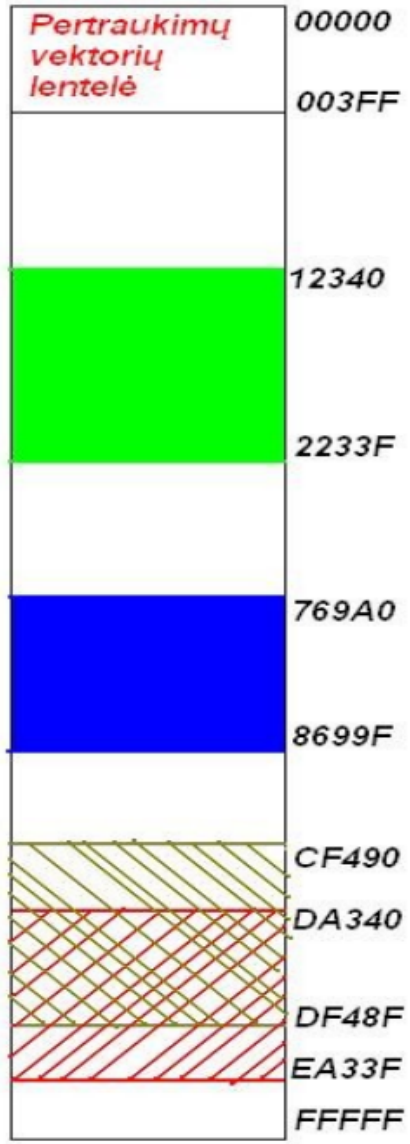
Atmintis ir jos sandara

- Mūsų nagrinėjamoje architektūroje atminties dydis yra 2^{20} baitų (1 MB)
- Paragrafas – 16 baitų (10h) dydžio blokas.
- Atmintis yra skirstoma į segmentus:
  - CS – Code Segment
  - DS – Data Segment
  - ES – Extra Segment (extra data segment / papildomas duomenų segmentas)
  - SS – Stack Segment
- Segmento dydis: 10000h (arba 64K)
- Ir atmintis, ir segmentai yra **cikliniai:**
  - FFFFF+1=00000h
  - FF8A+AB=0035h



- Segmentiniai registrai yra 16 bitų reikšmėse. Pagal pateiktą paveikslėlį (*gali būti kitaip!)* segmentinių registrų reikšmės yra:
  - CS = 1234h
  - DS = 769Ah
  - ES = CF49h
  - SS = DA34h
- Segmentinis registras nurodo segmento pradžią atmintyje, kuri yra adresu: segmento registro reikšmė * paragrafo dydis (10h)
  - Pvz.: DS pradžia atmintyje yra 769A*10=769A0h

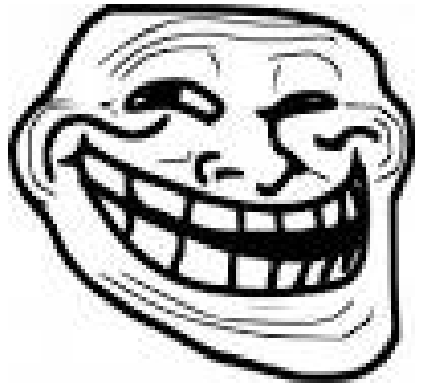
Efektyvus ir absoliutus adresas

- Efektyvus adresas (EA): 4 šešioliktainiai skaitmenys (pvz. ACDCh). Jis parodo poslinkį nuo segmento pradžios.
- Absoliutus adresas (AA): 5 šešioliktainiai skaitmenys (pvz. BABA5h). Jis parodo elemento adresą atmintyje.

