

## Strojový kód

**instrukce** = příkaz (zakódovaný jako „číslo“)

Obsahuje (popř. může obsahovat) tyto informace:

- ① **co** se má provést (jaká operace)
- ② **s čím** se to má provést (operandy) a **kam** se má uložit výsledek
- ③ **kde** se má pokračovat (adr. násl. instr.)

Tyto informace mohou být obsaženy:

- explicitně v instrukci
    - Př.: počítač SAPO — 5 adresové instrukce:
      - ① ... tzv. operační znak
      - ② ... 2+1 adresa
      - ③ ... 2 adresy — následující instrukce při záporném a nezáporném výsledku
  - zčásti explicitně v instrukci, zčásti určeny implicitně architekturou procesoru, např.:
 

<ol style="list-style-type: none"> <li>① ... <b>operační znak</b> — OZ</li> <li>② ... <b>adresová část</b> instrukce</li> <li>③ ... von Neumannova koncepce ⇒ další instrukce na následující adrese ← architektura</li> </ol>	}	instrukce
---	---	-----------
- operační kód** = **soubor instrukcí**: OZ ~ operace

## Strojový kód

dále předpokl.: von Neumannova koncepce ⇒

⇒ **programový čítač PC** [Program Counter] obsahuje adresu následující instrukce

⇒ skoky (nepodmíněné a podmíněné) „neproduktivní“ operace

⇒ kratší instrukce

**3 adresová instrukce:**

OZ	$a_1$	$a_2$	$a_3$
----	-------	-------	-------

- „nejpřirozenější“: 2 operandy + 1 výsledek  
např.:  $\langle a_1 \rangle - \langle a_2 \rangle \rightarrow a_3$
- poměrně dlouhá

**2 adresová instrukce:**

OZ	$a_1$	$a_2$
----	-------	-------

- výsledek se ukládá na místo prvního operandu  
např.:  $\langle a_1 \rangle - \langle a_2 \rangle \rightarrow a_1$
- je třeba zavést „neproduktivní“ operaci přesun:  
 $\langle a_2 \rangle \rightarrow a_1$

Př.:  $\langle x \rangle - \langle y \rangle \rightarrow z \equiv \begin{cases} \langle x \rangle \rightarrow z \\ \langle z \rangle - \langle y \rangle \rightarrow z \end{cases}$

## Strojový kód

**1 adresová instrukce:**

OZ	$a$
----	-----

- zvl. registr — **střadač S** [Accumulator]
- 1. operand a výsledek např.:  $\langle S \rangle - \langle a \rangle \rightarrow S$
- operace přesunu:  $\langle a \rangle \rightarrow S$      $a$      $\langle S \rangle \rightarrow a$

Př.:  $\langle x \rangle - \langle y \rangle \rightarrow z \equiv \begin{cases} \langle x \rangle \rightarrow S \\ \langle S \rangle - \langle y \rangle \rightarrow S \\ \langle S \rangle \rightarrow z \end{cases}$

více střadačů ⇒ číslo střadače ∈ instrukce  
⇒ lze zavést operace mezi střadači

Př.: Motorola — řada 68000  
datové registry  $D_0, D_1, \dots, D_7$  à 32 b  
odčítání 32 b:  $D_n - \text{paměť} \rightarrow D_n$

OZ: 

9	$n$	0	B	9
---	-----	---	---	---

instrukce: 

OZ	adresa
----	--------

$\langle D_3 \rangle - \langle 18 \text{ FF20} \div 18 \text{ FF23} \rangle \rightarrow D_3$   
96B9 0018 FF20

## Zápis programu

**strojový kód:** strojové instrukce — „čísla“

Př.: Intel 8086 (zjednodušeno)  
**AX** — střadač 16 b  
A10401  $\langle 104 \div 105 \rangle \rightarrow \text{AX}$   
2B060601  $\langle \text{AX} \rangle - \langle 106 \div 107 \rangle \rightarrow \text{AX}$   
A30201  $\langle \text{AX} \rangle \rightarrow 102 \div 103$

