Instrukce mikropočítače Intel 8051

Použité symboly:

#d8	-	osmibitová konstanta (přímá osmibitová data)
#d16	-	šestnáctibitová konstanta (přímá šestnáctibitová data)
A	-	střadač (akumulátor)
B	-	registr B
Ri	-	registr R0 až R7
C	-	příznak přenosu CY
@Ri	-	paměťové místo adresované obsahem registru R0 nebo R1 (*nepřímá adresa*)
@DPTR	-	paměťové místo adresované registrem DPTR
dir	-	osmibitová *přímá adresa* vztažená k vnitřní RWM včetně SFR
bit	-	osmibitová adresa bitu ve vntřní RWM včetně SFR
adr16	-	šestnáctibitová přímá adresa
adr11	-	osm bitů jedenáctibitové adresy (nejvyšší tři bity jsou součástí operačního znaku instrukce) Tato adresa se používá v instrukcích skoků pro rychlý pohyb v rozmezí 2 kbytů paměti (2^{11}= 2048 = 2k).
rel	-	osmibitová relativní adresa (v rozmezí -128 až +127, nejvyšší bit určuje znaménko)

Rozdělení instrukcí:

1. Instrukce přesunů
2. Aritmetické instrukce
3. Instrukce pro logické operace a pro rotace
4. Instrukce bitových operací
5. Instrukce větvení programu

V následujícím přehledu instrukcí je v prvním sloupci uveden zápis instrukce, v druhém operace, kterou instrukce vykoná, ve třetím ovlivněné příznaky, ve čtvrtém počet bytů a v pátém počet strojových cyklů instrukce.

1. Instrukce přesunů

Přesuny ve vnitřní paměti dat (včetně SFR)

MOV A,Ri	(Ri) → A ; i = 0 až 7	P	1B	1C
MOV A,dir	(dir) → A	P	2B	1C
MOV A,@Ri	((Ri)) → A ; i = 0,1	P	1B	1C
MOV A,#d8	d8 → A	P	2B	1C

MOV Ri,A	(A) → Ri ; i = 0 až 7	-	1B	1C
MOV Ri,dir	(dir) → Ri ; i = 0 až 7	-	2B	1C
MOV Ri,#d8	d8 → Ri ; i = 0 až 7	-	2B	1C
MOV dir,A	(A) → dir	-	2B	1C
MOV dir,Ri	(Ri) → dir	-	2B	1C
MOV dirx,diry	(diry) → dirx	-	3B	2C
MOV dir,@Ri	((Ri)) → dir ; i = 0,1	-	2B	1C
MOV dir,#d8	d8 → dir	-	3B	2C
MOV @Ri,A	(A) → (Ri) ; i = 0,1	-	1B	1C
MOV @Ri,dir	(dir) → (Ri) ; i = 0,1	-	2B	1C
MOV @Ri,#d8	d8 → (Ri) ; i = 0,1	-	2B	1C
MOV DPTR,#d16	d16 → DPTR	-	3B	2C
Přístup k paměti programu				
MOVC A,@A+DPTR	((A)+(DPTR)) → A	P	1B	2C
MOVC A,@A+PC	((A)+(PC)) → A	P	1B	2C
Přístup k vnější paměti dat				
MOVX A,@Ri	((Ri)) → A ; i = 0,1	P	1B	2C
MOVX A,@DPTR	((DPTR)) → A	P	1B	2C
MOVX @Ri,A	A → (Ri) ; i = 0,1	-	1B	2C
MOVX @DPTR,A	A → (DPTR)	-	1B	2C
Práce se zásobníkem				
PUSH dir	(SP)+1 → SP (dir) → (SP)	-	2B	2C
POP dir	((SP)) → dir (SP)-1 → SP	-	2B	2C
Vzájemné výměny ve vnitřní paměti dat (včetně SFR)				
XCH A,Ri	(A) ↔ (Ri) ; i = 0 až 7	P	1B	1C
XCH A,dir	(A) ↔ (dir)	P	2B	1C
XCH A,@Ri	(A) ↔ ((Ri)) ; i = 0,1	P	1B	1C
XCHD A,@Ri	(AL) ↔ 3.až 0.bit ((Ri))	P	1B	1C

