0 <= r = 1 Strojový kód: |1100011r| 7 0 Činnost: dočasný reg. <- ((Rr)) (Rr) <- (A) A <- (dočasný reg.) Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce vymění obsah paměťové buňky vnitřní paměti dat adresované registrem r s obsahem střadače.

XCH A,Rr Exchange Register with Accumulator
Výměna obsahu registru s obsahem střadače
Operandy: A Střadač
Rr Registr 0 <= r <= 7 Strojový kód: |11001rrr| 7 0 Činnost: dočasný reg. <- (Rr) Rr <- (A) A <- (dočasný reg.) Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce vymění obsah registru r s obsahem střadače.

XCH A,data adr Exchange Memory with Accumulator Výměna obsahu datové adresy s obsahem střadače
Operandy: A Střadač data adr 0 <= data adr <= 255 Strojový kód: |11000101|data adr| 7 0 7 0 Činnost: dočasný reg. <- (data adr) data adr <- (A) A <- (dočasný reg.) Slabik: 2 Cyklů: 1 Příznaky: P Popis: Instrukce vymění obsah určené datové adresy s obsahem střadače.

XCHD A,@Rr Exchange Low Nibbles (Digits) of Indirect Addres with Accumulator Výměna nižších řádů (číslic) mezi obsahem nepřímé adresy a střadačem. Operandy: A Střadač Rr Registr 0 <= r <= 1 Strojový kód: |1101011r| 7 0 Činnost: dočasný reg. <- $((Rr))_{0-3}$ $(Rr)_{0-3}$ <- $(A)_{0-3}$ A_{0-3} <- (dočasný reg.) Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce vymění obsah nižších řádů (bity 0-3) buňky vnitřní paměti dat adresované registrem r s obsahem nižších řádů (bity 0 až 3) střadače. Vyšší řády (bity 4 až 7) jsou ponechány beze změny.

SWAP A Exchange Nibbles in Accumulator
Výměna nižších a vyšších řádů ve střadači
Operandy: A Střadač
Strojový kód: |11000100| 7 0



Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce provede výměnu nižších (bity 0 až 3) a vyšších (bity 4 až 7) řádů střadače.

Podrobný popis instrukcí pro nulování

CLR A Clear Accumulator
Vynulování obsahu střadače
Operandy: A Střadač
Strojový kód: |11100100| 7 0 Činnost: A <- 0 Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce vynuluje obsah střadače

---

CLR C Clear Carry Flag Vynulování příznaku přenosu Operandy: C Příznak přenosu Strojový kód: |11000011| 7 0 Činnost: C <- 0 Slabik: 1 Cyklů: 1 Příznaky: C Popis: Instrukce nastaví příznak přenosu do logické hodnoty 0.

CLR bit adr Clear Bit Vynulování bitu Operandy: bit adr 0 <= bit adr <=255 Strojový kód: |11000010| bit adr| 7 0 7 0 Činnost: bit adr <- 0 Slabik: 2 Cyklů: 1 Příznaky: - Popis: Instrukce nastaví bit určený bitovou adresou do logické hodnoty 0.

Podrobný popis instrukcí pro nastavení

SETB C Set Carry Flag Nastavení příznaku přenosu Operandy: C Příznak přenosu Strojový kód: |11010011| 7 0 Činnost: C <- 1 Slabik: 1 Cyklů: 1 Příznaky: C Popis: Instrukce nastavuje příznak přenosu do logické hodnoty 1

SETB bit adr Set Bit Nastavení bitu Operandy: bit adr 0 <= bit adr <= 255 Strojový kód: |11010010| bit adr| 7 0 7 0 Činnost: bit adr <- 1 Slabik: 2 Cyklů: 1 Příznaky: - Popis: Instrukce nastavuje bit určený bitovou adresou do logické hodnoty 1.

Aritmetické operace Pro jednoduché aritmetické úlohy , tj. sčítání, odčítání, násobení a dělení má 8051 tyto instrukce: - sčítání ADD A,<operand> - sčítání s přičtením příznaku C ADDC A,<operand> - odečítání s odečtením příznaku SUBB A,<operand> - zvyš obsah o jedničku INC <operand> - sniž obsah o jedničku DEC <operand> - násobení MUL AB - dělení DIV AB Instrukce sčítání ADD přičte k obsahu akumulátoru zadaný operand. Instrukce ADDC přičte navíc k výsledku ještě obsah C. Příklad: Sečtěte dvě 16-ti bitová čísla a výsledek uložte na místo prvního z nich. První z čísel je uloženo na adrese 20H a 21H, druhé číslo na adresách 22H a 23H v následující podobě: 20H 21H 22H 23H

vyšší řády	nižší řády	vyšší řády	nižší řády

OPER1 EQU 20H ;uložení prvního operandu OPER2 EQU 22H ;uložení druhého operandu MOV R0,#OPER1+1 ;načtení adresy spodního ;bajtu prvního operandu MOV R1,#OPER2+1 ;načtení adresy spodního ;bajtu druhého operandu MOV B,#2 CLR C ;nulování příznaku C CYKL: MOV A,@R0 ;přenos z prvního op. ADDC A,@R1 ;přičtení druhého op. MOV @R0,A ;uložení na místo DEC R0 ;další bajty operandů DEC R1 DJNZ B,CYKL ;opakuj pro všechny bajty ;operandů Instrukce pro odečítání SUBB odečítá od akumulátoru daný operand a příznak přetečení C. Neexistuje ekvivalent instrukce pro odečítání bez příznaku C !!! Instrukce INC a DEC slouží k přičítání jedničky a odečítání jedničky k danému operandu. Tyto instrukce (na rozdíl od instrukcí sečítání a odečítání) nenastavují příznaky v PSW registru !!! Po provedení těchto instrukcí nelze tedy provádět test příznaku C (přenos) ani OV (přetečení). Instrukční soubor 8051 zahrnuje speciální instrukce MUL a DIV pro násobení a dělení operandů. Instrukce MUL provádí součin hodnot v akumulátoru a registru B. Vypočtený výsledek (16-ti bitový) se uloží tak, že v A leží spodní bajt a v B horní bajt výsledku. Instrukce DIV dělí akumulátor obsahem registru B.

Celočíselný výsledek dělení je uložen v A, zbytek po dělení je uložen v registru B. Příklad: Instrukci DIV použijte k rozdělení dvou BCD číslic, uložených v registru A na dvě samostatné číslice, z nich jedna bude v registru A, druhá v registru B. MOV A,#15H ;v A jsou dvě BCD číslice 1 a 5 MOV B,#16 DIV AB Podělením registru A šestnácti získáme ve spodních čtyřech bitech registru A první BCD číslici (1) a ve spodních čtyřech bitech registru B druhou BCD číslici (5).

Pordobný popis instrukcí pro aritmetické operace

INC @Rr Increment Indirect Address Zvýšení obsahu nepřímé adresy o jedničku Operandy: Rr Registr 0 <= r <= 1 Strojový kód: |0000011r| 7 0 Činnost: (Rr) <- ((Rr)) + 1 Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce přičítá jedničku k obsahu paměťové buňky vnitřní paměti dat adresované obsahem registru r. Výsledek se umístí do adresované paměťové buňky.

INC A Increment Accumulator
Zvýšení obsahu střadače o jedničku
Operandy: A Střadač
Strojový kód: |00000100| 7 0 Činnost: A <- (A) + 1 Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce přičte jedničku k obsahu střadače. Výsledek se umístí do střadače.

INC DPTR Increment Data Pointer Zvýšení obsahu ukazatele dat o jedničku Operandy: DPTR Ukazatel dat Strojový kód: |10100011| 7 0 Činnost: DPTR <- (DPTR) + 1 Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce připočte k obsahu šestnáctibitového ukazatele dat jedničku. Výsledek se umístí do ukazatele dat.

INC Rr Increment Register
Zvýšení obsahu registru o jedničku
Operandy: Rr Registr 0 <= r <= 7
Strojový kód: |00001rrr| 7 0 Činnost: Rr <- (Rr) + 1 Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce přičte jedničku k obsahu registru r. Výsledek se umístí do registru r.

INC data adr Increment Memory
Zvýšení obsahu datové adresy o jedničku
Operandy: data adr 0 <= data adr <= 255
Strojový kód: |00000101|data adr| 7 0 7 0 Činnost: data adr <- (data adr) + 1 Slabik: 2 Cyklů: 1 Příznaky: - Popis: Instrukce přičte jedničku k obsahu určené datové adresy. Výsledek uloží do adresovaného paměťového místa.