- pracné programování i změny
- nepřehledný zápis

**vyšší programovací jazyk (VPJ)**  
(např. Pascal)

Př.:  $a := b - c$

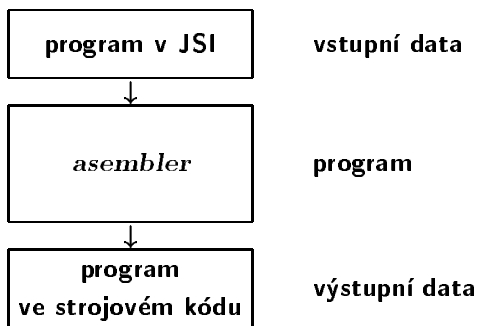
- omezení (zejm. pro „speciální“ účely)
- méně efektivní (?)

**jazyk symbolických instrukcí (JSI):**

operační znak i adresy zapsány symbolicky

Př.: MOV AX,b  
SUB AX,c  
MOV a,AX

## Vytváření programů ve strojovém kódu

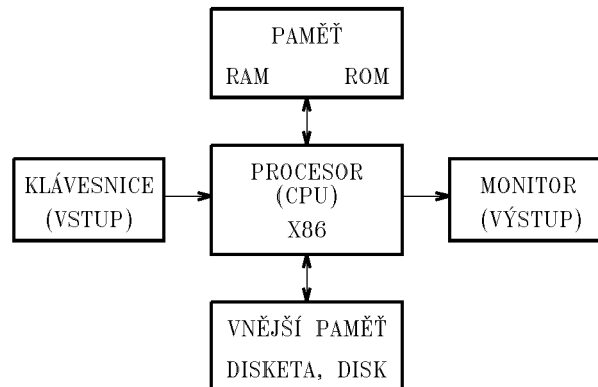


VPJ: analogicky místo assembleru jiný program — překladač (kompilátor)

někdy:

program v JSI } → překladač → modul  
 program ve VPJ }  
 moduly → spojovací program [linker] → stroj. kód

## Ideové schéma technické platformy (IBM PC-kompatibilní osobní počítač)



## Řada procesorů Intel 80x86

rok		tranz. [tis.]	MIPS	při MHz	dat.	adr.
1971	*4004	2,3	0.06	0,108	4	10
1978	8086	29	0.33	5	16	20
1979	8088	29	0.33	5	8	20
1982	80286	134	1.2	8	16	24
1985	80386DX	275	6	16	32	32
1988	80386SX	275	2.5	16	16	24
1989	i486DX	1200	20	25	32	32
1993	Pentium	3100	112	66	64	32
1995	Pentium Pro	5500	300	133	64	36
1997	Pentium II	7500		300	64	36
1999	Pentium III	28000		733	64	36

1981 IBM PC  
 1984 IBM PC-AT

## Typy instrukcí X86 (=INSTRUKČNÍ SOUBOR)

- přesun dat (MOV,...,IN,OUT,...PUSH,POP,CLI,..)
- aritmetické i. (ADD,...,INC,..)
- logické i. (AND,...,XOR,...)
- posuvy (SHL,...,RCR,...)
- skoky (JMP,...,JC,JNC,...)
- cykly (LOOP,...,LOOPZ,...)
- podprogramy (CALL,RET,...)
- přerušení (INT,IRET,...)

