Status Flag

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	*X*	*X*	*X*	OF	DF	IF	TF	SF	ZF	*X*	AF	*X*	PF	*X*	CF

Parity Flag (PF) požymis (poz. – 2)

Lyginumo požymis

PF požymis pasako ar rezultato lauko jauniausiam baite yra lyginis kiekis ‘1‘ bitų.

Pavyzdžiai:

`0011 0100` `1111 0000` `0000 0000`
PF = 0 *PF = 1* *PF = 1*

Zero Flag (ZF) požymis (poz. – 6)

Nulio požymis

ZF požymis pasako, ar rezultatų laukas sudarytas vien iš nuliukų. (1, jei taip)
PASTABA: nepasimaukite, jei rezultato laukas nulis, tai ZF **NĖRA nulis**, o vienas.

Pavyzdžiai:
`1111 0000` `0000 0000`
ZF = 0 *ZF = 1*

Overflow tikrinimas, pasiverčiam dvejetainį į jo atitikmenį su ženklu ir žiūrim ar gaunam tokį patį atsakymą kaip ir dešimtainėj sistemoj. Pvz.: 193+191

1100 000 + 1011 1111 = 1000 0000; 1100 0000 = -65; 1011 1111 = -63; -65+(-63) = -128;

Atsakymas 1000 0000 = -128 = -128, todėl overflow nebus

Galima pasiversti į neigiamą arba tiesiog sk be ženklo pasiverčiant į bitus, invertuojant ir tuomet pridedant 1, arba pirmą bitą inti -128 ir tiesiog pridedinėt klo gausim atsakymą. Pvz.: -80

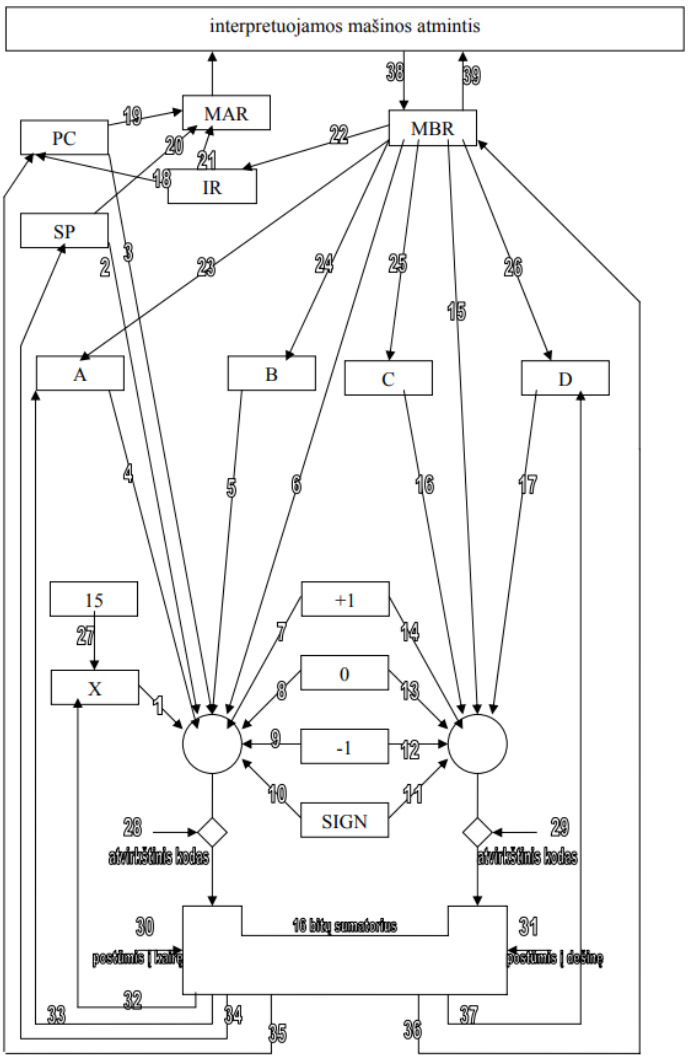
80 = 0101 0000 (invertuojam) = 1010 1111 +1 = 1011 0000

-128+64=-64 > -80, netinka; -128+32=-96, gerai; -96+16 = -80 atsakymas

-128	64	32	16	8	4	2	1
1	0	1	1	0	0	0	0

MPL

Mikroprograminio lygio architektūra



Pavadinimas	*Šešioliktainė reikšmė*	*Dvejetainė reikšmė*	*Dešimtainė reikšmė*	
			Be ženklo	*Su ženklu*
1	0001	0000 0000 0000 0001	+1	+1
0	0000	0000 0000 0000 0000	0	0
-1	FFFF	1111 1111 1111 1111	+65535	-1
SIGN	8000	1000 0000 0000 0000	+32768	-32768
15	000F	0000 0000 0000 1111	+15	+15