ADD A,#data Add Immediate Data
Přičtení přímých dat ke střadači
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |00100100| data | 7 0 7 0 Činnost: A <- (A) + data Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte hodnotu osmibitových přímých dat s obsahem střadače. Výsledek umístí do střadače.

ADD A,@Rr Add Indirect Address
Přičtení obsahu nepřímé adresy ke střadači
Operandy: A Střadač
Rr Registr 0 <= r = 1 Strojový kód: |0010011r| 7 0 Činnost: A <- (A) + ((Rr)) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte obsah buňky vnitřní paměti dat adresované registrem r s obsahem střadače. Výsledek se umístí do střadače.

ADD A,Rr Add Register

Přičtení obsahu registru ke střadači
Operandy: A Střadač
Rr Registr 0 <= r <= 7 Strojový kód: |00101rrr| 7 0 Činnost: A <- (A) + (Rr) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte obsah registru r s obsahem střadače. Výsledek umístí do střadače.

ADD A,data adr Add Memory
Přičtení obsahu datové adresy ke střadači
Operandy: A Střadač
data adr 0 <= data adr <= 255 Strojový kód: |00100101|data adr| 7 0 7 0 Činnost: A <- (A) + (data adr) Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte obsah určené datové adresy s obsahem střadače. Výsledek se umístí do střadače.

ADDC A,#data Add Carry Plus Immediate Data to Accumulator
Přičtení příznaku přenosu a přímých dat ke střadači
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |00110100| data | 7 0 7 0 Činnost: A <- (A) + C + data Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte hodnotu příznaku přenosu (0 nebo 1) s obsahem střadače. K tomuto mezivýsledku přičte hodnotu osmibitových přímých dat. Výsledný součet všech tří hodnot se umístí do střadače a aktualizuje se stav příznaku přenosu.

ADDC A,@Rr Add Carry Plus Indirect Address to Accumulator
Přičtení příznaku přenosu a obsahu nepřímé adresy ke střadači
Operandy: A Střadač
Rr Registr 0 <= r = 1 Strojový kód: |0011011r| 7 0 Činnost: A <- (A) + C + ((Rr)) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte hodnotu příznaku přenosu (0 nebo 1) s obsahem střadače. K tomuto mezivýsledku přičte obsah buňky vnitřní paměti dat adresované registrem r. Výsledný součet všech tří hodnot se umístí do střadače a aktualizuje se stav příznaku přenosu.

ADDC A,Rr Add Carry Plus Register to Accumulator
Přičtení příznaku přenosu a obsahu registru ke střadači
Operandy: A Střadač
Rr Registr 0 <= r <= 7 Strojový kód: |00111rrr| 7 0 Činnost: A <- (A) + C + (Rr) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte hodnotu příznaku přenosu (0 nebo 1) s obsahem střadače. K tomuto mezivýsledku přičte obsah registru r. Výsledný součet všech tří hodnot se umístí do střadače a aktualizuje se stav příznaku přenosu.

ADDC A,data adr Add Carry Plus Memory to Accumulator
Přičtení příznaku přenosu a obsahu datové adresy ke střadači
Operandy: A Střadač
data adr 0 <= data adr <= 255 Strojový kód: |00110101|data adr| 7 0 7 0 Činnost: A <- (A) + C + (data adr) Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce sečte hodnotu příznaku přenosu (0 nebo 1) s obsahem střadače. K tomuto mezivýsledku se přičte obsah datové adresy. Výsledný součet všech tří hodnot se umístí do střadače a aktualizuje se stav příznaku přenosu.

DEC @Rr Decrement Indirect Address Snížení obsahu nepřímé adresy o jedničku Operandy: Rr Registr 0 <= r <= 1 Strojový kód: |0001011r| 7 0 Činnost: (Rr) <- ((Rr)) - 1 Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce odečítá jedničku od obsahu paměťové buňky vnitřní paměti dat adresované obsahem registru r. Výsledek se umístí do adresované paměťové buňky.

DEC A Decrement Accumulator
Snížení obsahu střadače o jedničku
Operandy: A Střadač

Strojový kód: |00010100| 7 0 Činnost: A <- (A) - 1 Slabik: 1 Cyklů: 1 Příznaky: P Popis: Instrukce odečte jedničku od obsahu střadače. Výsledek se umístí do střadače.

DEC Rr Decrement Register
Snížení obsahu registru o jedničku
Operandy: Rr Registr 0 <= r <= 7
Strojový kód: |00011rrr| 7 0 Činnost: Rr <- (Rr) - 1 Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce odečte jedničku od obsahu registru r. Výsledek se umístí do registru r.

DEC data adr Decrement Memory
Snížení obsahu datové adresy o jedničku
Operandy: data adr 0 <= data adr <= 255
Strojový kód: |00010101|data adr| 7 0 7 0 Činnost: data adr <- (data adr) - 1 Slabik: 2 Cyklů: 1 Příznaky: - Popis: Instrukce odečte jedničku od obsahu určené datové adresy. Výsledek se umístí do specifikované datové adresy.

SUBB A,#data Subtract Immediate Data from Accumulator with Borrow
Odečtení přímých dat od obsahu střadače s výpujčkou
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |10010100| data | 7 0 7 0 Činnost: A <- (A) - C - data Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce odečte od obsahu střadače hodnotu příznaku přenosu a hodnotu přímých dat. Výsledek se umístí do střadače.

SUBB A,@Rr Subtract Indirect Address from Accumulator with Borrow
Odečtení obsahu nepřímé adresy od střadače s výpujčkou
Operandy: A Střadač
Rr Registr 0 <= r = 1 Strojový kód: |1001011r| 7 0 Činnost: A <- (A) - C - ((Rr)) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce odečte od obsahu střadače hodnotu příznaku přenosu a obsah paměťové buňku vnitřní paměti dat adresované registrem r. Výsledek se umístí do střadače.

SUBB A,Rr Subtract Register from Accumulator with Borrow
Odečtení obsahu registru od obsahu střadače s výpojčkou
Operandy: A Střadač
Rr Registr 0 <= r <= 7 Strojový kód: |10011rrr| 7 0 Činnost: A <- (A) - C - (Rr) Slabik: 1 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce odečte od obsahu střadače hodnotu příznaku přenosu a obsah registru r. Výsledek se umístí do střadače.

SUBB A,data adr Subtract Memory from Accumulator with Borrow
Odečtení obsahu datové adresy od obsahu střadače s výpujčkou
Operandy: A Střadač
data adr 0 <= data adr <= 255 Strojový kód: |10010101|data adr| 7 0 7 0 Činnost: A <- (A) - C - (data adr) Slabik: 2 Cyklů: 1 Příznaky: C, AC, OV, P Popis: Instrukce odečte od obsahu střadače hodnotu příznaku přenosu a obsah určené datové adresy. Výsledek se umístí do střadače.

MUL AB Multiply Accumulator By B
Násobení obsahu střadače obsahem registru B
Operandy: AB Dvojice registrů pro násobení
nebo dělení
Strojový kód: |10100100| 7 0 Činnost: AB <- (A) * (B) Slabik: 1 Cyklů: 4 Příznaky: C, OV, P Popis: Instrukce vynásobí obsah střadače obsahem registru B. Oba operandy jsou pokládány za celá čísla bez znaménka. Nižší řády součinu jsou umístěny ve střadači, vyšší řády součinu jsou umístěny v registru B.

Příznak přenosu se vždy vynuluje. Jsou-li vyšší řády součinu nenulové, nastaví se příznak přetečení do logické hodnoty 1; jinak se příznak přetečení vynuluje.

DIV AB Divide Accumulator by B
Dělení obsahu střadače obsahem registru B
Operandy: AB Dvojice registrů pro násobení
nebo dělení Strojový kód: |10000100| 7 0 Činnost: AB <- (A) / (B) Slabik: 1 Cyklů: 4
Příznaky: C, OV, P Popis: Instrukce dělí obsah střadače obsahem registru B. Oba operandy jsou pokládány za celá čísla bez znaménka. Po dělení obsahuje střadač podíl a registr B zbytek.
Příznak přenosu se vždy vynuluje. Dělení nulou nastavuje příznak přetečení do logické hodnoty 1; jinak se příznak přetečení nuluje.

