


Kompiuterių architektūra.

Pratybos 2018 09 24,26,27

8086/88 instrukcijos

*Jeigu nieko nesigauna ir dar nieko
nesulaužei – perskaityk pagaliau
instrukciją*



CPU instrukcijos

Instrukcijų klasės

- Duomenų pernešimų instrukcijos
- Aritmetinės
- Loginės
- Manipuliacijos su sekomis (baitų/žodžių)
- Valdymo kontrolės perdavimo
- Procesoriaus kontrolės

Duomenų pernešimo instrukcijos

Instrukcija MOV <op1>, <op2>

- <op1> - registras, <op2> - registras
 - Pvz.: mov ax, bx
- <op1> - registras, <op2> - atmintis / arba atvirkščiai
 - Pvz., mov cx, word skaicius
 - ... mov bl, byte skaicius [si]
 - ... mov byte [bp+2], ch

Duomenų pernešimo instrukcijos

...tęsinys

- <op1> - registras (nesegmentinis), <op2> - atitinkamo ilgio skaičius
 - Pvz.: `mov ax, 1233h`
 - ... `mov dl, 'A'`
- <op1> - atmintis, <op2> - atitinkamo ilgio skaičius
 - Pvz., `mov word skaicius, 0ABCDh`
 - ... `mov byte skaicius[si], 0FFh`

Duomenų pernešimo instrukcijos

xchg : sukeičia operandus vietomis

- xchg <atminties operandas>, <registras>
 - xchg bute ptr sk, al
- xchg <registras> , <atminties operandas>

Prieš nagrinėjant instrukcijų sistemą

- Daug instrukcijų apart tiesioginio rezultato, pvz., dviejų skaičių sudėjimo, papildomai gali turėti tam tikrus požymius, pvz., ar įvyko perpildymas (overflow), pernešimas (carry), rezultato ženklo (sign) ir kt. Ši informacija rašoma POŽYMIŲ registre, kuris vadinamas FLAGS.

FLAGS registras

Požymiai:

1. Carry flag (CF),
2. Parity flag (PF),
3. Auxiliary flag (AF),
4. Zero flag (ZF),
5. Sign flag (SF),
6. Trap flag (TF),
7. Interrupt flag (IF),
8. Direction flag (DF),
9. Overflow flag (OF)

Požymių registras

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z	-	A	-	P	-	C
				F	F	F	F	F	F		F		F		F

Konkrečiai kiekvieno požymio reikšmę matysime iš instrukcijų pavyzdžių

Instrukcijos su FLAGS

- Galima pakrauti **J** AH:
 - **LAHF**
- Galima pakrauti **IŠ** AH:
 - **SAHF**

..tęsinys

- Nustatymo/valymo instrukcijos:
 - CLC/STC – išvalyti (padaryti nulių)/nustatyti (padaryti 1) CF;
 - CMC – CF invertavimas;
 - CLD/STD – išvalyti/nustatyti DF;
 - CLI/STI – išvalyti/nustatyti IF



MOV instrukcijos

Pagrindinis šaltinis

- Oficialioji Intel dokumentacija:
 - <http://datasheets.chipdb.org/Intel/x86/808x/datashts/8088/231456-006.pdf>
 - **Nuo 26 puslapio;**
 - Šia atspausdinta medžiaga galima bus naudotis per INSTRUKCIJŲ testą (spalio pabaigoje)
 - Taip pat – labai rimtas šaltinis apie dabartinius Intel procesorius:
 - <http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>

Instrukcijų žinojimo svarba

- Klausimas: ar galima rašyti kur nors:
 - `Mov [bx], al`
 - `Arba Mov [dx], al`
 - `Arba Mov [ah], al ?`
 -

Adreso formavimo idioma

- Formuojant adresą galima naudotis tik griežtą („trijų stulpelių“) schema:

POSLINKIS

8 ar 16
bitų
skaičius

BAZĖ

Tik vienas iš

BP

arba

BX

INDEKSAS

Tik vienas iš

SI

arba

DI

Galimi adreso formavimo būdai

[BX+SI]

[BX+DI]

[BP+SI]

[BP+DI]

[SI]

[DI]

posl 16

[BX]

[BX+SI]+posl8

[BX+DI]+posl8

[BP+SI]+posl8

[BP+DI]+posl8

[SI]+posl8

[DI]+posl8

[BP]+posl8

[BX]+posl8

[BX+SI]+posl16

[BX+DI]+posl16

[BP+SI]+posl16

[BP+DI]+posl16

[SI]+posl16

[DI]+posl16

[BP]+posl16

[BX]+posl1

posl8 -- baitas su ženkle, t. y., nuo -128 iki 127;
posl16 -- visada beženklis skaičius !

Pastaba apie poslinkius

- Programuojant assembleriu poslinkis gali formuotis įvairiais būdais:

```
zodis      dw    0x1234
```

```
du_zodziai dw    0x0001, 0x0002
```

```
...
```

```
mov cx, word zodis
```

```
                ;pakraus 1234h
```

```
mov dx, word  du_zodziai + 1
```

```
                ;pakraus 0100h
```

Pastabos

- Pagal nutylėjimą, jeigu adreso formavime dalyvauja BP registras, tai atskaitos segmentas yra **SS**, kitais atvejais – **DS**;
- Procesoriuose nuo 386-to galima naudoti ir sudėtingesnius adreso formavimo būdus;
- REALUS (efektyvus) adresas suformuotas vienokiu ar kitokiu būdu priklauso nuo bazinių ir indeksinių registrų reikšmių: tiesiog reikia sumuoti registrų ir poslinkio reikšmes ir atsižvelgti į atitinkamo segmento registro reikšmę

Kaip formuojamas instrukcijos kodas?

- Instrukcijos ilgis – iki 6 baitų, bet dar gali būti PREFIKSAS(-I);
 - Mes panagrinėsime keletą instrukcijų – visas kitas – per paskaitas :)

Registrai: instrukcijų apraše naudojami specialūs žymėjimai, REG – vienas iš jų

REG is assigned according to the following table:

16-Bit ($w = 1$)	8-Bit ($w = 0$)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Registrai: instrukcijų apraše naudojami specialūs žymėjimai, REG – vienas iš jų

REG is assigned according to the following table:

16-Bit ($w = 1$)	8-Bit ($w = 0$)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instrukcijos veikimo kryptis ir operandų tipas aprašė

if $d = 1$ then “to” reg; if $d = 0$ then “from” reg
if $w = 1$ then word instruction; if $w = 0$ then byte
instruction

Komanda **mov**

MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/Memory to/from Register	1 0 0 0 1 0 dw	mod reg r/m		
Immediate to Register/Memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate to Register	1 0 1 1 w reg	data	data if w = 1	
Memory to Accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to Memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/Memory to Segment Register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment Register to Register/Memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		

Ką reiškia mod laukas?

if mod = 11 then r/m is treated as a REG field

if mod = 00 then DISP = 0*, disp-low and disp-high are absent

Bus skaidrė su pastaba

if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent

if mod = 10 then DISP = disp-high; disp-low

Ką reiškia r/m laukas?

if $r/m = 000$ then $EA = (BX) + (SI) + DISP$

if $r/m = 001$ then $EA = (BX) + (DI) + DISP$

if $r/m = 010$ then $EA = (BP) + (SI) + DISP$

if $r/m = 011$ then $EA = (BP) + (DI) + DISP$

if $r/m = 100$ then $EA = (SI) + DISP$

if $r/m = 101$ then $EA = (DI) + DISP$

if $r/m = 110$ then $EA = (BP) + DISP^*$

if $r/m = 111$ then $EA = (BX) + DISP$

DISP follows 2nd byte of instruction (before data if required)