AA = segmentinio_registro_reikšmė * 10h + EA

(dar užrašomas seg:EA, pvz.: DS:EA)

Adresavime svarbūs registrai



- SI – Source Index. Dažniausiai naudojamas su DS (DS:SI)
- DI – Destination Index. Dažniausiai naudojamas su ES (ES:DI)
- IP – Instruction Pointer. **Komandos vykdymo metu rodo į sekančios vykdymos komandos pradžią.** Rodo poslinkį nuo kodo segmento pradžios, dažniausiai naudojamas su CS (CS:IP)
- BP – Base Pointer. Dažniausiai naudojamas kartu su SS (SS:BP)
- SP – Stack Pointer. Rodo į steko viršūnė, dažniausiai naudojamas kartu su SS (SS:SP)

Uždavinukas: Registrų reikšmės: SS=ABCD, SP=0005, BP=1456, SI=BOOB, DI=7536. Duotame kode 9A yra operacijos kodas, EB yra automatizacijos plėtinys, FE yra mikrokomandos GOGO reikšmė. Vykdomoji komanda kode yra sudaryta iš 5 baitų. Kokia bus IP reikšmė komandos vykdymo metu?

1234 9A EBFE6789453124 CALL TEXT (1234 yra poslinkis kodo segmente)

Ats: 1234+5=1239h

Plėtimo pagal ženklą taisyklė

- Plėsti pagal ženklą reikia tuomet, kai prie žodžio (2 baitų) reikia pridėti 1 baitą.
- Plečiama **pagal ženklo bitą** (patį kairiausią bitą). Koks ženklo bitas, tokį bitą reikia prirašyti iš kairės 8 kartus.
  - 0011 0101 -> 0000 0000 0011 0101 (35h -> 0035h)
  - 1000 0001 -> 1111 1111 1000 0001 (81h -> FF81h)

1. IP registro reikšmė yra 2222h, poslinkis 7Fh. Kokia bus nauja IP reikšmė?

Ats.: 22A1h

2. IP registro reikšmė yra 4587h, poslinkis 87h. Kokia bus nauja IP reikšmė?

Ats.: 450Eh

Adresavimo būdai

- Tiesioginė adresacija:
  - Pagal adresavimo baitą
  - Pagal operacijos kodą:
    - Paprasta
    - Santykinė
    - Absoliuti
  - Netiesioginė adresacija

Adresavimo būdai

- Pagal adresavimo baitą:

  MOV AX, [bx+55] -> 8B 47 55 (*47 yra adresavimo baitas*)

- Paprasta tiesioginė adresacija:

  MOV AX, word ptr wild_stuff -> A1 05 00 (*0005 yra operando EA*)

- Santykinė tiesioginė adresacija:

  JE wild_place-> 74 F8 (*F8 yra poslinkis nuo esamos pozicijos*)

- Absoliuti tiesioginė adresacija:

  JMP 5511h:8722h -> EA 22 87 11 55 (*komandoje tiesiogiai nurodytas EA (8722) ir segmento registro reikšmė (5511)*)

- Netiesioginė adresacija:

  JMP [bx+si] -> FF 20 (*pagal adresavimo baitą suformuojama AA;* ***tuo adresu esanti reikšmė*** *laikoma operando adresu*)

Adresavimo baitas

Adresavimo (adresacijos) baitas eina po operacijos kodo (OPK). Adresavimo baito sandara:

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

Pvz.: Adresacijos baitas yra 8A.
8Ah=10|001|010
mod reg r/m

Mod nusako, koks yra poslinkis:

- 00 – 0 baitų poslinkis (išimtis, kai r/m = 110)
- 01 – 1 baito poslinkis (reiks plėst pagal ženklą!)
- 10 – 2 baitų poslinkis
- 11 – r/m laukas imamas kaip registras (t.y., nėra operando atminty)

Smagioji lentelė

Lentelę reikia mokėti mintinai!

w (word/width) – bitas, kuris nurodo, ar operuojama baitais, ar žodžiais. Šis bitas būna operacijos kodo viduje.

- w=0 (word yra FALSE) → operuojama baitais
- w=1 (word yra TRUE) → operuojama žodžiais

d (destination) – bitas OPK viduje, kuris nurodo kryptį (kas yra operandas-šaltinis, o kas yra operandas-rezultatas)

- d=0 → šaltinis = reg, rezultatas = r/m
- d=1 → šaltinis = r/m, rezultatas = reg

Operacijos kodas *Adresavimo būdas*

Pvz.: 1000 10dw mod reg r/m [poslinkis] – MOV registras ←→ registras/atmintis

FA nustatom pagal r/m!