DA A Decimal Adjust Accumulator
Desítková úprava obsahu střadače
Operandy: A Střadač
Strojový kód: |11010100| 7 0 Činnost: Úprava obsahu střadače (A) na desítkový tvar (viz popis)
Slabik: 1 Cyklů: 1 Příznaky: C, P Popis: Instrukce upravuje obsah střadače tak, aby odpovídal zhuštěnému tvaru dvojkově kódovaných desítkových číslic (BCD). Instrukce DA může být použita pouze po instrukcích sčítaní dvou čísel, která jsou v kódu BCD. Je-li příznak pomocného přenosu AC v logické hodnotě 1, nebo obsahují-li nižší řády (bity 0 až 3) střadače číslici větší než 9, přičte se k obsahu střadače číslo 6. Je-li příznak přenosu před nebo po tomto přičtení v logické hodnotě 1, nebo obsahují-li vyšší řády (bity 4 až 7) číslici větší než 9, přičte se k obsahu střadače číslo 60H. Střadač a příznak přenosu obsahují konečné hodnoty po provedené desítkové úpravě.

Logické operace
V instrukčmím souboru 8051 jsou následující instrukce pro logické operace:
- logické násobení ANL <oper1>,<oper2>
- logické sečítání ORL <oper1>,<oper2>
- logický exkluzivní součet XRL <oper1>,<oper2>

Uvedené instrukce provádějí příslušnou logickou operaci mezi dvěma operandy <oper1>,<oper2> po jednotlivých bitech a výsledek se uloží do prvního operandu <oper1>.
Typickým příkladem je vynulování jednotlivých bitů operandu. Používá se v těch případech, kdy je třeba část operandu vynulovat.
Příklady:

```
;nulování bitu 0..3 registru B
ANL B,#11110000b
;nulování bitů 0,2,4,6 registru A
ANL A,#10101010B
nulování celého registru B
ANL B,#o
```

Této operaci se říká **maskování** - operand se logicky vynásobí s tzv. maskou - v našem případě 11110000B, 10101010B nebo 0. Ty bity prvního opeandu které se vynásobí s nulou v příslušném bitu druhého operandu (masce), jsou vynulovány (vymaskovány).
Příklad:
V registru R0 máme dvě BCD číslice. Úkolem programu bude tyto dvě číslice uložit do registrů R1 a R2, každou zvlášť. Tedy například:
R0 = 15H -> R1 = 01H, R2 = 05H
Poznámka: BCD tvar číslic je způsob uchování číslic ve tvaru, kdy v jednom bytu jsou umístěny dvě číslice (v našem případě 1 a 5).

```
MOV A,R0 ;přesun do A
ANL A,#0FH ;nulování horní části bytu
MOV R2,A ;uschovej
MOV A,R0 ;znovu přesun
```

```
 ANL A,#0F0H ;nulování spodní části bytu
 SWAP A ;zaměň horní a dolní část
 MOV R1,A ;registru A a ulož
```
Instrukce ORL provádí logický součet (po bitech operace OR) dvou operandů. Časté použití této instrukce je pro nastavení příslušných bitů daného operandu do 1.

Příklad:
```
 ;nastavení bitu 0 registru A do jedničky
 ORL A,#1
 ;nastavení bitů 5,6,7 do jedničky
 ORL A,#11100000B
```
Poznámka

Pro nastavení, nulování a negaci jednotlivých bitů existují speciální instrukce, které jsou popsány dále v této části.

Instrukce XRL provádí výhradní logický součet operandů (po bitech operace XOR) a výsledek ukládá do prvého operandu. Praktické využití této instrukce je v negaci vybraných bitů daného operandu.

Příklad:
```
 MOV A,P3 ;načtení z portu P3 do A
 ANL A,#0F0H ;nulování spodních bitů
 XRL A,#0F0H ;invertování horních bitů
 MOV P1,A ;vyslání na port P1
```
Rotace

Pro rotace jsou v instrukčním souboru 8051 následující instrukce:

- rotace doleva RL A
- rotace doleva přes C RLC A
- rotace doprava RR A
- rotace doprava přes C RRC A

Rotovat operand je možno pouze v akumulátoru. Rotace mají široké použití v aritmetických programech. Ukážeme si další typické použití - vysílání tzv. **pochodující nuly** na port. Pochodující nula znamená, že na jeden port za sebou postupně vyšleme hodnoty:
```
11111110
11111101
11111011
11110111
11101111
11011111
10111111
01111111
```
Jak je vidět z tabulky, hodnoty, které se vysílají na port se liší pouze v pozici nuly - nula **pochoduje** přes celý port. Tato technika se používá při obsluze klávesnice.
```
 MOV R0,#8 ;počítadlo cyklů
 MOV A,#11111110B ;první hodnota do A
CYKL: MOV P1,A ;vyslání na port
 RL A ;rotace akumulátoru doleva
 DJNZ R0,CYKL ;celkem 8x
```
Další příklad ukazuje způsob rotace 16-ti bitového operandu doleva. K této rotaci se používají instrukce s přenosem do C bitu. Uvedený posuv se nazývá **logický posuv** a spočívá v tom, že bity D0 - D14 se posunou o jedno místo doleva, do bitu D0 se přesune bit C a bit D15 se přesune do C.
```
 ;rotace 16-ti bitového čísla, uloženého v R0 a R1
 MOV A,R0 ;načtení dolních 8 bitů
 RLC A ;rotace doleva s přenosem do C
 XCH A,R1 ;uschovej a načti horních 8 bitů
 RLC A ;rotuj horních 8 bitů
 XCH A,R1 ;ulož horních 8 bitů
 MOV R0,A ;ulož spodních 8 bitů
```

Booleovský procesor

Architekrura 8051 umožňuje pracovat přímo s jednotlivými bity. Instrukční soubor obsahuje instrukce, které umožňují přímou adresaci jednotlivých bitů (ať už ve vnitřní RAM nebo v oblasti SFR registrů). Jsou to tyto instrukce:

- přesun MOV C,<bit>
- komplementaci CPL <bit>
- nulování CLR <bit>
- nastavení SETB <bit>
- logický součet ORL C,<bit>
- logický součet s negovaným bitem ORL C,/<bit>
- logický součin ANL C,<bit>
- logický součin s negovaným bitem ANL c,/<bit>

Dále pak je možno stav kteréhokoliv bitu testovat na hodnotu **true** nebo **false** instrukcemi podmíněných skoků:

- skok, je-li bit roven jedné JB <adr>
- skok, je-li bit roven nule JNB <adr>
- skok, je-li bit roven jedné s následným nulováním bitu JBC <adr>

Uvedené instrukce představují v praxi velmi silný nástroj pro řešení konkrétních aplikací.

Příklad:

Pokud je na vývodu T0 log.0, je na vývodu T1 stejná hodnota, jako je na vývodu P1.0. Pokud je na vývodu T0 log.1, hodnota na výstupu T1 se mění periodicky (střídání 0 a 1).

```
TEST: JB T0,KOMPL ;test na stav vývodu T0
MOV C,P1.0 ;je nulový - načti P1.0
MOV T1,C ;a přenes na T1
SJMP TEST ;návrat na začátek
KOMPL: CPL T1 ;komplementace vývodu T1
SJMP TEST ;návrat na začátek
```

Příklad:

Synchronizace na přicházející signál. Úkolem je zjistit okamžik sestupné hrany (přechod z jedničky do nuly) na vývodu P2.5. Celý problém řeší zápis jedinné instrukce:

```
JB P2.5,$
```

nebo

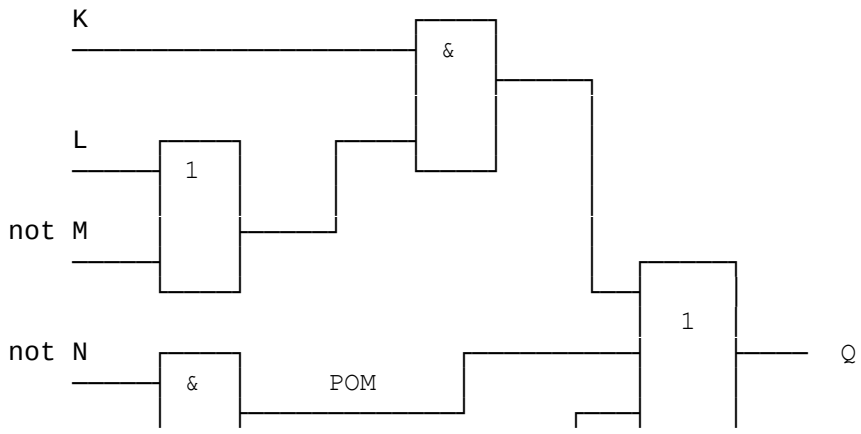
```
ZDE: JB P2.5,ZDE
```

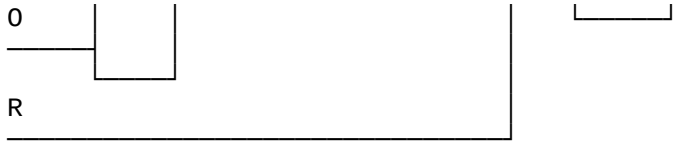
Oba zápisy jsou ekvivalentní, význam znaku **$** byl vysvětlen v kapitole 1. Činnost procesoru při provádění této instrukce je jednoduchá - pokud je na vývodu P2.5 logická jednička, provádí se skok na tutéž instrukci, tedy čeká se na okamžik, kdy dojde ke změně z log.1 na log.0. Podobně se dá realizovat čekání na vzestupnou hranu, celý puls, atd..