Poznámky:

- Instrukce mající jako cílový registr akumulátor ovlivňují příznak parity P, ostatní přesunové instrukce neovlivňují žádné příznaky.
- Instrukce *MOV A,dir* přesune obsah přímé adresy do akumulátoru.
- Instrukce *MOV A,@Ri* přesune obsah paměťového místa adresovaného obsahem registru Ri do akumulátoru. ***Pozor - pro nepřímé adresování může být použit pouze registr R0 nebo R1.***
- Instrukce *MOV dir,A* přesune obsah akumulátoru na adresu *dir* (přímá adresa).
- Instrukce *MOV dirx,diry* přesune obsah paměťového místa o adrese diry na adresu dirx.
- Instrukce *MOV @Ri,A* přesune obsah akumulátoru na paměťové místo adresované obsahem registru Ri (R0 nebo R1).
- Instrukce ***MOVC*** přesouvají byte z ***paměti programu*** do akumulátoru. Adresa bytu je dána součtem (A)+(DPTR) nebo (A)+(PC).
  Adresování registry DPTR (příp. PC) a A představuje tzv. *indexbázové adresování.* Obsah registru DPTR (příp. PC) určuje bázi, tj. adresu počátku nějaké tabulky dat, obsah registru A pak představuje index v této tabulce.
- Instrukce ***MOVX*** umožňují přístup k ***vnější paměti dat***. Pracují s nepřímou adresou buď v registru R0, příp. R1 (tak obsáhnou 256 B paměti) nebo v registru DPTR (pak je dosažitelná celá paměť 64 kB). Data se přenášejí pouze prostřednictvím akumulátoru.
- Instrukce *PUSH dir* uloží do zásobníku obsah přímé adresy *dir*.
- Instrukce *POP dir* přesune byte z vrcholu zásobníku na paměťové místo o adrese *dir*.
- Instrukce *XCHD A,@Ri* vymění navzájem dolní polovinu obsahu akumulátoru s dolní polovinou bytu adresovaného obsahem registru Ri.

Příklady:

Přímá adresa *dir* zahrnuje i adresy SFR registrů. V zápisech instrukcí můžeme za *dir* dosazovat symbolická označení SFR registrů (např. B, P0, P1, TMOD, SBUF). Překladač tato jména zná a při překladu za ně dosadí správné adresy registrů.

MOV A,#3FH	naplnění akumulátoru číslem 3FH
MOV A,3FH	naplnění akumulátoru *obsahem adresy* 3FH (3FH zde představuje přímou adresu *dir*)
MOV A,B	přesun obsahu registru B do akumulátoru (symbol B je dosazen za *dir*, překladač mu přiřadí správnou adresu)
MOV A,P0	čtení z portu P0 (P0 dosazeno za *dir*) - přečte se byte na vývodech portu, nikoliv obsah vyrovnávacího registru
MOV R2,64H	naplnění registru R2 *obsahem adresy* 64H

MOV TMOD,#10H	naplnění registru TMOD číslem 10H
MOV 2AH,@R0	přesun obsahu paměťového místa adresovaného registrem R0 na paměťové místo o adrese 2AH
MOV TL0,25H	naplnění registru TL0 (dirx) obsahem paměťového místa o adrese 25H (diry)
PUSH ACC	uložení obsahu akumulátoru do zásobníku
PUSH B	uložení obsahu registru B do zásobníku
PUSH PSW	uložení obsahu registru PSW (stavového slova) do zásobníku