Naudinga:
uosis.mif.vu.lt/~julius/Tools/asm/KomKodaiViso.pdf

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.

Pavyzdinė užduotis (1)

CS=1111, ES=5342, SS=6987, DS = 2345, BX=1111. Vykdoma komanda 8B. Duotas mašininis kodas: 8B 8F 12 34. Koks operando atmintyje EA ir AA?

- **8B – operacijos kodas**. Išskleidžiam: 1000 1011
  - d=1→šaltinis r/m, rezultatas reg
  - w=1→operuojama žodžiais
- **8F – adresavimo baitas.**

Išskleidžiam: 10 001 111

- mod=10 → poslinkis 2 baitų
- reg = 001, operuojam žodžiais (nes w=1) → komandoje naudotas registras yra CX
- r/m = 111 → BX + poslinkis

Kadangi mod=10, poslinkis yra 2 baitų, imam 2 baitus po adresavimo baito kaip poslinkį:

- 12 34 → poslinkis bus 34 12h

Svarbu: mašininiam kode žodis yra užrašomas tokia tvarka: *j.b v.b.,* tačiau assembler'yje (o ir atliekant veiksmus) tvarka yra *v.b. j.b.,* todėl reikia nepamiršt baitų apkeisti vietomis!

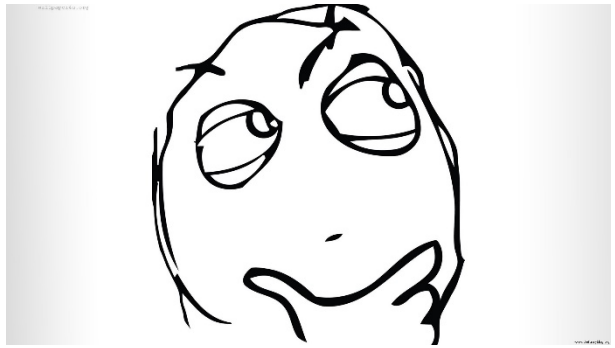
Pavyzdinė užduotis (2)

Koks bus EA?

- Iš adresavimo baito išsiaiškinom, kad EA formuosim taip: BX+ poslinkis → 1111+3412=4523h
- EA = 4523h

Kaip gaut AA?

- Žinom formulę: AA=seg*10h+EA.
- EA=4523, o koks čia segmentas?



Reik dar teorijos!

Segmento nustatymas adresavime

Prefiksas – tai prieš OPK mašininiame kode einantis baitas, kuris gali pakeisti naudojamą segmentą. *Pvz.:* 26 8B 8F 12 34

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)

Pavyzdinė užduotis (3)

Tikrinam pagal taisykles:

- Ar buvo prefiksas? Ne, nes maš. kodas buvo 8B 8F 12 34.
- Ar naudojom BP? Ne, nes EA sudarėm pagal BX + poslinkis.
- Reiškia, segmentas bus DS.

AA=DS*10h+EA=2345*10+4523=27973h

Ats.: AA = 27973h, EA = 4523h

Būtinai parašykit atsakymą!

Uždaviniai

1. Visų registrų reikšmės yra FFFF. Kokia komanda vykdoma ir koks operando atmintyje EA ir AA, jei komandos mašininis kodas yra: 8A F1 55 88.
2. AX=0000, BX=1111, CX=2222, DX=3333, SI=4444, DI=5555, BP=6666, SP=7777, IP=8888, CS=1234, DS=2345, SS=FFEE, ES=4567. Koks operando atmintyje EA ir AA, jei komandos mašininis kodas yra 2E 8C 1E 28 37.
3. Registrų reikšmės yra: DS=21FE, SS=5634, CS=31CC, ES=41E3, BP=9A32, BX= 7536, SI=45FA, DI=22F1. Apskaičiuoti operando efektyvų adresą pagal adresavimo baitą 82.
   Po adresavimo baito seka baitai: FE01
4. DS=FE21, SS=3456, CS=CC31, ES=E341, BP=329A, BX=3675, SI=FA45, DI=F122. Apskaičiuoti operando absoliutų adresą pagal adresavimo baitą 6E. Po adresavimo baito seka baitai: ABBA.