- instrukce pro práci s řetězci
- pseudoinstrukce (např. deklarace proměnných DB,DW,..)
- makroinstrukce



## aritmetické operace

Operandy a výsledek lze interpretovat jako:

- čísla v doplňkovém kódu, např. (pro 1 B = 8 b):

$$FF_{16} \sim (-1) \in \langle -128_{10}; 127_{10} \rangle$$

- nezáporná čísla, např. (pro 1 B = 8 b):

$$FF_{16} \sim FF_{16} = 255_{10} \in \langle 0; 255_{10} \rangle$$

$$\boxed{\text{ADD } \alpha, \beta} \quad \dots \quad \alpha := \alpha + \beta$$

$\alpha$ : datový reg. ukazatel paměť       $\beta$ : datový reg. ukazatel paměť konstanta

nelze: paměť := paměť + paměť

příznaky CF, OF, SF a ZF:

Př.: ADD AL, BL

nezáporná č.			doplňkový kód			příznaky			
AL	BL	AL+BL	AL	BL	AL+BL	CF	OF	SF	ZF
7A	78	0F2	~7A	~78	~(-E)	0	1	1	0
7A	FF	179	~7A	~(-1)	~79	1	0	0	0

pozn.: dále se nastavují příznaky PF a AF

## aritmetické operace

$$\boxed{\text{ADC } \alpha, \beta} \quad \dots \quad \alpha := \alpha + \beta + \text{CF}$$

$$\boxed{\text{SUB } \alpha, \beta} \quad \dots \quad \alpha := \alpha - \beta$$

analogické ADD, až na **příznak CF**:

$$\alpha < \beta \iff \text{CF} := 1 \quad (\alpha, \beta \text{ — nezáporná čísla})$$

zde: CF ... tzv. výpůjčka [borrow]

$$\boxed{\text{SBB } \alpha, \beta} \quad \dots \quad \alpha := \alpha - \beta - \text{CF}$$

$$\boxed{\text{CMP } \alpha, \beta} \quad \dots \quad \alpha - \beta$$

stejně jako SUB, ale neukládá se výsledek — nastaví se však **příznaky** !

$$\boxed{\text{NEG } \alpha} \quad \dots \quad \alpha := 0 - \alpha = -\alpha$$

(uvažuje se doplňkový kód)

Př.: MOV AL, 1 ... 0000 0001

NEG AL ... 1111 1111 = 0FFH

CF=1, ZF=0, SF=1, OF=0

$$\boxed{\text{INC } \alpha} \quad \dots \quad \alpha := \alpha + 1 \text{ nemění se příznak CF}$$

$$\boxed{\text{DEC } \alpha} \quad \dots \quad \alpha := \alpha - 1 \text{ nemění se příznak CF}$$

## logické operace

$$\boxed{\text{AND } \alpha, \beta} \quad \dots \quad \alpha := \alpha \wedge \beta \quad \text{logický součin}$$

$$\boxed{\text{OR } \alpha, \beta} \quad \dots \quad \alpha := \alpha \vee \beta \quad \text{logický součet}$$

$$\boxed{\text{XOR } \alpha, \beta} \quad \dots \quad \alpha := \alpha \oplus \beta \quad \text{součet modulo 2}$$

příznaky: CF := 0  
SF ... výsl. < 0  
ZF ... výsl. = 0  
OF := 0

	0	1	0	1
	0	0	1	1
AND	0	0	0	1
OR	0	1	1	1
XOR	0	1	1	0

$\boxed{\text{TEST } \alpha, \beta} \quad \dots \quad \alpha \wedge \beta$  stejné jako AND, ale neukládá se výsledek — nastaví se však **příznaky** !

$$\boxed{\text{NOT } \alpha} \quad \dots \quad \alpha := \bar{\alpha} \quad \text{negace } (\bar{0} = 1; \bar{1} = 0)$$