2. Aritmetické instrukce

ADD A,Ri	(A)+(Ri) → A ; i = 0 až 7	CY,AC,OV,P	1B	1C
ADD A,dir	(A)+(dir) → A	CY,AC,OV,P	2B	1C
ADD A,@Ri	(A)+((Ri)) → A ; i = 0,1	CY,AC,OV,P	1B	1C
ADD A,#d8	(A)+d8 → A	CY,AC,OV,P	2B	1C
ADDC A,Ri	(A)+CY+(Ri) → A ; i = 0,1	CY,AC,OV,P	1B	1C
ADDC A,dir	(A)+CY+(dir) → A	CY,AC,OV,P	2B	1C
ADDC A,@Ri	(A)+CY+((Ri)) → A ; i = 0,1	CY,AC,OV,P	1B	1C
ADDC A,#d8	(A)+CY+d8 → A	CY,AC,OV,P	2B	1C
SUBB A,Ri	(A)-CY-(Ri) → A ; i = 0,1	CY,AC,OV,P	1B	1C
SUBB A,dir	(A)-CY-(dir) → A	CY,AC,OV,P	2B	1C
SUBB A,@Ri	(A)-CY-((Ri)) → A ; i = 0,1	CY,AC,OV,P	1B	1C
SUBB A,#d8	(A)-CY-d8 → A	CY,AC,OV,P	2B	1C
INC A	(A)+1 → A	P	1B	1C
INC Ri	(Ri)+1 → Ri ; i = 0 až 7	-	1B	1C
INC dir	(dir)+1 → dir	-	2B	1C
INC @Ri	((Ri))+1 → (Ri) ; i = 0,1	-	1B	1C
INC DPTR	(DPTR)+1 → DPTR	-	1B	2C
DEC A	(A)-1 → A	P	1B	1C
DEC Ri	(Ri)-1 → Ri ; i = 0 až 7	-	1B	1C
DEC dir	(dir)-1 → dir	-	2B	1C
DEC @Ri	((Ri))-1 → (Ri) ; i = 0,1	-	1B	1C

MUL AB	(A)∗(B) → B,A nižší byte výsledku v A, vyšší v B	CY,OV,P	1B	4C
DIV AB	(A)/(B) → A,B celočíselná část výsledku v A, zbytek v B	CY,OV,P	1B	4C
DA A	úprava obsahu akumulátoru na desítkový tvar (po instrukcích ADD, ADDC)	CY,AC,P	1B	1C

Poznámky:

- *Nastavení příznaků instrukcí SUBB:*
  CY=1 byla-li při operaci nutná výpůjčka pro 7. bit
  AC=1 byla-li při operaci nutná výpůjčka pro 3. bit
  OV=1 při neekvivalenci výpůjček pro 6. a 7. bit
- Je-li v instrukci *INC dir* nebo *DEC dir* adresován port (např. *INC P0*), dochází k inkrementaci či dekrementaci vyrovnávacího registru portu.
- *Ovlivnění příznaků CY a OV instrukcí MUL AB:*
  CY=0 CY je po operaci vždy vynulován
  OV=1 je-li výsledek větší než FFH
- *Ovlivnění příznaků CY a OV instrukcí DIV AB:*
  CY=0 CY je po operaci vždy vynulován
  OV=1 při dělení nulou, jinak OV=0

Příklady:

ADD A,R5	(A)+(R5) → A
ADD A,B	(A)+(B) → A
ADD A,P1	(A)+ byte na vstupu P1 (nikoliv obsah vyrovnávacího registru portu) → A
ADDC A,50H	(A)+CY+ obsah paměťového místa o adrese 50H → A
SUBB A,#10H	(A)-CY-10H → A
ADD A,@R1	(A)+ obsah paměťového místa adresovaného obsahem registru R1 → A

3. Instrukce logických operací, rotace

ANL A,Ri	(A) and (Ri) → A ; i = 0 až 7	P	1B	1C
ANL A,dir	(A) and (dir) → A	P	2B	1C
ANL A,@Ri	(A) and ((Ri)) → A ; i = 0,1	P	1B	1C
ANL A,#d8	(A) and d8 → A	P	2B	1C