Bus skaidrė su pastaba

DISP reiškia tiesiog **poslinkį** nuo segmento pradžios
EA – efektyvus (nuo segmento pradžios) adresas

Pastaba

Jeigu **mod = 00 IR r/m = 110**,
tai **EA = [DISP-LOW][DISP-HIGH]**



Efektyvus adresas

DISP reiškia tiesiog **poslinkį** nuo segmento pradžios
EA – efektyvus (nuo segmento pradžios) adresas

MOD reg r/m bairas; MOD = 00

[BX+SI]	000
[BX+DI]	001
[BP+SI]	010
[BP+DI]	011
[SI]	100
[DI]	101
posl 16	110
[BX]	111

MOD reg r/m baitas; MOD = 01

[BX+SI]+posl8	000
[BX+DI]+posl8	001
[BP+SI]+posl8	010
[BP+DI]+posl8	011
[SI]+posl8	100
[DI]+posl8	101
[BP] +posl8	110
[BX]+posl8	111

MOD reg r/m baitas; MOD = 10

[BX+SI]+pos16	000
[BX+DI]+pos16	001
[BP+SI]+pos16	010
[BP+DI]+pos16	011
[SI]+pos16	100
[DI]+pos16	101
[BP] +pos16	110
[BX]+pos16	111

MOD reg r/m bairas; MOD = 11

AX/AL	000
CX/CL	001
DX/DL	010
BX/BL	011
SP/AH	100
BP/CH	101
SI/DH	110
DI/BH	111

Komanda **mov**: 1 atvejis

Register/Memory to/from Register

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 d w	mod reg r/m
-----------------	-------------

Komanda **mov**: 1 atvejis

Register/Memory to/from Register

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 d w	mod reg r/m
-----------------	-------------

Pavyzdys

- Užkoduokime

– **mov bx, [1234h+si]**

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 d w

mod reg r/m

d=1, nes „to register“

w=1, nes bx yra 16 bitų

...tęsinys

- Užkoduokime

– **mov bx, [1234h+si]**

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 d w	mod reg r/m
-----------------	-------------

mod=10, nes kiti variantai NETINKA

...tęsinys

- Užkoduokime

– **mov bx, [1234h+si]**

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 dw

mod reg r/m

reg=011, nes BX

r/m=100, nes [SI+DISP]

...rezultatas

- Užkoduokime

– **mov bx, [1234h+si]**

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
1 0 0 0 1 0 dw	mod reg r/m
1 0 0 0 1 0 1 1	1 0 0 1 1 1 0 0

8B 9C 34 12

Pratimas

- Užkoduokite instrukcijas (pasitikrinkite su **debug** arba su **td**)
 - mov cx,ax
 - mov si,bx

Komanda **mov.** Antras atvejis

Immediate to Register/Memory

1100011w	mod000r/m	data	data if w = 1
----------	-----------	------	---------------

Pavyzdys/pratimas

- `mov word ptr [si+1234h], 789ah`
- `mov byte ptr [si+1234h], 78h`

Trečias mov atvejis

Immediate to Register

1011w reg	data	data if w = 1
-----------	------	---------------

Pratimas:

```
mov cx, 1234h
```

```
mov cl, 34h
```

```
mov bp, 15h
```

Ketvirtas **mov** atvejis

Memory to Accumulator	1010000w	addr-low	addr-high
Accumulator to Memory	1010001w	addr-low	addr-high

Jeigu $w=1$, tai AX,
jeigu $w=0$, tai AL

Penktas **mov** atvejis

Register/Memory to Segment Register

1 0 0 0 1 1 1 0

mod 0 reg r/m

Segment Register to Register/Memory

1 0 0 0 1 1 0 0

mod 0 reg r/m

Pratimas

mov ds, cx

mov [1234h], es

Penktas **mov** atvejis

Register/Memory to Segment Register

1 0 0 0 1 1 1 0

mod 0 reg r/m

Segment Register to Register/Memory

1 0 0 0 1 1 0 0

mod 0 reg r/m

Pratimas

mov ds, cx

mov [1234h], es



Aritmetinės ir loginės instrukcijos

Mnemonikos

- ADD
- SUB
- ADC
- SBB
- MUL
- IMUL
- DIV
- IDIV
- INC
- DEC
- NEG
- AND
- OR
- NOT
- XOR
- SHL/SAL
- SHR
- SAR
- TEST
- ROL
- ROR
- RCR
- RCL
- ...