Veiksmas	*Šešioliktainė reikšmė*	*Dvejetainė reikšmė*	*Dešimtainė reikšmė*	
			Be ženklo	*Su ženklu*
COM(0)	FFFF	1111 1111 1111 1111	+65535	-1
COM(1)	FFFE	1111 1111 1111 1110	+65534	-2
COM(-1)	0000	0000 0000 0000 0000	0	0
COM(SIGN)	7FFF	0111 1111 1111 1111	+32767	+32767
MBR+COM(MBR)	FFFF	1111 1111 1111 1111	+65535	-1

JMP/CALL

Besąlyginis valdymo perdavimas

Vidinis artimas	1. Iš mašininio kodo po OPK imamas 1 baito poslinkis. 2. Baitas išplečiamas pagal plėtimo pagal ženklą taisyklę. 3. IP:= gautas_rezultatas + IP_reikšmė_komandos_vykdymo_metu (IP+op kodo bitai) *Pvz.: EB 8A*	
Vidinis tiesioginis	1. Iš mašininio kodo po OPK imamas 2 baitų poslinkis. 2. Baitai sukeičiami vietomis. 3. IP:= gautas_rezultatas + IP_reikšmė_komandos_vykdymo_metu (IP+op kodo bitai) *Pvz. E9 8A 5A*	
Išorinis tiesioginis	1. Iš mašininio kodo po OPK imami 4 baitai. 2. Jie priskiriami CS ir IP registrams tokiu eiliškumu: IP j.b., IP v.b, CS j.b., CS v.b. *Pvz.: EA 11 22 33 44. IP=2211, CS=4433.*	
Vidinis netiesioginis	Analizuojamas adresavimo baitas (reg dalis yra OPK plėtinys)	
	Mod=11 → IP registrui priskiriama žodinio registro reikšmė, kurią nurodo r/m.	Mod!=11 → einama į atminties vietą, kurią rodo operandas (mod ir r/m) ir paimami 2 baitai (IP j.b., IP v.b.), kurie tampa nauja IP reikšme.
Išorinis netiesioginis	Analizuojamas adresavimo baitas (reg dalis yra OPK plėtinys)	
	Mod negali būti 11!	Einama į atminties vietą, kurią rodo operandas atmintyje (mod ir r/m) ir paimami 4 baitai eiliškumu IP j.b., IP v.b., CS j.b., CS v.b.

JMP tipas	*Operacijos kodas*
Vidinis artimas	*EB*
Vidinis tiesioginis	*E9*
Išorinis tiesioginis	*EA*
Vidinis netiesioginis	*FF **100****
Išorinis netiesioginis	*FF **101****

CALL tipas	*Operacijos kodas*
Vidinis tiesioginis	*E8*
Išorinis tiesioginis	*9A*
Vidinis netiesioginis	*FF **010****
Išorinis netiesioginis	*FF **011****

RET	Vidinis	Išorinis
Be steko išlyginimo	C3	CB
Su steko išlyginimu	C2 j.b. v.b.	CA j.b. v.b.

- Kviečiant **CALL** komandą, prieš atliekant veiksmus į steką padedamos keičiamų registrų reikšmės:
  - Išorinio atveju PUSH CS, PUSH IP
  - Vidinio atveju PUSH IP
- Kviečiant **RET** komandą (grįžimą iš procedūros), grįžimo adresas imamas iš steko:
  - Išorinio atveju POP IP, POP CS
  - Vidinio atveju POP IP

Trys taisyklės, kurias reik žinoti:

1. Ar buvo panaudotas prefiksas? Jei taip, imam prefiksą atitinkantį segmentą.
2. Ar formuojant EA buvo panaudotas BP? Jei taip, imam SS.
3. Jei formuojant EA nepanaudotas BP → imam DS.