ANL dir,A	(dir) and (A) → dir	-	2B	1C
ANL dir,#d8	(dir) and d8 → dir	-	3B	2C
ORL A,Ri	(A) or (Ri) → A ; i = 0 až 7	P	1B	1C
ORL A,dir	(A) or (dir) → A	P	2B	1C
ORL A,@Ri	(A) or ((Ri)) → A ; i = 0,1	P	1B	1C
ORL A,#d8	(A) or d8 → A	P	2B	1C
ORL dir,A	(dir) or (A) → dir	-	2B	1C
ORL dir,#d8	(dir) or d8 → dir	-	3B	2C
XRL A,Ri	(A) xor (Ri) → A ; i = 0 až 7	P	1B	1C
XRL A,dir	(A) xor (dir) → A	P	2B	1C
XRL A,@Ri	(A) xor ((Ri)) → A ; i = 0,1	P	1B	1C
XRL A,#d8	(A) xor d8 → A	P	2B	1C
XRL dir,A	(dir) xor (A) → dir	-	2B	1C
XRL dir,#d8	(dir) xor d8 → dir	-	3B	2C
CLR A	nulování obsahu akumulátoru	P	1B	1C
CPL A	negace obsahu akumulátoru	P	1B	1C
RL A	rotace (A) o jeden bit vlevo; 7.bit → 0.bit	P	1B	1C
RLC A	rotace (A) o jeden bit vlevo přes CY; 7.bit → CY, CY → 0.bit	P,CY	1B	1C
RR A	rotace (A) o jeden bit vpravo; 0.bit → 7.bit	P	1B	1C
RRC A	rotace (A) o jeden bit vpravo přes CY; 0.bit → CY, CY → 7.bit	P,CY	1B	1C
SWAP A	(AH) ↔ (AL) vzájemná výměna nižší a vyšší čtveřice bitů obsahu akumulátoru	-	1B	1C

Příklady:

ANL B,#11110000B	nulování dolní čtveřice bitů obsahu registru B (*maskování*), operand 11110000B=F0H představuje masku
ANL A,#10101010B	nulování bitů 0,2,4,6 obsahu akumulátoru
ORL A,#1	nastavení bitu 0 obsahu akumulátoru do jedničky
ORL A,#11100000B	nastavení bitu 5,6,7 obsahu akumulátoru do jedničky
XRL P0,#0F0H	invertování horních čtyř bitů obsahu vyrovnávacího registru brány P0

4. Instrukce bitových operací

CLR C	nulování příznaku CY	CY	1B	1C
CLR bit	nulování přímo adresovatelného bitu	-	2B	1C
SETB C	nastavení příznaku CY do jedničky	CY	1B	1C
SETB bit	nastavení přímo adresovatelného bitu do jedničky	-	2B	1C
CPL C	negace příznaku CY	CY	1B	1C
CPL bit	negace přímo adresovatelního bitu	-	2B	1C
ANL C,bit	CY and (bit) → CY	CY	2B	2C
ANL C,/bit	CY and $\overline{(bit)}$ → CY	CY	2B	2C
ORL C,bit	CY or (bit) → CY	CY	2B	2C
ORL C,/bit	CY or $\overline{(bit)}$ CY	CY	2B	2C
MOV C,bit	(bit) → CY	CY	2B	2C
MOV bit,C	CY → bit	-	2B	2C

Příklady:

SETB ACC.0	nastavení nultého bitu akumulátoru do jedničky
SETB P0.7	zapsání jedničky do 7.bitu brány P0
CLR P1.3	zapsání nuly do 3.bitu brány P1
ANL C,5FH	logický součin příznaku CY a bitu na adrese 5FH (v bitově adresovatelném prostoru vnitřní RWM)
MOV C,P1.0	načtení 0.bitu brány P1 do CY
MOV T1,C	přenos CY na vývod T1 (tj. P3.5)
SETB RS0 CLR RS1	aktualizace 1.banky registrů R0 až R7 (RS0=1, RS1=0)

5. Instrukce větvení programu

Nepodmíněné skoky

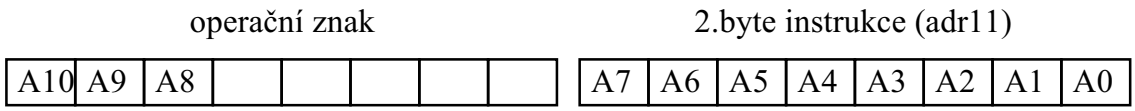
ACALL adr11	návratová adresa → zásobník (SP)+2 → SP adresa 11bitů → PC	-	2B	2C
LCALL adr16	návratová adresa → zásobník (SP)+2 → SP adr16 → PC	-	3B	2C
RET	(zásobník) → PC (SP)-2 → SP	-	1B	2C
RETI	(zásobník) → PC (SP)-2 → SP	-	1B	2C
AJMP adr11	adresa 11 bitů → PC	-	2B	2C
LJMP adr16	adr16 → PC	-	3B	2C
SJMP rel	(PC)+rel → PC	-	2B	2C
JMP @A+DPTR	(A)+(DPTR) → PC	-	1B	2C