Příklad:

Další z možnosti využití instrukcí pracujících s bity je ve vyhodnocení booleovského výrazu. Představme si, že máme vyhodnotit booleovskou funkci:

Q= K . (L + not M) + (not N . O) + R





Řešení:

```
K BIT 0 ;definice jednotlivých
L BIT 1 ;proměnných v oblasti bitově
M BIT 2 ;adresovatelné RAM
N BIT 3
O BIT 4
R BIT 5
Q BIT 6
POM BIT 7
MOV C,O ;výpočet (not N.O)
ANL C,/N
MOV POM,C ;úschova mezivýsledku do POM
MOV C,L ;výpočet (not M + L)
ORL C,/M
ANL C,K ;and K
ORL C,POM ;or POM
ORL C,R ;or R
MOV Q,C ;výsledek ulož do Q
END ;konec programu
```

Podrobný popis instrukcí logických

CPL A Complement Accumulator
Negace obsahu střadače
Operandy: A Střadač
Strojový kód: |11110100|
7 0
Činnost: A <- NOT (A)
Slabik: 1
Cyklů: 1
Příznaky: -
Popis: Instrukce vynuluje každý bit střadače, který obsahuje logickou hodnotu 1 a nastaví do logické hodnoty 1 každá bit střadače, který obsahuje logickou hodnotu 0.

CPL C Complement Carry Flag
Negace příznaku přenosu
Operandy: C Příznak přenosu
Strojový kód: |10110011|
7 0
Činnost: C <- NOT C
Slabik: 1
Cyklů: 1
Příznaky: C
Popis: Instrukce nastavuje příznak přenosu do logické hodnoty 1, obsahoval-li logickou hodnotu 0, a nuluje příznak přenosu, obsahoval-li logickou hodnotu 1.

CPL bit adr Complement Bit
Negace bitu

Operandy: bit adr 0 <= bit adr <= 255
Strojový kód: |10110010| bit adr|
7 0 0 7
Činnost: bit adr <- NOT (bit adr)
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce nastaví bit určený bitovou adresou do logické hodnoty 1, obsahoval-li logickou hodnotu 0, a nuluje adresovaný bit, obsahoval-li logickou hodnotu 1.

ANL A,#data Logical AND Immediate Data to Accumulator
Logický součin přímých dat s obsahem střadače
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |01010100| data |
7 0 7 0
Činnost: A <- (A) AND data
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součin hodnoty osmibitových přímých dat s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ANL A,@Rr Logical AND Indirec Adress to Accumulator
Logický součin nepřímé adresy s obsahem
střadače
Operandy: A Střadač
Registr 0 <= r <= 1
Strojový kód: |0101011r|
7 0
Činnost: A <- (A) AND ((Rr))
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součin obsahu paměťové buňky vnitřní paměti dat adresované registrem r s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ANL A,Rr Logical AND Register to Accumulator
Logický součin obsahu registru s obsahem
střadače
Operandy: A Střadač
Registr 0 <= r <= 7
Strojový kód: |01011rrr|
7 0
Činnost: A <- (A) AND (Rr)
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součin obsahu registru r s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ANL A,data adr Logical AND Memory to Accumulator

Logický součin obsahu datové adresy s obsahem
střadače
Operandy: A Střadač
data adr 0 <= data adr <= 255
Strojový kód: |01010101|data adr|
7 0 7 0
Činnost: A <- (A) AND (data adr)
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součin obsahu datové adresy s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ANL C,bit adr Logical AND Bit to Carry Flag
Logický součin obsahu bitové adresy s příznakem přenosu
Operandy: C Příznak přenosu
bit adr 0 <= bit adr <= 255
Strojový kód: |10000010| bit adr|
7 0 7 0
Činnost: C <- (C) AND (bit adr)
Slabik: 2
Cyklů: 2
Příznaky: C
Popis: Instrukce provede logický součin hodnoty příznaku přenosu s hodnotou bitu adresovaného bitovou adresou. Mají-li oba bity logickou hodnotu 1, je výsledek rovněž logická hodnota 1; jinak je výsledek logická hodnota 0. Výsledek se umístí do bitu příznaku přenosu.

ANL C,/bit adr Logical AND Complement of Bit to Carry Flag
Logický součin negace obsahu bitové adresy s příznakem přenosu
Operandy: C Příznak přenosu
bit adr 0 <= bit adr <= 255
Strojový kód: |10110010| bit adr|
7 0 7 0
Činnost: C <- (C) AND NOT(bit adr)
Slabik: 2
Cyklů: 2
Příznaky: C
Popis: Instrukce provede logický součin hodnoty příznaku přenosu s negací hodnoty bitu adresovaného bitovou adresou. Výsledek bude mít logickou hodnotu 1, bude-li mít příznak přenosu logickou hodnotu 1 a adresovaný bit logickou hodnotu 0. Výsledek se umístí do bitu příznaku přenosu. Hodnota bitu adresovaného bitovou adresou se nezmění.

ANL data adr,#data Logical AND Immediate Data to Memory
Logický součin přímých dat s obsahem datové adresy.
Operandy: data adr 0 <= data adr <= 255
data -256 <= data <= +255
Strojový kód: |01010011|data adr| data |
7 0 7 0 0 7
Činnost: data adr <- (data adr) AND data
Slabik: 3
Cyklů: 2
Příznaky: -
Popis: Instrukce provede logický součin hodnoty osmibitových přímých dat s obsahem určené datové adresy. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v

logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do paměti na specifikovanou datovou adresu.

ANL data adr,A Logical AND Accumulator to Memory Logický součin obsahu střadače s obsahem datové adresy
Operandy: data adr 0 <= data adr <= 255
A Střadač
Strojový kód: |01010010|data adr|
7 0 7 0
Činnost: data adr <- (data adr) AND A
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce provede logický součin obsahu střadače s obsahem specifikované datové adresy. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do paměti na určenou datovou adresu.

ORL A,#data Logical OR Immediate Data to Accumulator
Logický součet přímých dat s obsahem střadače
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |01000100| data |
7 0 7 0
Činnost: A <- (A) OR data
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součet hodnoty osmibitových přímých dat s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ORL A,@Rr Logical OR Indirec Address to Accumulator Logický součet nepřímé adresy s obsahem střadače
Operandy: A Střadač
Registr 0 <= r <= 1
Strojový kód: |0100011r|
7 0
Činnost: A <- (A) OR ((Rr))
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součet obsahu paměťové buňky vnitřní paměti dat adresované registrem r s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ORL A,Rr Logical OR Register to Accumulator
Logický součet obsahu registru s obsahem
střadače
Operandy: A Střadač
Registr 0 <= r <= 7
Strojový kód: |01001rrr|
7 0
Činnost: A <- (A) OR (Rr)
Slabik: 1
Cyklů: 1

Příznaky: P
Popis: Instrukce provede logický součet obsahu registru r s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ORL A,data adr Logical OR Memory to Accumulator
Logický součet obsahu datové adresy s obsahem
střadače
Operandy: A Střadač
data adr 0 <= data adr <= 255
Strojový kód: |01000101|data adr|
7 0 7 0
Činnost: A <- (A) OR (data adr)
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede logický součet obsahu datové adresy s obsahem střadače. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do střadače.

ORL C,bit adr Logical OR Bit to Carry Flag
Logický součet obsahu bitové adresy s příznakem přenosu
Operandy: C Příznak přenosu
bit adr 0 <= bit adr <= 255
Strojový kód: |01110010| bit adr|
7 0 7 0
Činnost: C <- (C) OR (bit adr)
Slabik: 2
Cyklů: 2
Příznaky: C
Popis: Instrukce provede logický součet hodnoty příznaku přenosu s hodnotou bitu adresovaného bitovou adresou. Mají-li jeden nebo oba bity logickou hodnotu 1, je výsledek rovněž logická hodnota 1; jinak je výsledek logická hodnota 0. Výsledek se umístí do bitu příznaku přenosu.