Daugiau uždavinių galite rasti Beno konspekte arba dėstytojo J. Andrikonio skaidrėse.

Stekas

- PUSH – 2 baitai (žodinis registras/nurodyta atminties vieta) yra įdedami į steką, prieš tai SP sumažinus 2 vienetais.
  1. SP:=SP-2;
  2. MOV SS:SP, jaunesnysis baitas (arba pirmas baitas iš atminties)
     MOV SS:(SP+1), vyresnysis baitas (arba sekantis baitas iš atminties)
- POP – 2 baitai iš steko yra patalpinami į nurodytą žodinį registrą/atminties vietą, po to SP padidinamas 2 vienetais.
  1. MOV jaunesnysis baitas, SS:SP
     MOV vyresnysis baitas, SS:(SP+1)
  2. SP:=SP+2;

Pavyzdinis uždavinys su steku (1)

AX=9876, SS=4567, SP=1973. Vykdoma komanda PUSH AX. Kokia baito, kurio absoliutus adresas atmintyje yra 46FE2h, reikšmė?

- Reikės ieškoti baito reikšmės atmintyje → pasibraižom atmintį.
- Vykdom komandą PUSH AX:
  1. SP:=1973-2=1971.
  2. MOV SS:SP, 76h (*į atminties vietą, kurios AA yra 4567:1971, patalpinom baitą, kurio reikšmė 76h*)
     MOV SS:(SP+1), 98h (*į atminties vietą, kurios AA yra 4567:1972 patalpinom baitą, kurio reikšmė yra 98h*)
- Įvykdėm PUSH - atsinaujinam atminties lentelę.

SP	Baito reikšmė
1969	XX
1970	XX
1971	XX
1972	XX
1973	XX

Atminties dalis prieš PUSH

SP	Baito reikšmė
1969	XX
1970	XX
1971	76
1972	98
1973	XX

Atminties dalis po PUSH

Pavyzdinis uždavinys su steku (2)

- Atminties vieta, kuri yra duota uždaviny: 47642h
- Reik patikrinti, galbūt mūsų turima atminties vieta yra ta pati, kurios ir reikia:
  1. AA=seg_reg_reikšmė*10h+EA
  2. Turim steką dalį, vadinasi, galim rasti tos dalies absoliučius adresus: SS*10h+SP (imam dabartinį SP, kuris yra 1971)
  3. 4567*10+1971 = 46FE1h
- Atsinaujinam lentelę (užsirašom ir absoliučius adresus)
- Pastebim, kad baito, kurio absoliutus adresas atmintyje yra 46FE2, reikšmė yra 98h

Absoliutus adresas	**SP**	**Baito reikšmė**
46FDF	1969	XX
46FE0	1970	XX
46FE1	1971	76
46FE2	1972	98
46FE3	1973	XX

Atmintis su absoliučiais adresais

<u>Ats.: 98h</u>

Valdymo perdavimo komandos

Valdymo perdavimas – toliau vykdomas ne kodas, einantis atmintyje po einamosios komandos, o iš bet kurios nurodytos atminties vietos.
Valdymo perdavimo komandos:

1. **Sąlyginės**. Valdymas bus perduotas tik tuomet, jei bus patenkinta tam tikra sąlyga (*pvz.: JE narnia, LOOP hogwarts*).
2. **Besąlyginės**. Valdymas bus perduotas į nurodytą vietą netikrindamas jokių sąlygų (*pvz.: JMP hell*).

Besąlyginis valdymo perdavimas (1)

Komandos: JMP, CALL, RET, INT, IRET

Besąlyginis valdymo perdavimas gali būti: vidinis/išorinis + tiesioginis/netiesioginis/artimas.

