

# Programátorský model počítača

Portál: edu.ukf.sk - Vzdelávací portál - Univerzita  
Konštantína Filozofa, Nitra

Kurz: Operačné systémy (KI/OS/15)

Kniha: Programátorský model počítača

Vytlačil(a): Zuzana Pavlendová

Dátum: Streda, 1 december 2021, 17:57

# Opis

TEKST

# Obsah

**Úvod: Programátorský model počítača**

**3.1 Základné inštrukcie**

**3.2 Aritmetické inštrukcie**

**3.3 Podmienené skoky**

**3.4 Zásobník**

**3.5 Programovanie V / V operácií**

**3.6 Multiprogramové spracovanie**

**3.7 Signál RESET**

**3.8 Virtuálna pamäť**

**3.9 Adresovacie techniky**

**3.10 Inštrukcia MOV**

**3.11 Aritmetické inštrukcie (celočíselné)**

**3.12 Logické inštrukcie**

**3.13 Rotácia**

**3.14 Posuvy**

**3.15 Vetvenie programu**

**3.16 Podmienené skoky**

# Úvod: Programátorský model počítača

## V tejto kapitole se dozviete:

- Ako popisuje programátor štruktúru počítača?
- Ktoré základné registre sú nevyhnutné pre počítač?
- Ktoré základné strojové inštrukcie sú k dispozícii programátorovi?
- Ako sú vykonávané (interpretované) jednotlivé strojové inštrukcie radičom počítača?

## Po jej preštudovaní byste mali byť schopní:

- Charakterizovať programátorský model počítača, včítane úlohy jednotlivých registrov.
- Poznať základné strojové inštrukcie počítača.
- Porozumieť spôsobu vykonávania jednotlivých inštrukcií radičom.
- Popísať štruktúru strojovo orientovaného programu.

## Kľúčové slová tejo kapitoly:

Strojový kód, inštrukcie, programátorský model počítača, registre počítača, prerušenie programu.

## Doba potrebná ku štúdiu: 6 hodín

### Sprievodca štúdiom

*Táto kapitola je veľmi náročnou témou študijného textu a sú vňom popísané princípy vytvárania strojovo orientovaných programov. Pomerne náročná téma je hlavne pre tých z Vás, ktorí doteraz nemajú žiadne znalosti z oblasti vytvárania strojovo orientovaných programov. Je však potrebné zvládnuť túto kapitolu, pretože Vám umožní ľahšie pochopiť vlastnú interpretáciu jednotlivých inštrukcií v počítači a pohľad systémového programátora na dôležité systémové programy, ako sú operačné systémy.*

*Na štúdium tejto časti si vyhradte aspoň 6 hodín. Odporúčame študovať s prestávkami vždy po pochopení jednotlivých podkapitol.*

Činnosť počítača je riadená programom, uloženým v operačnej pamäti. Program je tvorený postupnosťou inštrukcií, ktoré tvoria základné programovateľné kroky procesora. V nasledujúcej časti si vysvetlíme, akým spôsobom pozerá programátor na vytváranie programového kódu v strojovo orientovanom jazyku. Vychádzajme z obecného formátu inštrukcie:

Operačný kód (operačný znak)	Adresa operandu / operand	Adresa 2. operandu / 2. operand
------------------------------	---------------------------	---------------------------------

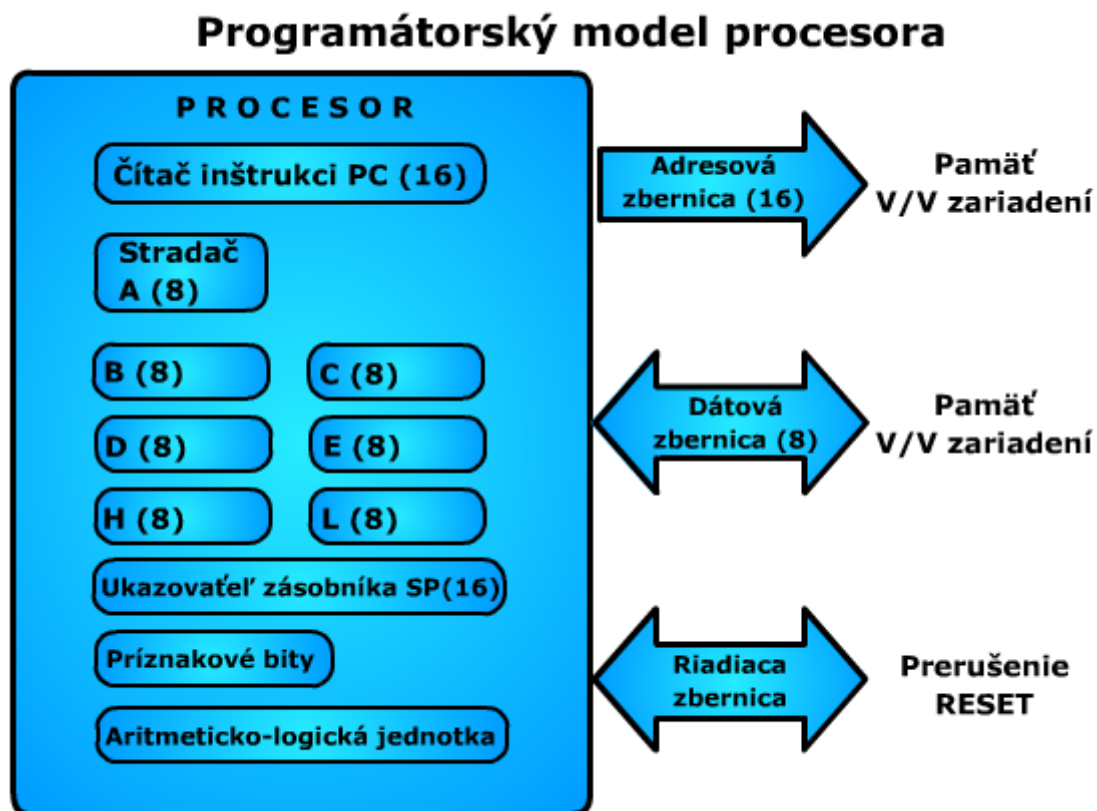
Predpokladajme, že procesor obsahuje základné registre:

- **A** (Accumulator) - stradač - 8 bitový,
- **PC** (Program Counter) - čítač inštrukcií - 16 bitový,
- sadu univerzálnych registrov B, C, D, E, H, L – 8 bitových,
- príznakové bity
  - **Z** (Zero) - = 1 pri nulovom výsledku operácie, =0 pri nenulovom
  - **S** (Sign) - Kópia znamienkového bitu výsledku operácie
  - **CY** (Carry) - Kópia bitu prenášaného z najvyššieho rádu výsledku operácie
  - **AC** (Auxiliary Carry) - prenos medzi bitom 3 a 4 výsledku
  - **P** (Parity) = 1 pri párnej parite výsledku, =0 pri nepárnej parite výsledku

Ďalej interné registre (programátorovi neviditeľné):

- **IR** - inštrukčný register (8bitový); je napojený na dekodér inštrukcií (radič),
- **DR** - údajový register (8 bitový); register pre čítanie/zápis údajov z/do pamäte,
- **AR** - adresový register (16 bitový); adresa pre čítanie/zápis z/do pamäte,
- **TA** - Temporary Address register (skladá sa z  $TA_H$  (TA High - 8 bitov),  $TA_L$  (TA Low - 8bitov)).

Príkladom takého procesora môže byť 8 bitový procesor Intel 8080.



## 3.1 Základné inštrukcie

Popíšme si teraz základné inštrukcie procesora a ich kódovania v operačnej pamäti.

Pozn. Písmeno h za číslom vyjadruje, že číslo je v hexadecimálnej sústave.

LDA adresa - Load A Direct

- naplní registr A obsahom slabiky z pamäti
- uloženie inštrukcie v pamäti:

Operačný znak      16 bitová adresa pamäte		
3Ah	nížšia slabika adresy	vyššia slabika adresy

STA adresa - Store A Direct

- Uloží register A do pamäte
- uloženie inštrukcie v pamäti

operačný znak      16 bitová adresa pamäte		
32h	nížšia slabika adresy	vyššia slabika adresy

JMP adresa - Jump Unconditional

- nepodmienенý skok na adresu
- uloženie inštrukcie v pamäti:

operačný znak      16 bitová adresa pamäte		
0DAh	nížšia slabika adresy	vyššia slabika adresy

**Príklad:** X=Y

LDA 101h

STA 100h

obsahy pamäti :

100h ..X

101h ..Y

**Ďalší príklad:**

200h: LDA 101h

203h: STA 100h

206h: ...

adresa	200h	201h	202h	203h	204h	205h	206h
obsah	3Ah	01	01	32h	00	01	...

Vykonávanie jednotlivých inštrukcií v tzv. mikroinštrukciách je nasledujúce:

Fázy inštrukcie **LDA**

- 200 > PC - počiatočné nastavenie PC
- PC > AR, 0 > WR, DR > IR - výber operačného znaku

- $PC + 1 > AR, 0 > WR, DR > TA_L$  - výber operandu
- $PC + 2 > AR, 0 > WR, DR > TA_H$  - výber operandu
- $TA > AR, 0 > WR$  - výber operandu
- $DR > A$  - vykonanie inštrukcie
- $P3 + 3 > PC$  - aktualizácia PC

#### Fázy inštrukcie **STA**

- $200 > PC$  - počiatočné nastavenie PC
- $PC > AR, 0 > WR, DR > IR$
- $PC + 1 > AR, 0 > WR, DR > TA_L$
- $PC + 2 > AR, 0 > WR, DR > TA_H$
- $A > DR$
- $TA > AR, 1 > WR$
- $PC + 3 > PC$

Inštrukcia presunu medzi registrami  $MOV R_1, R_2$  - kódovanie - 1 slabika (kombinácia registrov je súčasťou operačného znaku)

#### Fáza **MOV**

- $PC > AR, 0 > WR, DR > IR$
- $r1 > r2$
- $PC + 1 > PC$

## 3.2 Aritmetické inštrukcie

Fáza **INC** (Increment Register)