ORL C,/bit adr Logical OR Complement of Bit to Carry Flag
Logický součet negace obsahu bitové adresy s příznakem přenosu
Operandy: C Příznak přenosu
bit adr 0 <= bit adr <= 255
Strojový kód: |10100000| bit adr|
7 0 7 0
Činnost: C <- (C) OR NOT(bit adr)
Slabik: 2
Cyklů: 2
Příznaky: C
Popis: Instrukce provede logický součet hodnoty příznaku přenosu s negací hodnoty bitu adresovaného bitovou adresou. Výsledek bude mít logickou hodnotu 1, bude-li mít příznak přenosu logickou hodnotu 1 nebo adresovaný bit logickou hodnotu 0. Výsledek se umístí do bitu příznaku přenosu. Hodnota bitu adresovaného bitovou adresou se nezmění.

ORL data adr,#data Logical OR Immediate Data to Memory
Logický součet přímých dat s obsahem datové adresy.
Operandy: data adr 0 <= data adr <= 255
data -256 <= data <= +255
Strojový kód: |01000011|data adr| data |
7 0 7 0 0 7
Činnost: data adr <- (data adr) OR data

Slabik: 3
Cyklů: 2
Příznaky: -
Popis: Instrukce provede logický součet hodnoty osmibitových přímých dat s obsahem určené datové adresy. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do paměti na specifikovanou datovou adresu.

ORL data adr,A Logical OR Accumulator to Memory Logický součet obsahu střadače s obsahem datové adresy
Operandy: data adr 0 <= data adr <= 255
A Střadač
Strojový kód: |01000010|data adr|
7 0 7 0
Činnost: data adr <- (data adr) OR A
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce provede logický součet obsahu střadače s obsahem specifikované datové adresy. Bit n výsledku bude nastaven do logické hodnoty 1, bude-li bit n jednoho nebo obou operandů v logické hodnotě 1; jinak je bit n výsledku nulován. Výsledek se umístí do paměti na určenou datovou adresu.

XRL A,#data Logical Exclusive OR Immediate Data to Accumulator
Součet modulo 2 přímých dat s obsahem střadače
Operandy: A Střadač
data -256 <= data <= +255
Strojový kód: |01100100| data |
7 0 7 0
Činnost: A <- (A) XOR data
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede součet modulo 2 hodnoty osmibitových přímých dat s obsahem střadače. Bit n výsledku bude vynulován, bude-li mít bit n střadače a přímých dat stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do střadače.

XRL A,@Rr Logical Exclusive OR Indirec Address to Accumulator
Součet modulo 2 nepřímé adresy s obsahem střadače
Operandy: A Střadač
Registr 0 <= r <= 1
Strojový kód: |0110011r|
7 0
Činnost: A <- (A) XOR ((Rr))
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce provede součet modulo 2 obsahu paměťové buňky vnitřní paměti dat adresované registrem r s obsahem střadače. Bit n výsledku bude vynulován, bude-li mít bit n střadače a bit n adresované paměťové buňky stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do střadače.

XRL A,Rr Logical Exclusive OR Register to Accumulator
Součet modulo 2 obsahu registru s obsahem střadače
Operandy: A Střadač
Registr 0 <= r <= 7

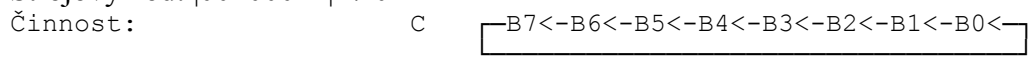
Strojový kód: |01101rrr|
7 0
Činnost: A <- (A) XOR (Rr)
Slabik: 1
Cyklů: 1
Příznaky: P
Popis: Instrukce provede součet modulo 2 obsahu registru r s obsahem střadače. Bit n výsledku bude vynulován, bude-li mít bit n střadače a bit n určeného registru stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do střadače.

XRL A,data adr Logical Exclusive OR Memory to Accumulator
Součet modulo 2 obsahu datové adresy s obsahem
střadače
Operandy: A Střadač
data adr 0 <= data adr <= 255
Strojový kód: |01100101|data adr|
7 0 7 0
Činnost: A <- (A) XOR (data adr)
Slabik: 2
Cyklů: 1
Příznaky: P
Popis: Instrukce provede součet modulo 2 obsahu určené datové adresy s obsahem střadače. Bit n výsledku bude vynulován, bude-li mít bit n střadače a bit n obsahu datové adresy stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do střadače.

XRL data adr,#data Logical Exclusive OR Immediate Data to Memory
Součet modulo 2 přímých dat s obsahem datové adresy.
Operandy: data adr 0 <= data adr <= 255
data -256 <= data <= +255
Strojový kód: |01100011|data adr| data |
7 0 7 0 0 7
Činnost: data adr <- (data adr) XOR data
Slabik: 3
Cyklů: 2
Příznaky: -
Popis: Instrukce provede součet modulo 2 hodnoty osmibitových přímých dat s obsahem určené datové adresy. Bit n výsledku bude vunulován, bude-li mít bit n přímých dat a bit n obsahu datové adresy stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do paměti dat na specifikovanou datovou adresu.

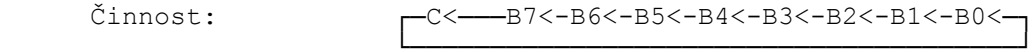
XRL data adr,A Logical Exclusive OR Accumulator to Memory Součet modulo 2 obsahu střadače s obsahem datové adresy
Operandy: data adr 0 <= data adr <= 255
A Střadač
Strojový kód: |01100010|data adr|
7 0 7 0
Činnost: data adr <- (data adr) XOR A
Slabik: 2
Cyklů: 1
Příznaky: -
Popis: Instrukce provede součet modulo 2 obsahu střadače s obsahem specifikované datové adresy. Bit n výsledku bude vunulován, bude-li mít bit n střadače a bit n obsahu datové adresy stejnou hodnotu; jinak je bit n výsledku nastaven do logické hodnoty 1. Výsledek se umístí do paměti dat na specifikovanou datovou adresu.

RL A Rotate Accumulator Left Rotační posuv obsahu střadače vlevo Operandy: A Střadač
Strojový kód: |00100011| 7 0



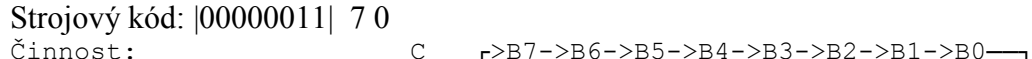
Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce posouvá každý bit střadače o jednu pozici vlevo. Nejvýznamnější bit (bit 7) se přesune do nejméně významného bitu (bit 0).

RLC A Rotate Accumulator and Carry Flag Left Rotační posuv obsahu střadače a příznaku přenosu vlevo Operandy: A Střadač Strojový kód: |00110011| 7 0



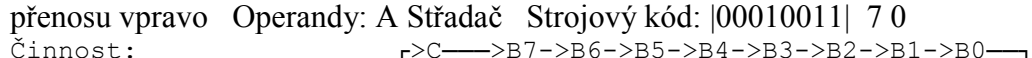
Slabik: 1 Cyklů: 1 Příznaky: C,P Popis: Instrukce posouvá každý bit střadače o jednu pozici vlevo. Nejvýznamnější bit (bit 7) se přesune do příznaku přenosu vzápětí po přesunutí příznaku přenosu do nejméně významného bitu (bit 0).

RR A Rotate Accumulator Right Rotační posuv obsahu střadače vpravo Operandy: A Střadač
Strojový kód: |00000011| 7 0



Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce posouvá každý bit střadače o jednu pozici vpravo. Nejméně významný bit (bit 0) se přesune do nejvýznamnějšího bitu (bit 7).

RRC A Rotate Accumulator and Carry Flag Right Rotační posuv obsahu střadače a příznaku přenosu vpravo Operandy: A Střadač Strojový kód: |00010011| 7 0



Slabik: 1 Cyklů: 1 Příznaky: C,P Popis: Instrukce posouvá každý bit střadače o jednu pozici vpravo. Nejméně významný bit (bit 0) se přesune do příznaku přenosu vzápětí po přesunutí hodnoty příznaku přenosu do nevýznamnějšího bitu (bit 7).

NOP No Operation
Neúčinná operace
Operandy: žádné
Strojový kód: |00000000|
7 0
Činnost: žádná operace Slabik: 1 Cyklů: 1 Příznaky: - Popis: Instrukce po dobu jednoho strojového cyklu neprovádí žádnou činnost. Program pak pokračuje sekvenčně následující instrukcí.