Segmentas	Maš. Kodas	Fokusas
ES	26	Europos Sąjunga
CS	2E	Žaidė CS'ą
SS	36	Sovietų Sąjunga
DS	3E	Dėstė Saikingai (Dėjo Sk**są)

Adresacijos baitas susideda iš:

mod	*reg*	*r/m*
2bit	3bit	3bit

reg laukas gali būti naudojamas ir kaip OPK plėtinys (tuo atveju, kai yra tik vienas operandas r/m)

Ką pasako atskiros adresacijos baito dalys:

mod – modifikatorius

mod	*Prasmė*
00	Po addr. baito eina 0 baitų poslinkis (jokio poslinkio neimama) , išskyrus r/m=110
01	Po addr. baito eina 1 baito poslinkis (jis plečiamas pagal ženklo plėtimo taisyklę iki 2 baitų)
10	Po addr. baito eina 2 baitų poslinkis (pirmiau jaunesnysis baitas, po to vyresnysis baitas)
11	r/m laukas suprantamas, kaip registras, nėra operando atminty

r/m – register or memory, registras arba atmintis.

Reg arba r/m	mod=00	mod=01, 10	mod=11	
			w=0	w=1
000	BX+SI+poslinkis		AL	AX
001	BX+DI+poslinkis		CL	CX
010	BP+SI+poslinkis		DL	DX
011	BP+DI+poslinkis		BL	BX
100	SI+poslinkis		AH	SP
101	DI+poslinkis		CH	BP
110	tiesioginis adresas*	BP+poslinkis	DH	SI
111	BX+poslinkis		BH	DI

***Tiesioginis adresas** – Dviejų baitų poslinkis, be jokių pridėtų registrų.

sreg – segmento registras;

sreg = { 00 – ES; 01 – CS; 10 – SS; 11 – DS;

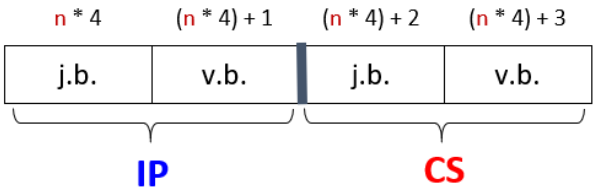
Taip pat esant operandams: registras, registras/atmintis svarbus d bitas (destination):
Jei *d=0*, tai operandai yra: r/m, **reg**
Jei *d=1*, tai operandai yra: **reg**, r/m
Pirmas operandas vadinamas rezultato, antrasis šaltinio, nes jei turint du operandus kažkuriame saugomas veiksmo rezultatas, tai saugomas pirmąjame (rezultato) operande.

Pertraukimai

Pertraukimo paprogramės pradžios radimas

(pertraukimo vektoriaus adresas)

n * 4



Absoliutus apdorojimo procedūros adresas:

CS * 10h + IP

Kviečiant pertraukimą PUSH SF, PUSH CS, PUSH IP (SP=SP-6)

Iš pertraukimo grįžtama IRET: POP IP, POP CS, POP SF (SP=SP+6)

n *reikšmė*	*Pavadinimas*	*Kada įvyksta**
0	*Dalyba iš nulio*	Dalybos perpildymas kilęs vykdant DIV arba IDIV komandą
1	*Žingsninis rėžimas*	Po kiekvienos komandos, jei flagas TF=1
2	*Nemaskuojamas išorinis*	Esant nemaskuojamam išoriniam pertraukimui
3	*Kontrolinis taškas „breakpoint“*	Sutikus komandą maš. kodu CC. Naudojama debugeriuose. Asemblerinė mnemonika *INT 3*
4	*Perpildymo apdorojimas naudojant komandą INTO*	Kai programoje vykdoma komanda INTO ***ir*** flagas OF=1
5-31	*OS reikmėms*	Priklauso nuo operacinės sistemos, įvyksta iškvietus atitinkamą procedūrą
32-255	*Naudotojo reikmėms*	Suprogramuojama kompiuterio vartotojo naudojamos programinės įrangos, įvyksta iškvietus atitinkamą procedūrą

Slankus kablelis

Trumpas realus (**keturiuose** baituose)

S (1b)	*charakteristika (8b)*	*mantisė (23b)*

- 1 ženklo bitas (0 reiškia pliusą, 1 reiškia minusą)
- 8 bitai charakteristika (eilė + 127_{10}) 127_{10}=**7Fh**

  Eilė – dvejeto laipsnis, kai skaičius užrašomas normalizuota forma (dešimtainėje populiariau vadinama standartine skaičiaus išraiška). T.y. sveikas skaičius, rodantis kokiu laipsniu reikia pakelti dvejetą, kad padauginus *1,mantisė* iš $2^{eilė}$ gautume tą patį skaičių.
- 23 bitų mantisė
- Skaičių būtina privesti prie normalizuotos formos, kuri atrodo taip:

$$(-1)^s * 2^{eilė} * 1,\text{mantisė}$$

Ilgas realus (**aštuoniuose** baituose)