Podmíněné skoky

JZ rel	je-li (A)=0, pak (PC)+rel → PC	-	2B	2C
JNZ rel	je-li (A)≠0, pak (PC)+rel → PC	-	2B	2C
JC rel	je-li CY=1, pak (PC)+rel → PC	-	2B	2C
JNC rel	je-li CY=0, pak (PC)+rel → PC	-	2B	2C
JB bit,rel	je-li (bit)=1, pak (PC)+rel → PC	-	3B	2C
JNB bit,rel	je-li (bit)=0, pak (PC)+rel → PC	-	3B	2C
JBC bit,rel	je-li (bit)=1, pak (PC)+rel → PC 0 → bit	-	3B	2C
CJNE A,dir,rel	je-li (A)≠(dir), pak (PC)+rel → PC	CY	3B	2C
CJNE A,#d8,rel	je-li (A)≠d8, pak (PC)+rel → PC	CY	3B	2C
CJNE Ri,#d8,rel	je-li (Ri)≠d8, pak (PC)+rel → PC	CY	3B	2C
CJNE @Ri,#d8,rel	je-li ((Ri))≠d8, pak (PC)+rel → PC	CY	3B	2C
DJNZ Ri,rel	(Ri)-1 → Ri , je-li Ri≠0 , pak (PC)+rel → PC	-	2B	2C
DJNZ dir,rel	(dir)-1 → dir , je-li (dir)≠0 , pak (PC)+rel → PC	-	3B	2C
NOP	prázdná instrukce	-	1B	1C

Poznámky:

- Instrukce *ACALL adr11* představuje volání podprogramu v rámci stránky paměti o velikosti 2 kB. 11bitová adresa je uložena v 7., 6., 5. bitu OZ a v druhém bytu instrukce:



  Instrukce vloží tuto adresu do dolních jedenácti bitů čítače instrukcí PC. Horních pět bitů zůstane nezměněno (určují stránku paměti o velikosti 2 kB).
- Instrukce *LCALL adr16* je instrukcí volání podprogramu v rámci celé paměti 64 kB (adresa je 16bitová).
- Instrukce *RETI* se používá k návratu z podprogramu *obsluhy přerušení*. Oproti instrukci *RET* navíc povolí přerušení stejné nebo nižší priority.
- *AJMP adr11* je skok v rámci stránky paměti o velikosti 2 kB.
- *LJMP adr16* je skok v rámci celé paměti 64 kB.
- Instrukce *SJMP a instrukce podmíněných skoků* používají osmibitovou relativní adresu *rel*. Cílová adresa skoku vznikne přičtením této relativní adresy (tj. čísla v rozsahu -128 až +127) k obsahu registru PC (ten v době provádění instrukce skoku ukazuje na následující instrukci).
- Instrukce *CJNE cíl,zdroj,rel* je instrukcí porovnání a skoku. Porovná dva operandy, při jejich nerovnosti uskuteční skok na relativní adresu, rovnají-li se, program pokračuje následující instrukcí (compare and jump if not equal).
  Při porovnání se ovlivní příznak CY:
  CY = 1, je-li cílový operand menší než zdrojový, jinak CY = 0.
- Instrukce *DJNZ operand,rel* dekrementuje operand (obsah Ri nebo *dir*), pokud po dekrementaci není operand roven nule, nastane skok, jinak program pokračuje další instrukcí.