- **Vidinis:** valdymas yra perduodamas segmento viduje (*keičiasi tik IP reikšmė*)
- **Išorinis:** valdymas yra perduodamas visos atminties ribose (*keičiasi ir IP, ir CS reikšmės*)

Besąlyginis valdymo perdavimas (2)

- **Tiesioginis** – nauja IP reikšmė (arba nauja IP ir CS reikšmės) imama **tiesiogiai iš vykdomo kodo**, t.y., jos yra tam tikri baitai po komandos OPK. *Pvz.: E9 F9 FF → IP:=IP_komandos_vykdymo_metu + FFF9*
- **Netiesioginis –** nauja IP reikšmė (arba nauja IP ir CS reikšmės) **randama naudojant adresavimo baitą**. Operandas atmintyje parodo, iš kur reikės pasiimti naują IP (arba CS ir IP) reikšmę. Jei mod=11, tai IP = žodinio registro (kurį nurodo r/m) reikšmė. Pastaba: netiesioginio valdymo perdavimo atveju adresavimo baito reg dalis yra OPK plėtinys. *Pvz.: FF 68 FD → analizuojam adresavimo baitą 68 ir naują IP pasiimam iš atminties vietos, kurią nurodė mod + r/m kombinacija*
- **Artimas –** tiesioginio tipas, kai valdymas perduodamas mažu atstumu [-128;127 baitai], todėl poslinkis užrašomas vienam baite. Pastaba: negalimas išorinis artimas atvejis. *Pvz.: EB EC → IP:=IP_komandos_vykdymo_metu+FFEC*

Besąlyginis valdymo perdavimas (3)

<table>
<tr><td>Vidinis artimas</td><td colspan="2">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</td></tr>
<tr><td>Vidinis tiesioginis</td><td colspan="2">1. Iš mašininio kodo po OPK imamas 2 baitų poslinkis.
2. Baitai sukeičiami vietomis.
3. IP:= gautas_rezultatas + IP_reikšmė_komandos_vykdymo_metu</td></tr>
<tr><td>Išorinis tiesioginis</td><td colspan="2">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.</td></tr>
<tr><td rowspan="2">Vidinis netiesioginis</td><td colspan="2">Analizuojamas adresavimo baitas (reg dalis yra OPK plėtinys)</td></tr>
<tr><td>Mod=11 → IP registrui priskiriama žodinio registro reikšmė, kurią nurodo r/m.</td><td>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.</td></tr>
<tr><td rowspan="2">Išorinis netiesioginis</td><td colspan="2">Analizuojamas adresavimo baitas (reg dalis yra OPK plėtinys)</td></tr>
<tr><td>Mod negali būti 11!</td><td>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.</td></tr>
</table>

Sąlyginis valdymo perdavimas (1)

Visi atvejai (LOOP, sąlyginiai JMP) išskyrus INTO: komanda užima 2 baitus. 1-as baitas – OPK, 2-as baitas – poslinkis.

- INTO – vykdo INT 4, jeigu OF = 1 (OPK = CE, visa komanda – 1 baitas).
  1. Ar OF = 1?
  2. Taip → vykdoma INT 4.
     Ne → vykdomas sekanti komanda, kode esanti po INTO
- LOOP atvejai:
  1. CX:=CX-1 (VISADA);
  2. Ar tenkinama sąlyga?
  3. Taip → šokama su vieno baito poslinkiu (IP:=IP_komandos_vykdymo_metu + poslinkis (išplėstas pagal ženklą))
     Ne → vykdoma sekanti komanda, kode einanti po nagrinėto LOOP atvejo

Sąlyginis valdymo perdavimas (2)

- Sąlyginiai JMP atvejai (JO, JNO, JAE, JZ, JBE ir t.t.):
  1. Ar tenkinama sąlyga?
  2. Taip → šokama su vieno baito poslinkiu (IP:=IP_komandos_vykdymo_metu + poslinkis (išplėstas pagal ženklą))
     Ne → vykdoma sekanti komanda, kode einanti po sąlyginio JMP

Komandų operacijų kodai

Reikia mokėti:

1. JMP
2. CALL
3. RET
4. IRET

Nebūtina mokėti:

1. Sąlyginiai JMP
2. LOOP atvejai
3. INT atvejai

Operacijų kodus galima rasti dėstytojo konspekte, Beno konspekte bei dėstytojo Andrikonio sudarytame sąraše.