Kai kurios instrukcijos

Tolimesnėse skaidrėse panagrinėsime kai kurias instrukcijas ir jų poveikį FLAGS registrai: likusias klausytojas lengvai panagrinės savarankiškai, pavyzdžiui, pagal šią nuorodą:

http://www.electronics.dit.ie/staff/tscarff/8086_instruction_set/8086_instruction_set.html

arba kitą :)

ADD

Formos:

- registras += (registras arba atmintis);
- (registras arba atmintis) += (skaičius);
- akumulatorius += (skaičius)

ADD

Formos:

- registras += (registras arba atmintis);
- (registras arba atmintis) += (skaičius);
- akumulatorius += (skaičius)

Pavyzdys 1. CF požymis

CLC

MOV AX, FFFF

ADD AX, 0001

Pavyzdys 2. ZF požymis

MOV AH, 00

SAHF

MOV AL, 00

ADD AL, 00

Pavyzdys 3. OF ir SF požymiai

MOV AH, 00

SAHF

MOV AL, 7F

ADD AL, 02

Pavyzdys 4. AF požymis

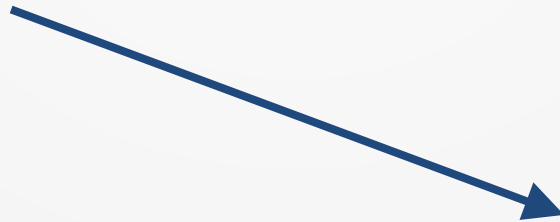
```
MOV AH, 00  
SAHF  
MOV AL, 0E  
ADD AL, 02
```



Ivyko pernešimas iš 3-o bito (skaičiuojant nuo nulio)

Pavyzdys 4. PF požymis

```
MOV AH, 00  
SAHF  
MOV AX, 0111  
ADD AX, 001A
```



Bitų skaičius
BAITO rezultate -
lyginis

Išvada

Net tokia paprasta operacija kaip sudėtis susieta su daugybe požymių :) ...Bet praktikoje dėl to problemų daug nekyla: tiesiog reikia žinoti su kokiais skaičiais (pavyzdžiui, su ženklu ar be ženklo) dirbame ir tinkamai juos apdoroti

ADC

Formos:

- registras += (registras arba atmintis) + CF;
- (registras arba atmintis) += (skaičius) + CF;
- akumulatorius += (skaičius) + CF

Pastabos.

1. CF pridėjimas vyksta automatiškai :)
2. Komanda turi poveikį tiems patiems požymiams kaip ir ADD
3. Gali būti naudojama ilgų (keletas baitų/žodžių) sudėjimui

ADC panaudojimo pvz.

Sudėkime du beženklūs 32 bitų skaičius, kurių pirmas yra registrų poroje BX:AX, o antras - DX:CX:

CLC

MOV AX, FFFF

MOV BX, 1234

MOV CX, 1111

MOV DX, 1234

ADD AX, CX

ADC BX, DX

Rezultatas bus registrų poroje **BX:AX**.

Bendru atveju rezultatas bus [CF]:BX:AX

MUL: skaičių be ženklų daugyba

8-bitų: AL - ką dauginame, o iš ko - atmintyje ar registre. Rezultatas: AX.