Příklady:

JC CYKL	je-li CY=1, skok na návěští CYKL (toto návěští nesmí být od adresy, na které je umístěn OZ instrukce následující za *JC CYKL,* vzdáleno o více než o -128 až +127)
AJMP ZAC	nepodmíněný skok na návěští *ZAC* (toto návěští musí být umístěno v rámci stránky 2 kB)
JNB F0,KONEC	skok na návěští *KONEC*, je-li bit F0=0 (příznak F0 v PSW)
JB P2.5,$	skok na tutéž instrukci, je-li na 5.bitu brány P2 log. jednička (totéž co *ZDE: JB P2.5,ZDE*) - program čeká, až se na P2.5 objeví log. nula znak $ představuje *stav čítače adres*

CJNE A,B,OPAK	porovnání obsahu akumulátoru a registru B, pokud se obsahy liší, nastane skok na návěští *OPAK*
CJNE R0,#10,CEK	pokud obsah R0 není roven deseti, nastane skok na návěští *CEK*
DJNZ R1,$	dekrementace obsahu R1 a skok na tutéž instrukci, pokud po dekrementaci není obsah R1 nulový (program setrvá ve smyčce, dokud se obsah R1 nevynuluje)
DJNZ B,ZPET	dekrementace obsahu B a skok na návěští *ZPET*, není-li po dekrementaci obsah B nulový

Otázky a úkoly:

1. Které registry je možné použít k nepřímému adresování?
2. Vysvětlete činnost instrukcí:
   a) MOV 30H,@R0 b) MOV 30H,50H c) MOV @R1,35H
3. Zdůvodněte, proč je následující zápis instrukce chybný: MOV @R2,#25H.
4. Vysvětlete činnost instrukcí:
   a) XCH A,3AH b) XCH A,@R0 c) XCHD A,@R1
5. Zdůvodněte, proč není správný následující zápis instrukce: XCH A,#35H.
6. V čem se liší činnost instrukcí XCH A,R1 a XCH A,@R1?
7. Jaké využití má registr DPTR?
8. Je-li registr DPTR využit k adresování *vnější paměti dat*, na které brány se vysílají obsahy registrů DPL a DPH?
9. Na kterou bránu se vysílá adresa, je-li pro adresování *vnější paměti dat* použit registr R0 nebo R1? Jak bude při vysílání adresy ovlivněna brána P2?
10. Kterými instrukcemi je dosažitelná *paměť programu*?
11. Jaké existují možnosti pro adresování bytu v *paměti programu*?
12. Co rozumíme pod pojmem *indexbázové adresování*?
13. Jakými způsoby je možné adresovat byte ve vnější paměti dat?
14. Vysvětlete rozdíl mezi činností instrukcí MOV A,3EH a MOV A,#3EH.
15. Co představuje označení *dir*, co můžeme za tento symbol dosadit?
16. Napište příklady instrukcí, kterými přečteme byte z brány či byte na bránu zapíšeme.
17. Vysvětlete, jakou činnost vykonají instrukce:
   a) ANL P0,#0FH b) ANL A,P0 c) XRL P1,#80H
18. Vysvětlete rozdíl mezi činnostmi instrukcí RL A a RLC A.
19. Najděte různé možnosti nastavení bitu P0.7 do logické jedničky.
20. Jakou činnost vykoná instrukce CLR 0?
21. Jaký je rozdíl v činnosti instrukcí SETB C a SETB PSW.7?
22. Jakou činnost vykoná instrukce MOV C,29H?
23. Vysvětlete, co způsobí instrukce:
   a) JNB P1.0,$ b) JBC P1.7,$
24. Vysvětlete činnost instrukce AJMP adr11.
25. Jakým způsobem je uložena adresa ve strojovém kódu instrukce ACALL adr11?
26. Co rozumíme pod pojmem relativní adresa?
27. Jakou funkci má instrukce SJMP rel?

28. Jakým způsobem otestujete, zda obsah akumulátoru je roven nule?
29. Otestujte platnost následujících relací:
    a) (A) < (B) b) (A) ≤ (B) c) (A) ≥ (B) d) (A) > (B)
30. Vysvětlete činnost instrukce CJNE. Jaké příznaky a jakým způsobem ovlivňuje?
31. Vysvětlete činnost instrukce DJNZ.
32. Vytvořte programovou smyčku, která do programu zařadí zpoždění 0,5 ms (uvažujte kmitočet oscilátoru 12 MHz).
33. Vytvořte programovou smyčku, která do programu zařadí zpoždění přibližně 50 ms (při kmitočtu oscilátoru 12 MHz).