- $PC > AR, 0 > WR, DR > IR$
- $r + 1 > r$
- $PC + 1 > PC$

Fáza **ADD** (Add register to A)

- $PC > AR, 0 > WR, DR > IR$
- $A + r \vee A$
- $PC + 1 > PC$

Fáza **CMA** (Complement A) ... (= inverzia všetkých bitov registra A)

- $PC > AR, 0 > WR, DR > IR$
- $NOT A > A$
- $PC + 1 > PC$

Fáza **CMP** r (Compare Register with A)

- $PC > AR, 0 > WR, DR > IR$
- $A - r >>$  nastavenie príznakov
- $PC + 1 > PC$

Príznakové bity nastavujú inštrukcie: INR, ADD, CMA

Príznaky nemenia inštrukcie: LDA, STA, JMP, MOV



## 3.3 Podmienené skoky

Podmienené skoky sú skoky podľa obsahu príznakového registra.

Vzor inštrukcia: *podmienka adresa*

- $PC > AR, 0 > WR, DR > IR$
- if podmienka then
  - $PC + 1 > AR, 0 > WR, DR > TA_L$
  - $PC + 2 > AR, 0 > WR, DR > TA_H$
- else
  - $PC + 3 > PC$
- endif

**Inštrukcia:**

**JC** -  $CY=0$

**JNC** -  $CY=1$

**JZ** -  $Z=1$

**JNZ** -  $Z=0$

**JP** -  $S=0$

**JM** -  $S=1$

**Príklady**

X .... 100h

Y .... 101 h

**X:=X + Y**

LDA 100h  
MOV B,A  
LDA 101h  
ADD B  
STA 100h

**IF X=Y THEN ÁNO ELSE NIE**

LDA 100h  
MOV B,A  
LDA 101h  
CMP B  
JZ ÁNO  
  
:NIE  
.....  
JP VON  
:ÁNO  
....  
:VON

**IF X<=Y THEN ÁNO ELSE NIE**

LDA 100h  
MOV B,A  
LDA 101h  
CMP B  
JP ÁNO  
  
:NIE  
.....  
JP VON  
:ÁNO  
....  
:VON

**X:=X - Y**

LDA 100h  
MOV B,A  
LDA 101h  
CMA  
INR A  
ADD B  
STA 100h

**IF X<Y THEN ÁNO ELSE NIE**

LDA 100h  
MOV B,A  
LDA 101h  
CMP B  
JM ÁNO  
  
:NIE  
.....  
JP VON  
:ÁNO  
....  
:VON

**WHILE Y>=X DO B**

opakuj: LDA 100h  
MOV B,A  
LDA 102h  
CMP B  
JP blok  
JMP Koniec  
  
Blok .....  
.....  
JMP Opakuj  
  
Koniec

### **Uloženie inštrukcií v pamäti:**

for i:= 1 TO X do B;

0FFh	1	
100h	X	
102h	I	
.....		
.....		
200h	LDA 0FFh	
203h	STA 102h	; i:= 1
206h	MOV B, A	; reg. B:= i
207h	LDA 100h	
20Ah	CMP B	; X - i
20Bh	JM 300Ah	
20Eh		
....	blok B	
....		
300h	LDA 102h	
303h	INR A	
304h	STA 102h	; i:= i + 1
307h	JMP 206h	
30Ah		

## 3.4 Zásobník

Zásobník má štruktúru Last - in, First - out (LIFO) a môže byť umiestnený kdekoľvek v operačnej pamäti. Na vrchol zásobníka ukazuje register **SP** (Stack Pointer (16 bitový))

Plnenie **SP** inštrukcií **LXISP** *hodnota* Load Immediate. Fáza inštrukcie:

- $PC > AR, 0 > WR, DR > IR$
- $PC + 1 > AR, 0 > WR, DR > TAL$
- $PC + 2 > AR, 0 > WR, DR > TAH$
- $TA\ PC + 2 > AR, 0 > WR, DR > TAL\ SP$
- $PC + 3 > PC$

### Práca so zásobníkom

Inštrukcie

- **PUSH** B | D | H | PSW
- **POP** B | D | H | PSW

Fáza inštrukcie **PUSH**

- $PC > AR, 0 > WR, DR > IR$
- $SP - 1 > AR, B | D | H | PSW > DR, 1 > WR$
- $SP - 2 > AR, B | D | H | PSW > DR, 1 > WR$
- $SP - 2 > SP$
- $PC + 1 > PC$

Fáza inštrukcie **POP**

- $PC > AR, 0 > WR, DR > IR$
- $SP > AR, 0 > WR, DR > C | E | L | FI$
- $SP + 1 > AR, 0 > WR, DR > B | D | H | A$
- $SP + 2 > SP$
- $PC + 1 > PC$

**Príklad:**

**LXISP** 100h

**PUSH** B

**PUSH** D

1000h	B		
0FFFh		C	
0FFEh	D		
0FFDh		E	< SP
0FFCh			
0FFBh			
0FFAh			
..	..		
..	..		
..	..		