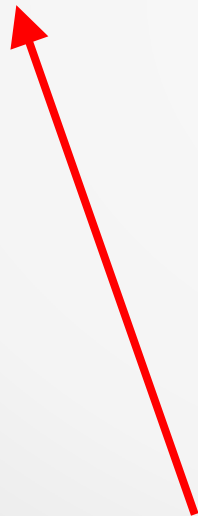
16-bitų: AX dauginamas iš reikšmės registre ar atmintyje, rezultatas bus DX:AX.

Požymiai: CF ir OF (kiti - neapibrėžti).

Kodas: |1111011w|mod100r/m|

MUL: pavyzdys

```
MOV AX, 0004  
MOV BX, 0005  
MUL BX
```



DX:AX bus 0000:0014

IMUL: skaičių su ženklu daugyba

8-bitų: AL - ką dauginame, o iš ko - atmintyje ar registre. Rezultatas: AX.
16-bitų: AX dauginamas iš reikšmės registrę ar atmintyje, rezultatas bus DX:AX.
Požymiai: CF ir OF (kiti - neapibrėžti).
Kodas: $|1111011w|mod101r/m|$
plg. su $|1111011w|mod100r/m|$ (MUL)

IMUL: pavyzdys

```
MOV AL,-1  
MOV AH,-1  
IMUL
```

AX bus 0001



NEG - priešingas skaičius

Keičia operandą į priešingą, t.y.,
operandas := -operandas

Požymiai:

AF, CF, OF, PF, SF ir ZF.

Kodas: |1111011w|mod011r/m|

NEG pavyzdys

```
MOV AX, -100  
MOV CX, AX  
NEG CX  
ADD AX, CX
```



AX bus 0000

AND

- Loginis AND atliekamas su kiekvienu bitu;
- Pvz.:

MOV AX, 000Ah

AND AX, 0003h ; AX bus 02h

Požymiai:
r – priklauso
nuo
rezultatų

[C]	[Z]	[S]	[O]	[P]
0	r	r	0	r

OR

- Loginis OR atliekamas su kiekvienu bitu;
- Pvz.:

MOV AX, 000Ah

OR AX, 0004h ; AX bus 000Bh

Požymiai:
r – priklauso
nuo
rezultatų

[C]	[Z]	[S]	[O]	[P]
0	r	r	0	r

XOR: eXclusive OR

Taisyklė: bitai a ir b duoda operacijoje a xor b vieneta, jeigu ir tik jeigu a nesutampa su b.

Požymiai: CF, OF, PF, SF ir ZF (AF neapibrėžtas).

Kodas (galimi 3 atvejai):

Registras/atmintis su registru:

|001100dw|modregr/m|

Skaičius su AX (AL):

|0011010w|--duom--|duom jeigu w=1|

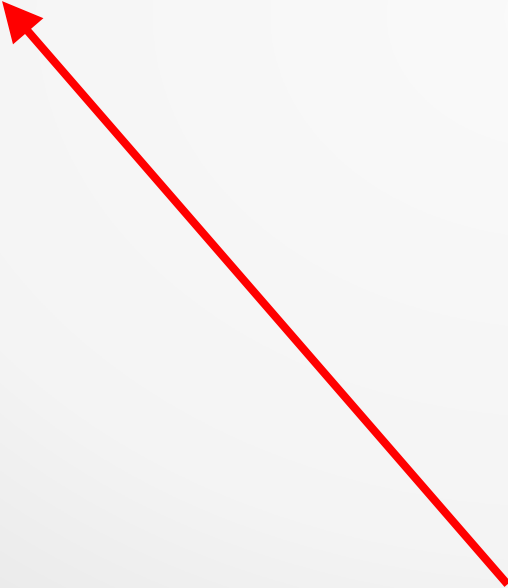
Skaičius su registru/atimintimi:

|1000000w|mod110r/m|--duom--|duom jeigu w=1|

XOR pavyzdys

```
MOV AX, 1234  
XOR AX, AAAAh  
XOR AX, AAAAh
```

AX bus 1234:
pakartotinis XOR
su tuo pačiu
skaičiumi





Pabaiga