S(1b)	*charakteristika (11b)*	*mantisė (52b)*

- 1 ženklo bitas (0 reiškia pliusą, 1 reiškia minusą)
- 11 bitų chrakteristika (eilė + 1023_{10}) 1023_{10} = **3FFh**
- 52 bitų mantisė
- Skaičių būtina privesti prie normalizuotos formos, kuri atrodo taip:

$$(-1)^s * 2^{eilė} * 1,\text{mantisė}$$

Vidinis realus (**dešimtyje** baitų)

S (1b)	*charakteristika (15b)*	*i bitas (1b)*	*mantisė (63b)*

- 1 ženklo bitas (0 reiškia pliusą, 1 reiškia minusą)
- 15 bitų charakteristika (eilė+16383_{10}) 16383_{10} = **3FFFh**
- 1 i bitas, jis parodo koks skaičiukas yra prieš kablelį po kurio eina mantisė (šiame formate nebūtina sudaryti normalizuotos formos, bet patartina tai padaryti ir i bitą visuomet žymėti vienetu)
- 63 bitų mantisė
- Normalizuota forma atrodytų taip: $(-1)^s * 2^{eilė} * 1,\text{mantisė (i bitas=1)}$
- ***Ne***normalizuota forma: $(-1)^s * 2^{eilė} * i,\text{mantisė}$

Ženklo bitas 1 – neigiamas, 0 – teigiamas. Charakteristika = eilė + sk(nuo dydžio 7Fh, 3FFh arba 3FFFh)

Jei daugiau už 1, pasiverčiam į bitus skaičių prieš kablelį ir left shift kol gausim tik 1 sk. prieš kablelį, kiek patraukėm tokia eilė. Visus skaičius likusius po kablelio parašom į mantisės dalį ir toliau dirbam su trupmena kuri buvo pradiniam skaičiuj prieš kablelį.(Beno konspektas 27 psl.)

Jei mažesnis už 1, dauginam iš 2 kol gaunam daugiau už 1. Visu dauginimo rezultatus, tarsi skaičiuojant liekaną normaliam pasirašom ir right shift kol vienas skaičius prieš kablelį, toliau mantisei dirbam su likusia liekana nuo 1. (Pvz.: 0,2*2= 0,4 (0 mintį)

0,4*2=0,8 (0 mintį)

0,8*2=1,6 (baigiam, 1gavom); mūsų skaičius 001, tai per 3 pastumti, kad butu 1,…. Eilė 3)

DAA / DAS (Decimal Adjust for Addition / Subtraction)

```
if ((AL and 0Fh) > 9 or AF == 1) then
      AL := AL + | - 6
      AF := 1
endif
if (AL > 9Fh or CF == 1) then
      AL := AL + | - 60h
      CF := 1
endif
```

Patikrinam, ar antrasis AL skaitmuo yra nebe dešimtainis

Grąžinam AL antrą skaitmenį į dešimtainį pavidalą (jei prie A pridėsim 6 gausim 16 + 0 => 0, jei prie B => 1, etc.)

Patikrinam, ar pirmasis AL skaitmuo yra nebe dešimtainis

Grąžinam pirmąjį AL skaitmenį į dešimtainį pavidalą naudodami tą pačią logiką, kaip ir aukščiau

Pastaba!
Vykdant tarpines sudėties / atimties operacijas NĖRA nustatomos SF reikšmės!

AAA / AAS (Ascii Adjust for Addition / Subtraction)

```
if ((AL and 0Fh) > 9 or AF == 1) then
      AL := AL + | - 6
      AH := AH + | - 1
      AF := 1
      CF := 1
else
      AF := 0
      CF := 0
endif
AL := AL and 0Fh
```

Patikrinam, ar AL yra nebe dešimtainis skaitmuo, jei taip, padidinam/sumažinam AH vienetu, o AL grąžinam į dešimtainę formą

Paliekam tik vieną (dešinįjį) dešimtainį skaitmenį AL

AAM

(Ascii Adjust for Multiplication)

```
AH := AL div 10₁₀
AL := AL mod 10₁₀
```

AAD

(Ascii Adjust for Division)

```
AL := AH * 10₁₀ + AL
AH := 0
```

Kitaip tariant, tai galima aprašyti ir tokia lentele:

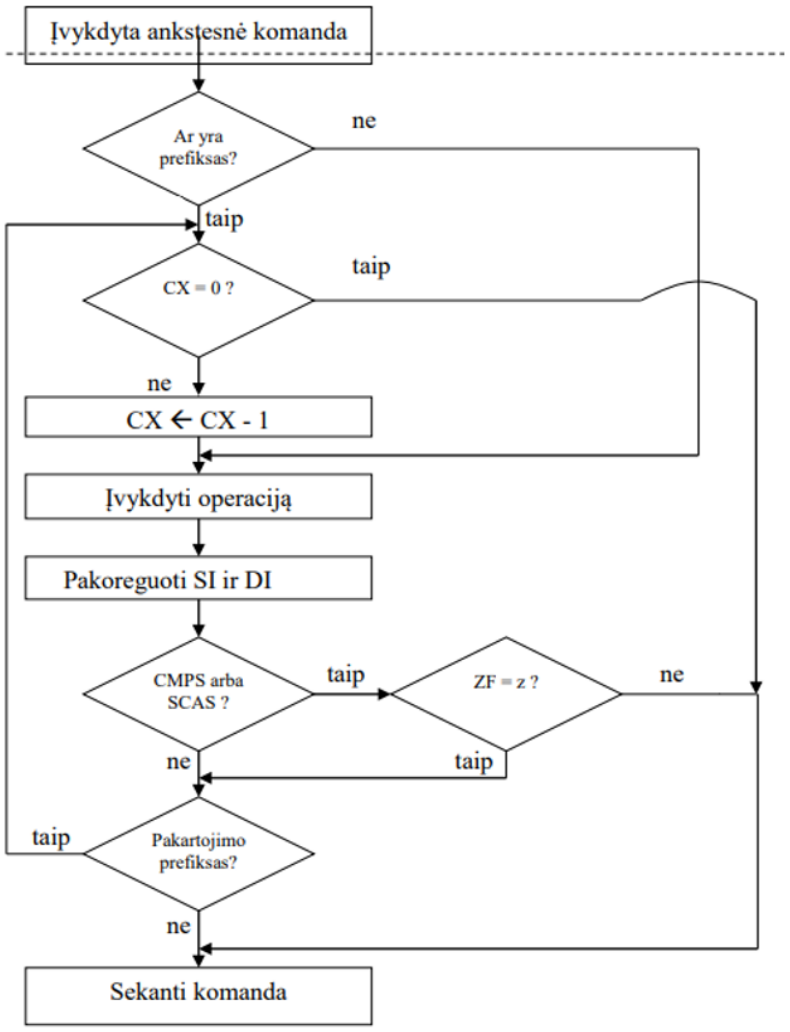
Direction Flagas	*Paskutinė komandos pavadinimo raidė*	*„Delta“ (ką reiktų pridėti prie panaudoto indeksinio registro)*
0	B	+1
0	W	+2
1	B	-1
1	W	-2

Prefikso asemblerinė mnemonika	*Prefikso mašininis kodas*	*z bito reikšmė*
REP REPE REPZ	***F3***	***1***
REPNE REPNZ	***F2***	***0***

Abu atvejai dvejetainėje sistemoje užrašomi tokiu baitu: ***1111 001z***.

Eilutinių komandų sąrašas:

Komandos pavadinimas	*Atliekamas veiksmas*
MOVSB	***mov*** es:[di], ds:[si]
MOVSW	
CMPSB	***cmp*** ds:[si], es:[di]
CMPSW	
SCASB	***cmp*** AL, es:[di]
SCASW	***cmp*** AX, es:[di]
LODSB	***mov*** AL, ds:[si]
LODSW	***mov*** AX, ds:[si]
STOSB	***mov*** es:[di], AL
STOSW	***mov*** es:[di], AX



DEBUG

Rašyt komandas: a (rašys į IP pozicija), arba a100 (rašys į atmintį nuo 100 pozicijos).

Parašyti mašininį kodą: a100->db (kodo bitai) (pvz: db 9A 26 15)

Pasižiūrėti parašytas komandas: u100, arba u.

Įvykdyti komandas t(įvykdo 1), t? (įvykdo ? komandų, pvz.: t3, įvykdys 3 komandas)

Išsispausdinti registrus r.

Flags:

Flag Name	Set	Clear
Overflow(yes/no)	OV	NV
Direction(increment/decrement)	DN	UP
Interrupt(enable/disable)	EI	DI
Sign(negative/positive)	NG	PL
Zero(yes/no)	ZR	NZ
Auxiliary carry(yes/no)	AC	NA
Parity(even/odd)	PE	PO
Carry(yes/no)	CY	NC

Paruošė: Dominykas Višnevskis