Pozor, žiadna kontrola podtečenia !

## Zásobník a volanie podprogramu

Inštrukcia

**CALL** adresa

**RET**

Príklad:

<i>*1</i>			
100h	<b>CALL</b>	200h	
103h	<i>*3</i>		
.....			
200h			
	<i>*2</i>		
210h	<b>RET</b>		
211h	.....		
		1000h	
		0FFEh	
		0FFCh	
		0FFAh	< SP
			<i>*1 *3</i>
		0FF8h	103h < SP
			<i>*2</i>
		0FF6h	
		....	

Fáza inštrukcie **CALL**

- PC > AR, 0 > WR, DR > IR
- PC + 3 > TA
- SP - 1 > AR, TAH > DR, 1 > WR
- SP - 2 > AR, TAL > DR, 1 > WR
- SP - 2 > SP
- PC + 1 > AR, 0 > WR, DR > TAL
- PC + 2 > AR, 0 > WR, DR > TAH
- TA > PC

Fáza inštrukcie **RET**

- PC > AR, 0 > WR, DR > IR
- SP > AR, 0 > WR, DR > TAL
- SP + 1 > AR, 0 > WR, DR > TAL
- SP + 2 > SP

TA > PC

## 3.5 Programovanie V / V operácií

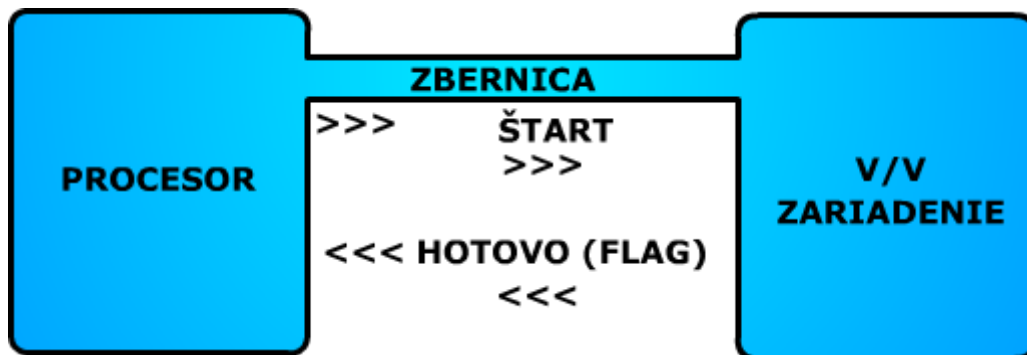
Inštrukcie

**OUT** - zapíše obsah A na V/V zbernicu

**IN** - prečíta obsah V/V zbernice do A

**START** - zahájí V /V operáciu

**FLAG** adresa - skok na adresu, ak nie je operácia hotová



Príklady:

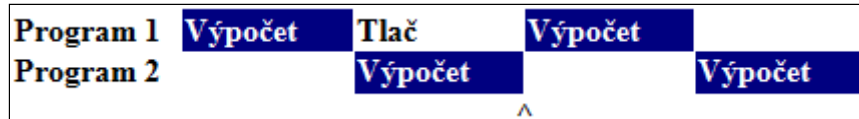
Prenos A (100h) do výstupného zariadenia

1000h	LDA 100h
1003h	OUT
1004h	START
1005h	FLAG 1005h
1008h	

Čítanie vstupného zariadenia a uloženie do A(100h)

1000h	START
1001h	FLAG 1001h
1004h	IN
1005h	STA 100h
1008h	

## 3.6 Multiprogramové spracovanie



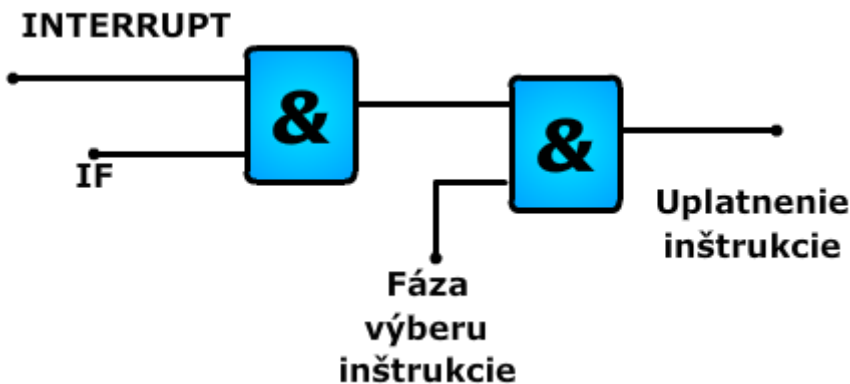
Prerušovací systém (interrupt system) umožňuje prerušenie bežiaceho procesu a aktivuje rutinu pre obsluhu prerušenia.

Činnosť pri prerušení:

1. Prerušenie vykonávania programu
2. Upratanie PC, A, .....
3. Vykonanie obslužnej rutiny
4. Obnovenie PC, A .... a tým pokračovanie vo vykonávaní programu

Kedy je možné prerušiť proces ?