Pavyzdinis uždavinys (1)

Registrų reikšmės yra: DS=21FE, SS=5634, CS=0ADF, ES=41E3, BP=9A32, BX=7100, SI=0011, DI=22F1. Koks bus procedūros išorinio iškvietimo absoliutus adresas:

9715 2E FF 9F 16 26 call cs: number (9715– poslinkis kodo segmente)

- Tikslas – rasti procedūros iškvietimo absoliutų adresą
- Sąlygoj duota, kad vyksta CALL. Kadangi OPK – FF ir plėtinys yra 011, tai išorinis netiesioginis CALL.
- Vadinasi, vyksta besąlyginis **išorinis netiesioginis** valdymo perdavimas – absoliutus adresas suformuojamas paėmus 4 baitus iš atminties.

Pavyzdinis uždavinys (2)

1. Analizuojam adresavimo baitą. 9F = 10 011 111.
2. Mod = 10 → poslinkis 2 baitų.
3. R/m=111 → operando efektyvus adresas formuojamas taip: BX + 2 baitų poslinkis.
4. EA= 7100 + 2616 = 9716h

- Naudotas prefiksas 2E → vadinasi, operando atmintyje absoliutaus adreso formavimui naudojamas CS.
- Gavome, kad iš atminties vietos CS:9716 reikia paimti 4 baitus ir formuoti naujus IP ir CS (nes tai **išorinis netiesioginis** atvejis)
- Kokį atminties gabalą mes turim?

Pavyzdinis uždavinys (3)

- Prisimenam sąlygą:
  9715 2E FF 9F 15 26 call cs: number (9715– poslinkis kodo segmente)
- Vadinasi, duotas atminties gabalas iš kodo segmento su poslinkiu 9715. Reiškia, mūsų ieškomi baitai iš atminties su adresu CS:9716 yra matomi sąlygoje.
- Imam 4 baitus nuo CS:9716 reikiama tvarka:
  FF – IP j.b.
  9F – IP v.b.
  15 – CS j.b.
  26 – CS v.b.
- Turim, kad CS=2615, IP=9FFF. Formuojam absoliutų adresą pagal formulę AA=seg_reg_reikšmė * 10h + EA.
- Procedūros iškvietimas yra formuojamas pagal CS ir IP, vadinasi, AA=CS*10h+IP
- AA = 2615*10+9FFF = 3014F.

Ats.: 3014Fh.

Uždaviniai

1. AL=03, BL=02, CL=00, DL=01, AH=00, BH=01, CH=02, DH=03, ES=0000, CS=ABCD, SS=1234, DS=FE21, SP=2222, SF=0000. Įvykdžius nurodytą komandą, apskaičiuoti registrų reikšmių sumą: AL + BL + CL + DL + IP
   0100 E2 90 90 LOOP… (0100 yra poslinkis kodo segmente)
2. AX=0003, BX=0002, CX=0000, DX=0001. Įvykdžius nurodytą komandą, apskaičiuoti sekančios vykdomos komandos efektyvų adresą.
   FFFE EB FE JMP number (FFFE yra poslinkis kodo segmente)
3. DS=1111, ES=2222, CS=FFFF, SS=4444. Koks bus sekančios komandos absoliutus adresas, jei ta komanda bus vykdoma kodo segmente.
   0F01 E8 FF 00 CALL mom (0F01 yra poslinkis kodo segmente)
4. AL=01, BL=02, CL=03, DL=04, AH=52, BH=63, CH=74, DH=FF. Koks bus sekančios komandos efektyvus adresas, jei vykdoma komanda:
   FAAF FF E2 AA BB JMP places (FAAF yra poslinkis kodo segmente)

Uždavinių atsakymai

1. 197h
2. FFFEh
3. 00FF3h
4. FF04h

Interrupt‘ai pagal kilmę yra skirstomi į **vidinius** ir **išorinius**.