Používání zásobníku

Zásobník je část paměti v oblasti vnitřní RAM. Je definován svým počátkem a může ležet kdekoli ve vnitřní oblasti RAM. V oblasti SFR registrů je definován ukazatel zásobníku - registr SP. Tento registr ukazuje vždy na vrchol zásobníku. Při ukládání dat na zásobník se nejprve hodnota SP registru zvýší o 1 - zásobník *roste* směrem do oblasti vyšších adres vnitřní RAM - a potom se data uloží tam, kam ukazuje SP. Při vybírání dat ze zásobníku je postup opačný - nejprve se uloží data, adresovaná SP a potom se SP sníží o 1.

Nastavení ukazatele zásobníku a starost o to, aby nedošlo ke kolizi zásobníku s ostatními údaji v RAM je zcela na zodpovědnosti programátora. Vždy je nutno pro zásobník vyhradit dostatečně velkou část paměti tak, aby mohl pojmout návratovou adresu i toho nejvíce vnořeného podprogramu. U systému s přerušením je nutno počítat s tím, že přerušena může být libovolná část programu, takže kapacita zásobníku musí být navržena s rezervou i pro tento případ.

Po resetu mikropočítače se hodnota SP nastaví na 7.

Zásobník je využíván pro odkládání návratových adres při volání podprogramů nebo při přerušení (viz bod 7).

Pro ukládání a vybírání dat ze zásobníku slouží instrukce:

ulož PUSH <parametr>
vyjmi POP <parametr>

Nejtypičtějším použitím zásobníku je přechodné ukládání mezivýsledků pro pozdější zpracování nebo pro úschovu pracovních registrů při vstupu do procedury.

Příklad: ACALL PROC ;volání procedury

```
.
.
PROC: PUSH PSW ;úschova PSW
PUSH ACC ;úschova A
PUSH B ;úschova B
.
;nyní mohu v proceduře používat registry A,B
;před návratem z procedury je jejich obsah obnoven
.
.
POP B ;obnovení B
POP ACC ;obnovení A
POP PSW ;obnovení PSW
RET
```

Instrukce pro práci se zásobníkem

POP data adr Pop Stack to Memory
Obnovení obsahu datové adresy ze zásobníku
Operandy: data adr 0 <= data adr <= 255
Strojový kód: |11010000|data adr|
7 0 7 0
Činnost: data adr <- ((SP)) SP <- (SP) - 1 Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce umístí obsah slabiky adresované ukazatelem zásobníku na určenou datovou adresu. Obsah ukazatele zásobníku se zmenší o 1.

PUSH data adr Push Memory onto Stack Uložení obsahu datové adresy do zásobníku Operandy: data adr 0 <= data adr <= 255 Strojový kód: |11000000|data adr| 7 0 7 0 Činnost: SP <- (SP) + 1 (SP) <- data adr Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce zvýší o 1 obsah ukazatele zásobníku a uloží obsah určené datové adresy na adresu, která je v ukazateli zásobníku.

Větvení programů S některými instrukcemi pro větvení programu jsme se seznámili v předchozí kapitole. Mezi instrukce skoků patří dále:

skok, je-li C=1 JC <adr>
skok, je-li C=0 JNC <adr>
skok, je-li A=0 JZ <adr>
skok, je-li A<>0 JNZ <adr>

Všechny uvedené instrukce (JB,JNB,JBC,JC,JNC,JZ,JNZ) jsou tzv. podmíněné skoky, kdy skok na zadanou adresu je podmíněn splněním určité podmínky - (bit nastaven/vynulován, akumulátor je nulový...). Jejich další společnou vlastností je, že cíl skoku může ležet pouze v rozsahu <-128,+127> bajtů.

Další skupinou instrukcí, které umožňují větvení programu, jsou tzv.nepodmíněné instrukce:

nepodmíněný skok v rozsahu <-128,+127> SJMP <adr>
nepodmíněný skok v rozsahu 2 kB AJMP <adr>
nepodmíněný skok v celém adr. rozsahu LJMP <adr>
nepodmíněný skok v celém adr. rozsahu JMP @A+DPTR

Význam a použití prvních třech instrukcí pro nepodmíněný skok je zřejmý - liší se pouze v rozsahu, ve kterém může ležet adresa skoku. Povšimněme si blíže instrukce JMP @A+DPTR. Tato instrukce provede skok na cílovou adresu, která se vypočítá jako součet obsahu akumulátoru a DPTR. To je podstatný rozdíl oproti předchozím typům instrukcí, kdy cílovou adresu skoku jsme museli znát již při překladu instrukce do strojového kódu, kdežto adresa skoku u instrukce JMP @A+DPTR se vypočítává až **za běhu programu**!

Příklad:

Máme provést rozskok na jednu z osmi adres programu v závislosti na obsahu akumulátoru, který může nabývat hodnot z intervalu 1..8. Např. je-li A=2, skočí se na podprogram DRUHA.

```
MOV DPTR,#TAB ;adresa rozskokové tab.
DEC A ;korekce na počátek tab.
RL A ;násobení A dvěma (prvky
JMP @A+DPTR ;v tabulce jsou 2-bytové)

TAB: AJMP PRVNI
AJMP DRUHA
.
.
AJMP SEDMA
```

Vytváření cyklů

Při programování cyklů se používají v podstatě tři postupy, známe z vyšších programovacích jazyků: **repeat - until**, **while** a **for**.

```
repeat
.
.
.
until <podmínka>
```

```
while <podmínka> do
.
.
.
end
```

```
for <počet cyklů> do
.
.
.
end
```

A. U cyklu repeat-until se podmínka pro ukončení cyklu testuje vždy až na konci cyklu - vždy tedy dojde k tomu, že program projde cyklem alespoň jednou.

Příklad:

```
REPEAT:
.
.
;tělo cyklu
.
.
JC REPEAT ;until <podmínka>
```

Pokud je příznak C nastaven, program přejde na návěští REPEAT a celý cyklus se opakuje znovu.

B. Cyklus typu **while** vyhodnocuje podmínku pro vstup do cyklu na jeho počátku - může se tedy stát, že program cyklem neprojde ani jednou.

```
WHILE: JNC KONEC
.
.
;tělo cyklu
.
.
SJMP WHILE
KONEC:
```

Pokud příznak C není nastaven, provede se skok na návěští KONEC a program se do cyklu nedostane.

C. Cyklus **for** je tzv. tvrdý počítaný cyklus. Na rozdíl od předchozích typů je počet průchodů cyklu předem znám. Pro tento účel je přímo předeslána instrukce:

odečti a skoč, není-li výsledek nula DJNZ <oper>,<adr>

Příklad:

Jedno z možných použití tohoto cyklu je při vytváření časových smyček. Např zpoždění 100 Ţs při krystalu 6MHz:

```
;zpozdění 100 mikrosekund (6 MHz)
DELAY: MOV R0,#24 ;naplň počítadlo cyklů
DJNZ R0,$ ;skáče sám na sebe (24x)
NOP
```

Poznámka: Pro výpočet časových smyček je vždy nutno znát frekvenci krystalu, se kterou procesor pracuje a potom délku každé instrukce v cyklech. Pro náš případ:

```
MOV R0,#24 ; 2 Ţs
DJNZ R0,$ ; 24*4=96 Ţs
NOP ; 2 Ţs
                    ————————
100 Ţs
```

K vytváření cyklů a větvení programu slouží také instrukce:
porovnej a skoč CJNE <oper1>,<oper2>,<adr>
při nerovnosti
Tato instrukce nejprve porovná oba operandy a pokud se nerovnají, provede relativní skok na zadanou adresu, jinak program pokračuje další instrukcí. Tato instrukce je v celém repertoáru instrukcí 8051 jediná, která porovnává dva operandy a nastavuje příznak C beze změny hodnot operandů !!!!
Příklad:
Pro vytvoření cyklu repeat-until:

```
MOV B,#5 ;konečná hodnota
MOV A,#0 ;počáteční hodnota
REPEAT: INC A
CJNE A,B,REPEAT ;porovnej A a B, pokud se
;liší, skoč na REPEAT
```

Pro vytvoření cyklu for:

```
MOV R0,#0
FOR: .
;tělo cyklu for
.
INC R0
CJNE R0,#10,FOR ;dokud je R0 menší než 10,
;opakuj cyklus
```

Podrobný popis instrukcí skoků a volání podprogramů

AJMP kód adr Absolute Jump within 2 KB Page Nepodmíněný skok uvnitř 2 KB slabikové stránky Operandy: kód adr 0 <= kód adr <= 2047 Strojový kód: |aaa00001|aaaaaaaa| 7 0 7 0 Činnost: PC <- (PC) + 2 PC_{0-10} <- adr stránky Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce nejprve inkrementuje obsah programového čítače. Pak nahradí jedenáct nižších bitů programového čítače jedenácti bity kódové adresy. Cílová adresa specifikovaná v instrukci bude tedy v téže 2 K slabikové stránce paměti, ve které je umístěna instrukce následující za AJMP.