- len po vykonaní inštrukcie (nie v priebehu > inštrukcia musí dokončiť všetky svoje fázy),
- ak je to povolené (každý procesor má príznak, ktorým sa prerušenie zakazuje a povoľuje). Napr. IF (Interrupt FLAG),  
Inštrukcia STI (prerušenie povolené > IF:=1)  
Inštrukcia CLI (prerušenie povolené > IF:=0),
- procesor nie je možné prerušiť bezprostredne po zahájení obsluhy predchádzajúceho prerušenia,
- prerušenie sa vyvolá signálom Interrupt.



Pri prerušení sa uplatnia tieto fázy:

- $PC > TA$
- $SP - 1 > AR, TAH > DR, 1 > WR$
- $SP - 2 > AR, TAL > DR, 1 > WR$
- $SP - 2 > SP$
- $0 > IF$
- adresa programu pre obsluhu prerušenia > PC

**Príklad konštrukcie programu pre obsluhu prerušenia**

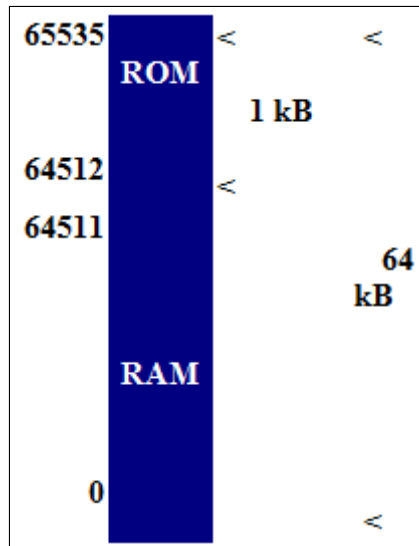
<b>100h</b>	<b>PUSH PSW</b>	<b>Upratanie registra A</b>
<b>101h</b>	<b>..</b>	<b>Obsluha</b>
<b>.....</b>	<b>..</b>	<b>prerušenia</b>
<b>.....</b>	<b>..</b>	
<b>.....</b>	<b>POP PSW</b>	<b>Obnovenie registra A</b>
<b>.....</b>	<b>STI</b>	<b>Povolenie prerušenia</b>
<b>.....</b>	<b>RET</b>	<b>Návrat do prerušeného procesu</b>



## 3.7 Signál RESET

Nastavenie počítača do počiatočných podmienok a odovzdanie riadenia zavádzajúcemu programu v permanentnej pamäti.

Príklad: Rozdelenie pamäte 'nášho' pomyselného počítača



Signál Reset sa uplatní **kdekoľvek** - tzn. aj vo vnútri fáz inštrukcie

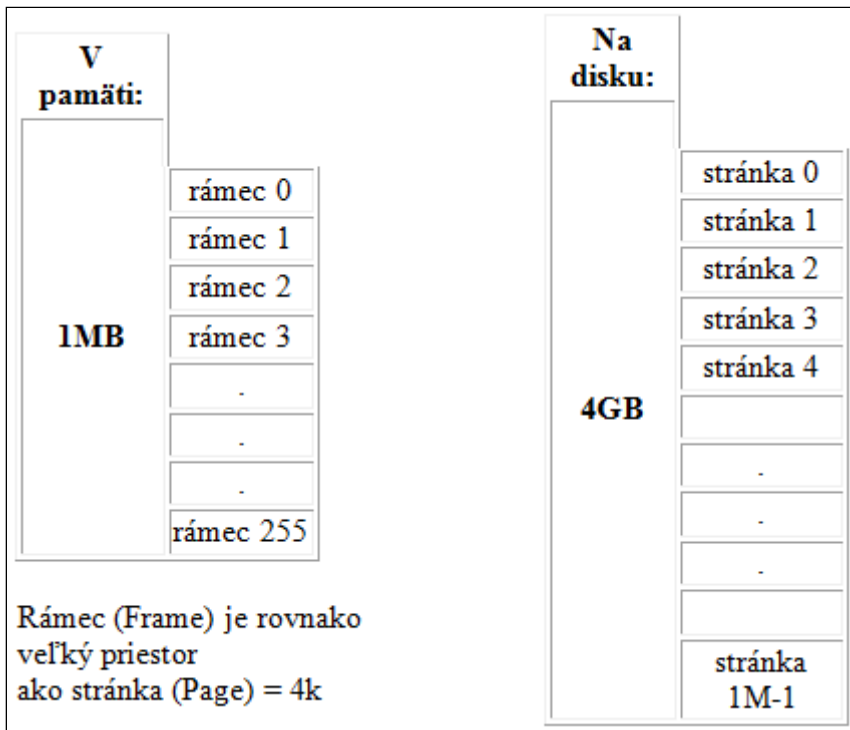
Fáza **RESET**:

- $0 > IF$  (zakázané prerušenie)
- $64512 > PC$  (skok do ROM)