Př.: MOV DH, 13H ... 0001 0011

XOR DH, 86H ... 1000 0110

1001 0101 = 95H

ZF=0, SF=1

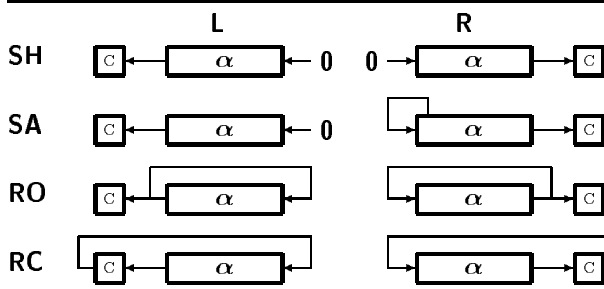
NOT DH ... 0110 1010 = 6AH

ZF=0, SF=0

## posuvy

posuv { logický SH [SHift]  
aritmetický SA [Shift Arithmetic]  
cyklický RO [ROtate]  
cyklický přes C RC [RotatE through C]

posuv { vlevo L [Left]  
vpravo R [Right]



$$\boxed{\text{SHL } \alpha, o\_kolik} \quad o\_kolik = \begin{cases} 1 \\ \text{CL} \end{cases}$$

analogicky: SHR, SAL, ... RCL

kromě příznaku C  $\equiv$  CF se mění některé další příznaky

Př.: MOV AX, 1234H ... AX = 1234H

MOV CL, 4

ROL AX, CL ... AX = 2341H CF=1

## skoky

**nepodmíněný skok** (nejjednodušší varianta):

**JMP *náv*** *náv* ... návěští instrukce

Př.:           MOV AX,1           ①  
               JMP NAVESTI   ②  
 COSI:       ADD BX,CX  
 NAVESTI:   RCR AX,1       ③

**podmíněné skoky:**

**JC *náv*** ... je-li C=1, skok na *náv* (jinak nic)

**JNC *náv*** ... je-li C=0, skok na *náv* (jinak nic)

Př.:   AX := max (BX,CX) — nezáporná čísla:

MOV AX,BX  
 CMP CX,BX  
 JC OK  
 MOV AX,CX

OK:

analogicky: JZ, JNZ, JS, JNS, JO, JNO, JP, JNP

## skoky

**podmíněné skoky orientované na srovnání:**

relace	doplňkový kód	nezáporná čísla
<	JL	JB
≤	JLE	JBE
=	JE	JE
≠	JNE	JNE
>	JGE	JAЕ
>	JG	JA

E ... Equal  
 L ... Less  
 G ... Greater  
 B ... Bellow  
 A ... Above

podmínka skoku  $\Leftarrow$  příslušné příznaky, např.:

JB ... CF = 1  $\Rightarrow$  JB  $\equiv$  JC

JBE ... CF $\vee$ ZF = 1

JL ... SF $\oplus$ OF = 1

JGE ... SF $\oplus$ OF = 0 atd.

pozn.: v předch. příkl. lze JC nahradit JB

Př.: AX := max (BX,CX) — doplňkový kód :

řešení: v předch. příkl. JC nahradit JL

**rozsah podm. skoků:** -128  $\div$  127 vůči IP

Př.: JC OK  
       MOV AX,CX  $\leftarrow$  IP  $\uparrow$ 128 $\uparrow$   $\downarrow$ 127 $\downarrow$

**„delší“ skoky:** opačný podm. skok + nepodm. skok

## cykly

Př.:           MOV CX,300  
       CYKL:   :  
               DEC CX       ; mění příznaky  
               JNZ CYKL

**LOOP *náv*** CX := CX - 1;  
           je-li CX  $\neq$  0, skok na *náv*

**nemění příznaky**

Př.:           MOV CX,300  
       CYKL:   :  
               LOOP CYKL   ; nemění příznaky

**LOOPE *náv*** CX := CX - 1;  
           je-li CX  $\neq$  0 a ZF = 1, skok na *náv*

**LOOPNE *náv*** CX := CX - 1;  
           je-li CX  $\neq$  0 a ZF = 0, skok na *náv*

**LOOPZ *náv*** jako LOOPE *náv*

**LOOPNZ *náv*** jako LOOPNE *náv*

rozsah skoku stejný jako u podmíněných skoků