LJMP kód adr Long Jump Dlouhý skok Operandy: kód adr 0 <= kód adr <= 65535 Strojový kód: |00000010|kód adr - v.ř.|kód adr - n.ř.| 7 0 7 0 7 0 Činnost: PC <- kód adr Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce předá řízení na šestnáctibitovou kódovou adresu uvedenou v její operandové části.

SJMP kód adr Short Jump Krátký skok Operandy: kód adr Strojový kód: |10000000|rel posun.| 7 0 7 0 Činnost: PC <- (PC) + 2 PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce předává řízení na určenou kódovou adresu. Programový čítač je inkrementován a

ukazuje na následující instrukci. K obsahu inkrementovaného čítače se přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklá adrese.

JMP @A+DPTR Jump to Sum of Accumulator and Data Pointer Skok podle součtu obsahu střadače a ukazatele dat Operandy: A Střadač DPTR Ukazatel dat Strojový kód: |01110011| 7 0 Činnost: PC <- (A) + (DPTR) Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce sečte obsah střadače s obsahem ukazatele dat. Řízení se předá na kódovou adresu vytvořenou z tohoto součtu.

JMP kód adr General Jump Obecný skok Operandy: kód adr 0 <= kód adr <= 65535 Strojový kód: Pseudoinstrukce kompilátoru - překládá se podle potřeby buď jako AJMP nebo LJMP nebo SJMP Činnost: Jako AJMP nebo LJMP nebo SJMP Popis: Instrukce se překládá jako SJMP, neobsahuje-li určená kódová adresa dopředné odkazy a nachází-li se v rozsahu -128 až +127 od adresy následující instrukce. Jako AJMP se instrukce přeloží tehdy, neobsahuje-li kódová adresa dopředné odkazy a nachází-li se v rozsahu současné 2 K slabikové stránky. V ostatních případech se instrukce JMP překládá jako LJMP. Jsou-li v cílové adrese skoku užity dopředné odkazy, nemusí být generovaná skoková instrukce vždy nejefektivnější, jak ukazuje dále uvedený příklad. Podrobnosti jsou uvedeny v popisu instrukcí SJMP, AJMP a LJMP. Příklad: JMP PRESKOC ;Generuje se LJMP ;i když by stačila ;SJMP FF: INC A ;Zvýšení obsahu ;střadače o 1 PRESKOC: INC R5 ;Zvýšení obsahu ;registru 5 o 1

JB bit adr, kód adr Jump if Bit Is Set Skok při nenulovém bitu Operandy: bit adr 0 <= bit adr <= 255 kód adr Strojový kód: |00100000| bit adr | rel posun | 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (bit adr)=1, pak PC <- (PC) + rel posun Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce testuje stav bitu určeného bitovou adresou. Má-li bit logickou hodnotu 1, řízení přechází na specifikovanou kódovou adresu. Obsahuje-li bit logickou hodnotu 0, program pokračuje sekvenčně následující instrukcí. Programový čítač je inkrementován a ukazuje na následující instrukci. Při úspěšném testu bitu se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JBC bit adr, kód adr Jump and Clear if Bit Is Set Skok při nenulovém bitu a jeho vynulování Operandy: bit adr 0 <= bit adr <= 255 kód adr Strojový kód: |00010000| bit adr | rel posun | 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (bit adr) = 1, pak bit adr <- 0 PC <- (PC) + rel posun Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce testuje stav bitu určeného bitovou adresou. Má-li bit logickou hodnotu 1, vynuluje se a řízení přechází na specifikovanou kódovou adresu. Obsahuje-li bit logickou hodnotu 0, program pokračuje sekvenčně následující instrukcí.
Programový čítač je inkrementován a ukazuje na následující instrukci. Při úspěšném testu bitu se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JC kód adr Jump if Carry Is Set Skok při nenulovém příznaku přenosu Operandy: kód adr Strojový kód: |01000000| rel posun | 7 0 7 0 Činnost: PC <- (PC) + 2 Je-li C = 1, pak PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce testuje stav bitu příznaku přenosu. Je-li v logické hodnotě 1, přechází řízení na určenou kódovou adresu; jinak program pokračuje sekvenčně následující instrukcí.
Programový čítač je inkrementován a ukazuje na následující instrukci. Při úspěšném testu bitu příznaku přenosu se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JNB bit adr,kód adr Jump if Bit is Not Set Skok při nenulovém bitu Operandy: bit adr 0 <= bit adr <= 255 kód adr Strojový kód: |00110000| bit adr | rel posun | 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (bit adr) = 0, pak PC <- (PC) + rel posun Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce testuje stav bitu určeného bitovou adresou. Má-li bit logickou hodnotu 0, řízení přechází na specifikovanou kódovou adresu. Obsahuje-li bit logickou hodnotu 1, program pokračuje sekvenčně následující instrukcí.

Programový čítač je inkrementován a ukazuje na následující instrukci. Při úspěšném testu bitu se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JNC kód adr Jump if Carry Is Not Set Skok při nulovém příznaku přenosu Operandy: kód adr Strojový kód: |01010000| rel posun | 7 0 7 0 Činnost: PC <- (PC) + 2 Je-li C = 0, pak PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce testuje stav bitu příznaku přenosu. Je-li v logické hodnotě 0, přechází řízení na určenou kódovou adresu; jinak program pokračuje sekvenčně následující instrukcí.

Programový čítač je inkrementován a ukazuje na následující instrukci. Při úspěšném testu bitu příznaku přenosu se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JNZ kód adr Jump if Accumulator Is Not Zero Skok při nenulovém obsahu střadače Operandy: kód adr Strojový kód: |01110000| rel posun | 7 0 7 0 Činnost: PC <- (PC) + 2 Je-li (A) <> 0, pak PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce testuje obsah střadače. Je-li nenulový, přechází řízení na určenou kódovou adresu; jinak program pokračuje sekvenčně následující instrukcí.

Programový čítač je inkrementován a ukazuje na následující instrukci. Je-li obsah střadače nenulový, přičte se k inkrementovanému programovému čítači relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

JZ kód adr Jump if Accumulator Is Zero Skok při nulovém obsahu střadače Operandy: kód adr Strojový kód: |01100000| rel posun | 7 0 7 0 Činnost: PC <- (PC) + 2 Je-li (A) = 0, pak PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce testuje obsah střadače. Je-li nulový, přechází řízení na určenou kódovou adresu; jinak program pokračuje sekvenčně následující instrukcí.

Programový čítač je inkrementován a ukazuje na následující instrukci. Je-li obsah střadače nulový, přičte se k inkrementovanému programovému čítači relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

CJNE @Rr,#data,kód adr Compare Indirect Address to Immediate Data, Jump if Not Equal Porovnání obsahu nepřímé adresy s přímými daty, skok při nerovnosti

Operandy: Rr Registr 0 <= r <= 1 data -256 <= data <= +255 kód adr Strojový kód: |1011011r| data |rel posun| 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li ((Rr)) <> data, pak PC <- (PC) + rel posun Je-li ((Rr)) < data, pak C <- 1 jinak C <- 0 Slabik: 3 Cyklů: 2 Příznaky: C Popis: Instrukce srovnává hodnotu přímých dat s obsahem buňky vnitřní paměti dat adresované registrem r. Při nerovnosti obou údajů přechází řízení na určenou kódovou adresu. Při rovnosti obou údajů program pokračuje sekvenčně následující instrukcí.

Mají-li přímá data hodnotu větší než je obsah adresované paměťové buňky, nastaví se příznak přenosu do logické hodnoty 1; v ostatních případech se vynuluje.

Obsah programového čítače se inkrementuje a ukazuje na následující instrukci. Při nerovnosti operandů se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

CJNE A,#data,kód adr Compare Immediate Data to Accumulator, Jump if Not Equal Porovnání přímých dat s obsahem střadače, skok při nerovnosti

Operandy: A Střadač data -256 <= data <= +255 kód adr Strojový kód: |10110100| data |rel posun| 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (A) <> data, pak PC <- (PC) + rel posun Je-li (A) < data, pak C <- 1 jinak C <- 0 Slabik: 3 Cyklů: 2 Příznaky: C Popis: Instrukce srovnává hodnotu přímých dat s obsahem střadače. Při nerovnosti obou údajů přechází řízení na určenou kódovou adresu. Při rovnosti obou údajů program pokračuje sekvenčně následující instrukcí.