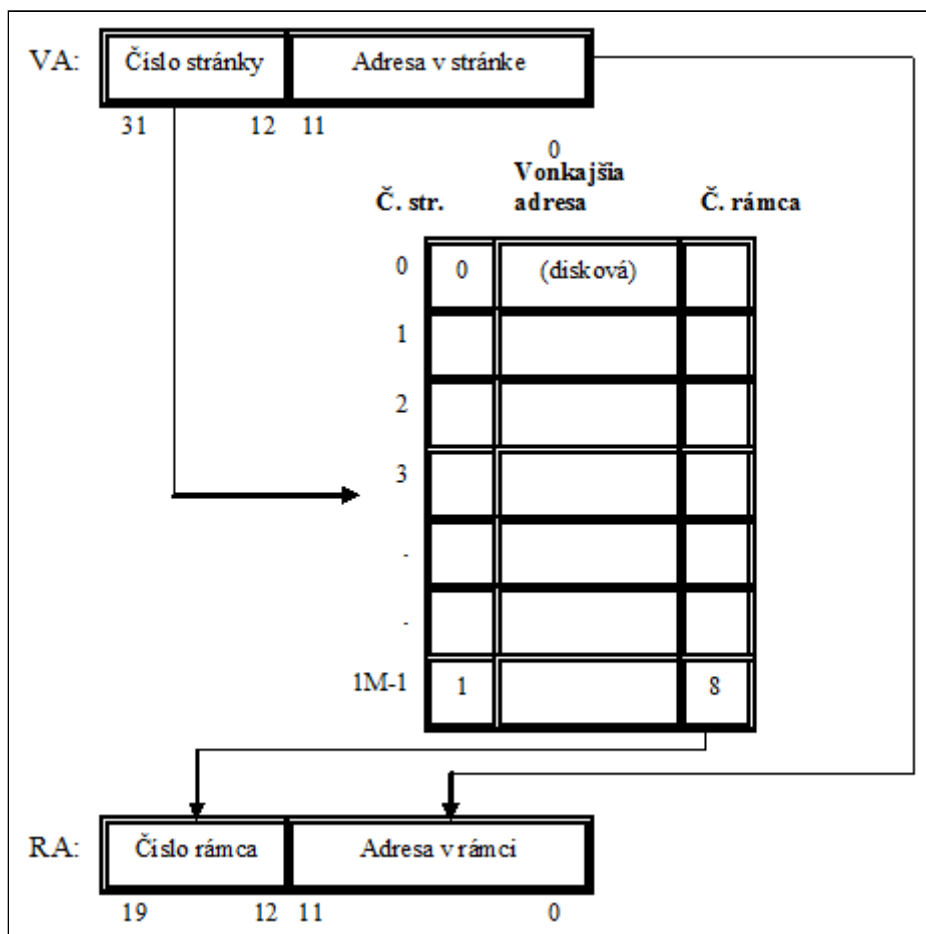
Činnosti po zapnutí počítača:

1. vyčkanie asi 1s (doba nábehu a ustálenie zdroja)
2. generovanie signálu RESET

## 3.8 Virtuálna pamäť



Každý odkaz na pamäť obsahuje virtuálnu adresu:



Tá sa transformuje cez tabuľku obsadenia rámcov na reálnu adresu

## 3.9 Adresovacie techniky

Inštrukcia:

Operačný kód

Operand(y)

*Register*

*Priamy operand*

*Priama adresa*

(totožné s MOV AH,[PROM])

*Nepriama adresa (SI, DI, SP, BP, BX)*

*Bázovaná adresa (BP, BX)*

*Indexovaná adresa (SI, DI)*

*Báza + Index (BP, BX + SI, DI)*

*Priama + Báza + Index*

**Príklad**

MOV AH,BL

MOV AH,50

PROM DB ?

MOV AH,PROM

alebo MOV AH,DS:[101]

MOV AH,[BX]

MOV AH,[BP+PROM]

MOV AH,[PROM+SI]

alebo MOV AH,PROM[SI]

MOV AH,[BX][DI]

alebo MOV AH,[BP+DI]

MOV  
AH,PROM[BX][DI]

alebo MOV  
AH,[PROM+BX+DI]

alebo MOV  
AH,[PROM+BX+DI+1]


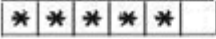
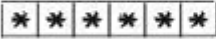

## 3.10 Inštrukcia MOV

⊕ Inštrukcia MOV príznaky nemeni!!

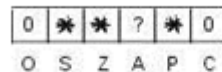


MOV r/m8,r8	MOV AL,BL	AL:=BL
	MOV Slabika,CH	Slabika:=CH
MOV r/m16,r16	MOV BX,CX	
MOV r8,r/m8		
MOV r16,r/m8		
MOV r/m16,segmentový register	MOV AX,CS	
MOV segmentový register,r/m16	MOV DS,AX	
	Nie je možné MOV CS,.. !!!	
MOV r/m8,imm8	MOV Slabika,10	
MOV r/m16,imm16		

Po inštrukcii **MOV SS,...** je po dobu trvania nasledujúcej inštrukcie zakázané prerušenie !