Vidiniai interrupt‘ai iš pačio procesoriaus (arba jie išreikštai kviečiami programinio kodo arba neišreikštai, iškilus dėmesio reikalaujančiai situacijai). J ie turi **didžiausią** vykdymo prioritetą.

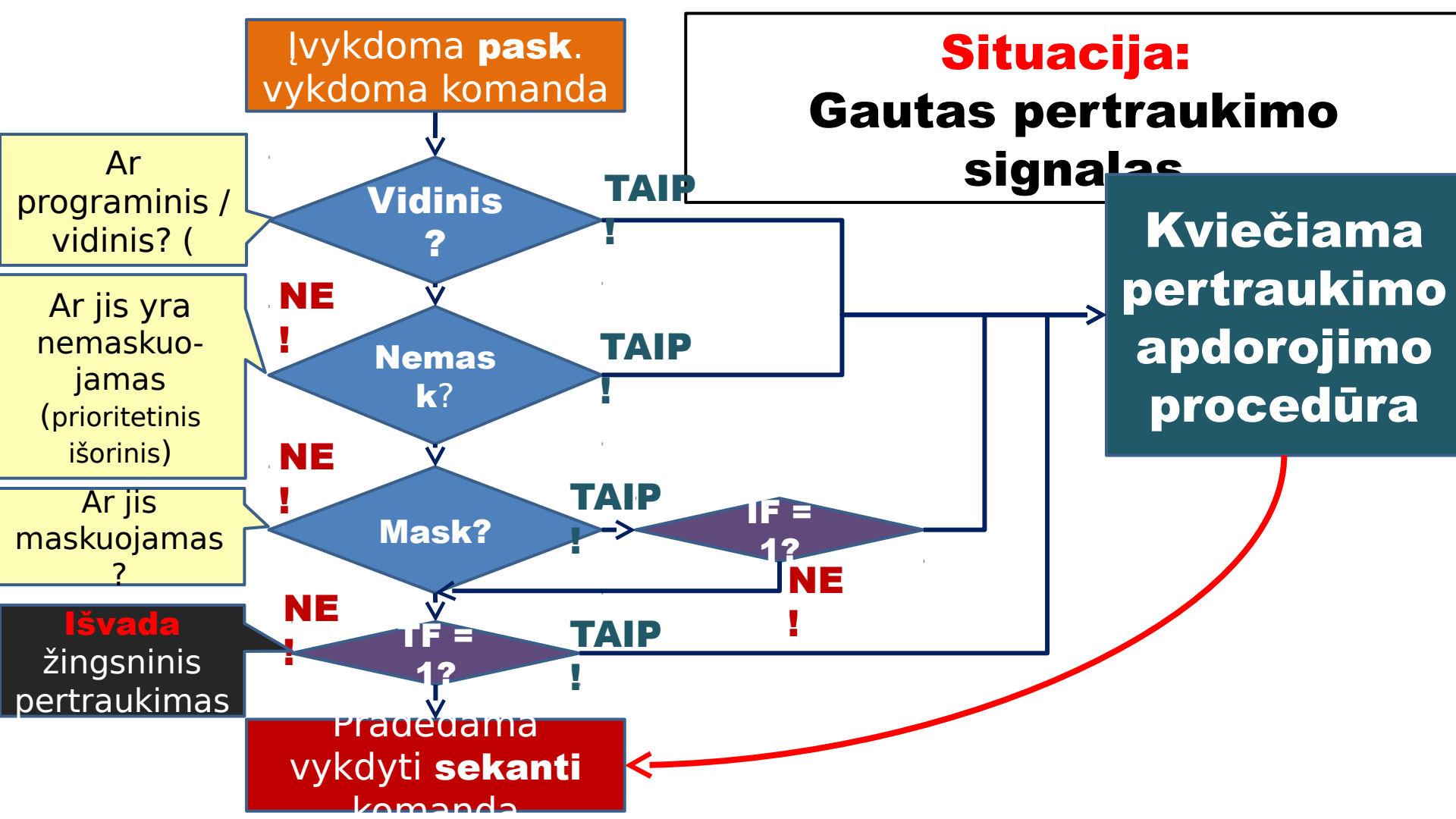
Išoriniai interrupt‘ai kyla dėl išorinių priežasčių. Kadangi ne visada logiška yra reaguoti į absoliučiai visus išorinių įrenginių pertraukimus, **išoriniai** interrupt‘ai yra papildomai skirstomi į **Maskuojamus** ir **Nemaskuojamus**.