Mají-li přímá data hodnotu větší než je obsah střadače, nastaví se příznak přenosu do logické hodnoty 1; v ostatních případech se vynuluje.

Obsah programového čítače se inkrementuje a ukazuje na následující instrukci. Při nerovnosti operandů se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

CJNE A,data adr,kód adr Compare Memory to Accumulator, Jump if Not Equal
Porovnání obsahu datové adresy se střada
čem, skok při nerovnosti
Operandy: A Střadač data adr 0 <= data adr <= 255 kód adr Strojový kód: |10110101|data adr |rel posun| 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (A) <> (data adr), pak PC <- (PC) + rel posun Je-li (A) < (data adr), pak C <- 1 jinak C <- 0 Slabik: 3 Cyklů: 2 Příznaky: C Popis: Instrukce srovnává obsah určené datové adresy s obsahem střadače. Při nerovnosti obou údajů přechází řízení na určenou kódovou adresu. Při rovnosti obou údajů program pokračuje sekvenčně následující instrukcí.
Je-li obsah datové adresy větší než je obsah střadače, nastaví se příznak přenosu do logické hodnoty 1; v ostatních případech se vynuluje.
Obsah programového čítače se inkrementuje a ukazuje na následující instrukci. Při nerovnosti operandů se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

CJNE Rr,#data,kód adr Compare Immediate Data to Register, Jump if Not Equal
Porovnání přímých dat s obsahem registru, skok při nerovnosti
Operandy: Rr Registr 0 <= r <= 7 data -256 <= data <= +255 kód adr Strojový kód: |10111rrr| data |rel posun| 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 Je-li (Rr) <> data, pak PC <- (PC) + rel posun Je-li (Rr) < data, pak C <- 1 jinak C <- 0 Slabik: 3 Cyklů: 2 Příznaky: C Popis: Instrukce srovnává hodnotu přímých dat s obsahem registru r. Při nerovnosti obou údajů přechází řízení na určenou kódovou adresu. Při rovnosti obou údajů program pokračuje sekvenčně následující instrukcí.
Je-li hodnota přímých dat větší než je obsah registru, nastaví se příznak přenosu do logické hodnoty 1; v ostatních případech se vynuluje.
Obsah programového čítače se inkrementuje a ukazuje na následující instrukci. Při nerovnosti operandů se k inkrementovanému programovému čítači přičte relativní posun a bude se provádět instrukce umístěná na takto vzniklé adrese.

DJNZ Rr,kód adr Decrement Register and Jump if Not Zero
Snížení obsahu registru a skok při nenulovém výsledku Operandy: Rr Registr 0 <= r <= 7 kód adr Strojový kód: |11011rrr|rel posun| 7 0 7 0 Činnost: PC <- (PC) + 2 Rr <- (Rr) - 1 Je-li (Rr) <> 0, pak PC <- (PC) + rel posun Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce odečte od obsahu registru r jedničku a výsledek umístí do téhož registru. Je-li výsledek odečtení nulový, pokračuje program sekvenčně následující instrukcí; jinak se řízení předává na určenou kódovou adresu.
Obsah programového čítače je inkrementován a ukazuje na následující instrukci. Je-li výsledek odečtení jedničky nenulový, přičte se relativní posun k inkrementovanému programovému čítači a bude se provádět instrukce umístěná na takto vzniklé adrese.

DJNZ data adr, kód adr Decrement Memory and Jump if Not Zero
Snížení obsahu datové adresy a skok při nenulovém výsledku
Operandy: data adr 0 <= data adr <= 255 kód adr Strojový kód: |11010101|data adr |rel posun| 7 0 7 0 7 0 Činnost: PC <- (PC) + 3 data adr <- (data adr) - 1 Je-li (data adr) <> 0, pak PC <- (PC) + rel posun Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce odečte od obsahu datové adresy jedničku a výsledek umístí na tutéž adrsu. Je-li výsledek odečtení nulový, pokračuje program sekvenčně následující instrukcí; jinak se řízení předává na určenou kódovou adresu.
Obsah programového čítače je inkrementován a ukazuje na následující instrukci. Je-li výsledek odečtení jedničky nenulový, přičte se relativní posun k inkrementovanému programovému čítači a bude se provádět instrukce umístěná na takto vzniklé adrese.

ACALL kód adr Absolute Call Within 2 KB Page

Volání podprogramu uvnitř 2 K slabikové paměti Operandy: kód adr Strojový kód: |aaa10001|aaaaaaaa| 7 0 7 0 Činnost: PC <- (PC) + 2 SP <- (SP) + 1 (SP) <- (PC n.ř.) SP <- (SP) + 1 (SP) <- (PC v.ř.) PC_{0-10} <- adr ve stránce Slabik: 2 Cyklů: 2 Příznaky: - Popis: Instrukce uloží inkrementovaný obsah programového čítače (tj. návratovou adresu) do zásobníku. Slabika nižších řádů programového čítače se do zásobníku ukládá jako první. Cílová adresa počátku podprogramu se vytvoří z pěti nejvyšších bitů inkrementovaného programového čítače, z bitů 7 až 5 operačního znaku a z druhé slabiky strojového kódu instrukce ACALL. Volaný podprogram tedy musí začínat uvnitř téže 2 K slabikové stránky paměti, ve které je umístěna instrukce následující za ACALL

Jedenáztibitová adresa uvnitř 2 K slabikové stránky se skládá ze ří nejvyšších bitů operačního znaku a z osmi bitů druhé slabiky strojového kódu instrukce ACALL.

LCALL kód adr Long Call

Dlouhé volání podprogramu Operandy: kód adr Strojový kód: |00010010|kód adr - v.ř|kód adr - n.ř.| 7 0 7 0 7 0 Činnost: PC <- (PC) + 2 SP <- (SP) + 1 (SP) <- (PC n.ř.) SP <- (SP) + 1 (SP) <- (PC v.ř.) PC <- kód adr Slabik: 3 Cyklů: 2 Příznaky: - Popis: Instrukce uloží inkrementovaný obsah programového čítače (tj. návratovou adresu) do zásobníku a předá řízení na šestnáctibitovou kódovou adresu uvedenou v její operandové části.

CALL kód adr General Call

Obecné volání podprogramu Operandy: kód adr Strojový kód: překládá se podle potřeby jako ACALL nebo LCALL Činnost: Jako ACALL nebo LCALL Popis: Pseudoinstrukce kompilátoru - překládá se jako ACALL, neobsahuje-li specifikovaná kódová adresa dopředné odkazy a nachází-li se tato adresa uvnitř současné 2 K slabikové stránky; jinak se překládá jako LCALL. Generování instrukce LCALL v případě dopředných odkazů nemusí být vždy nejefektivnější, jak ukazuje dále uvedený příkad. Podrobnosti jsou uvedeny v popisu instrukcí ACALL a LCALL.

Příklad:

```
ORG 80DCH
CALL PPG3 ;Volání PPG3 (PPG3
;je dopředný odkaz
;takže se generuje
;LCALL, i když by
;v tomto případě
;stačila ACALL)
....
....
PPG3: POP 55H ;předpokládáme, že
;instrukce POP je
;na adrese 8233H
```

RET Return from Subroutine

(Non - interrupt) Návrat z podprogramu (ne z obsluhy přerušení) Operandy: žádné Strojový kód: |00100010| 7 0 Činnost: PC v.ř. <- ((SP)) SP <- (SP) - 1 PC n.ř. <- ((SP)) SP <- (SP) - 1 Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce provede návrat z podprogramu. Řízení přechází na adresu, která je uložena ve dvou slabikách v zásobníku. Vyšší řády návratové adresy se vyberou ze zásobníkové paměti jako první, nižší řády návratové adresy jako druhé. Obsah ukazatele zásobníku se sníží o 2.

RETI Return from Interrupt Routine

Návrat z podprogramu pro obsluhu přerušení Operandy: žádné Strojový kód: |00110010| 7 0 Činnost: PC v.ř. <- ((SP)) SP <- (SP) - 1 PC n.ř. <- ((SP)) SP <- (SP) - 1 Slabik: 1 Cyklů: 2 Příznaky: - Popis: Instrukce provede návrat z podprogramu pro ošetření přerušení a znovu povolí přerušení stejné nebo nižší priority. Řízení přechází na adresu, která je uložena ve dvou slabikách v zásobníku. Vyšší řády návratové adresy se vyberou ze zásobníkové paměti jako první, nižší řády návratové adresy jako druhé. Stavové slovo programu (PSW) se automaticky neobnovuje. Obsah ukazatele zásobníku se sníží o 2.