## 3.11 Aritmetické inštrukcie (celočíselné)

<b>ADD</b>	r/m8,imm8	ADD AL,80	AL:=AL+80
		ADD Slabika,-10	
	r/m16,imm16	ADD CX,10000	
	r/m16,imm8	ADD CX,10	<-rozšírenie s rešpektovaním znamienka
	r/m8,r8	ADD CH,CL	
	r/m16,r16	ADD AX,BX	
	r8,r/m8		
	r16,r/m16		
<b>ADC</b>	<i>ADD WITH CARRY</i>		
		ADC AL,CL	AL:=AL+CL+CF
<b>SUB</b>	<i>SUBTRACTION</i>		
		SUB AL,CL	AL:=AL-CL
<b>SBB</b>	<i>SUBTRACTION WITH BORROW</i>		
		SBB AL,CL	AL:=AL-CL-CF
<b>CMP</b>	<i>COMPARE</i>		
		CMP AL,CL	F:=AL-CL
<b>INC</b>	<i>INCREMENT</i>		
			
	INC r/m8	INC Slabika	Slabika:=Slabika+1
	INC r/m16	INC DX	
<b>DEC</b>	<i>DECREMENT</i>		
		DEC Slabika	Slabika:=Slabika-1
<b>NEG</b>	<i>DVOJKOVÝ DOPLŇEK</i>		
			
		NEG Slabika	Slabika:= -Slabika
<b>CBW</b>	<i>CONVERT BYTE TO WORD</i>		
			
		CBW	AX:=AL se zachovaním znaménka.
<b>CWD</b>	<i>CONVERT WORD TO DOUBLEWORD</i>		

## 3.12 Logické inštrukcie

<b>AND</b>	<i>LOGICKÝ SÚČIN PO BITOCH</i>	
	kombinácia parametrov vid'. "ADD"	AND AL,7    AL:=AL $\wedge$ 7 AND Slovo,1FFFh    Slovo:=Slovo $\wedge$ 1FFFh
<b>OR</b>	<i>LOGICKÝ SÚČET</i>	OR AL,7    AL:=AL $\vee$ 7
<b>XOR</b>	<i>NONEKVIVALENCIA</i>	XOR AL,7    AL:=AL $\oplus$ 7
<b>NOT</b>	<i>INVERZIA BITOV</i>	
	<i>(JEDNOTKOVÝ DOPLNOK)</i>	
	NOT r/m 8	NOT AH    AH:= $\overline{AH}$ NOT Slab    Slab:= $\overline{Slab}$
	NOT r/m16	NOT SI    SI:= $\overline{SI}$ NOT Slovo    Slovo:= $\overline{Slovo}$
<b>TEST</b>	<i>LOGICAL COMPARE</i>	
	TEST r/m8,imm8	TEST AL,7    F:=AL $\wedge$ 7 TEST Slab,15    F:=Slab $\wedge$ 15
	TEST r/m16,imm16	
	TEST r/m8,r8	
	TEST r/m16,r16	
<b>AND, OR, XOR</b> ..... r/m16,imm8 ..... znamienkové rozšírenie		

## 3.13 Rotácia

### ROL *ROTATE LEFT*

ROL r/m8,1  
 ROL r/m8,CL  
 ROL r/m16,1  
 ROL r/m16,CL

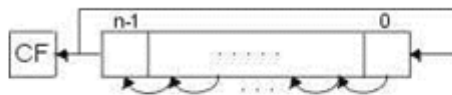
**8086 : CL**

**neomedzené**

**286,.. : CL  $\wedge$  1Fh**

OF je definované len pri rotácii o 1 bit:

ROL:  $OF := CF \oplus \text{bit}_{n-1}$   
 tj. OF sa nastaví, pokiaľ sa hodnota CF nerovná novému najvyššiemu bitu.



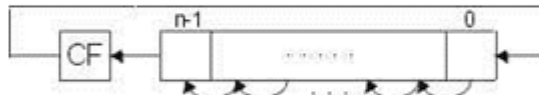
### ROR *ROTATE RIGHT*

ROR:

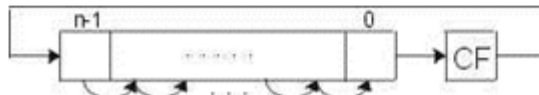
$OF := \text{bit}_{n-1} \oplus \text{bit}_{n-2}$



### ROL *ROTATE LEFT THROUGH CARRY*



### RCR *ROTATE RIGHT THROUGH CARRY*



## 3.14 Posuvy

<b>SAL</b>	<i>SHIFT ARITHMETIC LEFT</i>	Obidve vykonajú rovnakú akciu.
<b>SHL</b>	<i>SHIFT LOGICAL LEFT</i>	
<div> <div> <div>*</div><div>*</div><div>*</div><div>?</div><div>*</div><div>*</div> </div> <div> <div>O</div><div>S</div><div>Z</div><div>A</div><div>P</div><div>C</div> </div> </div>	Varianty vid'. Inštrukcia "ROL" znamienko aritmetického násobenia $2^n$ $OF := CF \oplus \text{bit}_{n-1}$	
<b>SAR</b>	<i>SHIFT ARITHMETIC RIGHT</i>	
	$OF := 0$	
<b>SHR</b>	<i>SHIFT LOGICAL RIGHT</i>	
	$OF := \text{pôvodný bit}_{n-1}$	

tekst



## 3.15 Vetvenie programu

JMP

JUMP=NEPODMIENENY SKOK

- **priamy skok** (cieľová adresa je v inštrukcii)
  - **mení CS** - vzdialený skok (far jump)  
- plní CS:IP
    - **nemení CS** - IP := IP + vzdialenosť
      - vzdialenosť <0; 65535>

sčíta sa neznamienkovo!!  
= blízky skok (near jump)
  - vzdialenosť <-128; +127>

sčíta sa znamienkovo!!  
= krátky skok (short jump)
- **nepriamy skok** : v inštrukcii je odkaz na:
  - Register SI, DI, SP, BP alebo BX (obsahuje offset cieľové adresy)
  - Pamäť - segment : offset  
- offset

JMP rel8	JMP SHORT	krátky skok
	návestie	IP:=IP+vzdialenosť
		návestia
JMP rel16	JMP návestie	blízky skok
		IP:=IP+vzdialenosť
		návestia
JMP ptr16:16	JMP FAR_PTR	vzdialený skok
	návestie	CS:IP:= segment:offset
		návestia
JMP r/m16	JMP [BX]	nepriamy blízky skok
		IP:=BX
	JMP [slovo]	IP:=slovo
JMP m16:16	JMP [dvojslovo]	CS:IP:=dvojslovo
		nepriamy skok vzdialený

## 3.16 Podmienené skoky

### JUMPS CONDITIONAL

Len krátke skoky: vzdialenosť <-128; +127>

⊕ Neuvádza sa "short"

	JUMP SHORT IF...	TESTOVANÁ PODMIENKA	VÝSLEDOK POSLEDNEJ OPERÁCIE
<b>JE</b>	<i>EQUAL</i>	ZF=1	rovný
<b>JZ</b>	<i>ZERO</i>		nulový
<b>JNE</b>	<i>NOT EQUAL</i>	ZF=0	rôzny
<b>JNZ</b>	<i>NOT ZERO</i>		nenulový
<b>JP</b>	<i>PARITY</i>	PF=1	párna
<b>JPE</b>	<i>PARITY EVEN</i>		parita
<b>JNP</b>	<i>NOT PARITY</i>	PF=0	nepárna
<b>JPO</b>	<i>PARITY ODD</i>		parita
<b>JS</b>	<i>SIGNUM</i>	SF=1	Záporný
<b>JNS</b>	<i>NOT SIGNUM</i>	SF=0	kladný ∨ nulový
<b>JC</b>	<i>CARRY</i>	CF=1	nastal prenos
<b>JNC</b>	<i>NOT CARRY</i>	CF=0	nenastal prenos
<b>JO</b>	<i>OVERFLOW</i>	OF=1	nastalo pretečenie
<b>JNO</b>	<i>NOT OVERFLOW</i>	OF=0	nenastalo pretečenie
<b>JB</b>	<i>BELOW</i>	CF=1	nz. Menší
<b>JNAE</b>	<i>NOT ABOVE NOR EQUAL</i>		
Príklad:            +2 : 0 1 0   nz. +5 : 1 0 1			
			CMP 2, 5            0 1 0 ZF=0 <u>- 1 0 1</u> OF=1 nz. 2<5            1 1 0 1 SF=1  CF=1
<b>JL</b>	<i>LESS</i>	SF ≠ OF	z. menší
<b>JNGE</b>	<i>NOT ABOVE NOR EQUAL</i>		
Príklad:            +2 : 0 1 0   z. -3 : 1 0 1			
			CMP 2, -3    z: 2 ≠ -3 CF=1 SF=1 OF=1 ZF=0  CMP -3, 2            1 0 1 ZF=0 <u>- 0 1 0</u> OF=1 0 0 1 1  SF=0 CF=0
<b>JA</b>	<i>ABOVE</i>	(CF=0) ∧ (ZF=0)	nz. väčší
<b>JNBE</b>	<i>NOT BELOW NOR EQUAL</i>		
<b>JG</b>	<i>GREATER</i>	(SF=OF) ∧ (ZF=0)	z. väčší
<b>JNLE</b>	<i>NOT LESS NOR EQUAL</i>		
<b>JBE</b>	<i>BELOW OR</i>	(CF=1) ∨ (ZF=1)	nz. menší alebo rovný
<b>JNA</b>	<i>EQUAL NOT ABOVE</i>		
<b>JLE</b>	<i>LESS OR EQUAL</i>	(SF ≠ OF) ∨ (ZF=1)	z. menší alebo rovný
<b>JNG</b>	<i>NOT GREATER</i>		
<b>JAЕ</b>	<i>ABOVE OR</i>	CF=0	nz. väčší alebo rovný
<b>JNB</b>	<i>EQUAL NOT BELOW GREATER OR</i>		

<b>JGE</b>	<i>GREATER OR</i>		
<b>JNL</b>	<i>EQUAL</i>	SF=OF	z. väčší alebo rovný
	<i>NOT LESS</i>		
<b>JCXZ</b>	<i>JUMP SHORT IF</i>		
	<i>CX=0</i>		
	používa sa pre		
	riadenie cyklov		