Maskuojami pertraukimai gali būti ignoruojami, nustačius $Interrupt\ žymę\ (IF) = 0$.

Interrupt‘ai skirstomi tam, kad, jei kiltų keli interrupt‘ai vienu metu, būtų galima juos vykdyti tam tikra, logiška ir aprašyta, tvarka.

Vykdymo prioritetai:

1. Dalyba iš nulio
2. Programinis
3. INTO (overflow)
4. Nemask išoriniai
5. Mask išoriniai
6. Žingsninis (debug)


Įvykdoma pask. vykdoma komanda
Situacija:
Gautas pertraukimo signalas
Ar programinis / vidinis? (
Vidinis ?
TAIP
!
Kviečiama pertraukimo apdorojimo procedūra
NE
!
Ar jis yra nemaskuo-jamas (prioritetinis išorinis)
Nemask?
TAIP
!
NE
!
Ar jis maskuojamas ?
Mask?
TAIP
!
IF = 1?
NE
!
Išvada žingsninis pertraukimas
NE
!
IF = 1?
TAIP
!
Pradedama vykdyti sekanti komanda

Pertraukimo procedūrų adresavimas

Kadangi į skirtingus pertraukimus reikia reaguoti skirtingai, skirtingų pertraukimų apdorojimo paprogramių (procedūrų) yra labai daug. Tam, kad būtų užtikrintas išplėčiamumas ir suderinamumas ateičiai, pagal susitarimą procesoriai laiko visų pertraukimų adresus pirmajame atminties kilobaite.

Kiekvienam pertraukimui (kurių gali būti 256) yra skiriami 4 baitai, vadinami **pertraukimo vektoriumi**, kurie atitinka to konkretaus pertraukimo adresą atmintyje.

Pertraukimo vektoriaus keturi baitai atitinka pertraukimo procedūros **IP_J, IP_V, CS_J, CS_V**

Pertraukimo vektoriaus adresas apskaičiuojamas pagal formulę **n*4**, kur **n** yra pertraukimo numeris.

Pertraukimo procedūrų adresavimo pvz.

Pertraukimo **INT 3h** vektoriaus **absoliutus adresas** yra **0000Ch**.
baitai adresu atitinka procedūros:
0000Ch: $\mathbf{IP_J}$, 0000Dh: $\mathbf{IP_V}$, 0000Eh: $\mathbf{CS_J}$, 0000Fh: $\mathbf{CS_V}$

Jei norime rasti pertraukimo procedūros absoliutųjį adresą, tereikia nueiti į pertraukimo vektorių, pasiimti keturis baitus ir iš jų „sulipdyti“ absoliutų adresą.

Jeigu atmintis būtų tokia, tai:

AA=00000: 10 11 12 13 14 15 16 17 18 19 20 21 **22 23 24 25**
pertraukimo procedūros INT 3h: IP_J IP_V CS_J CS_V

CS = 2524h **IP** = 2322h **AA** = **CS***10h + **IP**
pertraukimo procedūros AA = 27562

Sąvokų žodynėlis

- ***Pertraukimo vektorius (vektoriaus adresas)*** – adresas, nuo kurio keturiuose baituose aprašytas pertraukimo apdorojimo paprogramės adresas (Ipj, Ipv, Csj, Csv).
- ***Pertraukimo absoliutus adresas*** – pertraukimo aprodorojimo paprogramės absoliutus adresas
- ***Pertraukimo numeris*** – pertraukimo numeris (skaičius rašomas po INT arba mašininiame kode po komandinio baito CD)
  pvz.: CD 05 reiškia, kad bus įvykdytas **INT** 5h

Pertraukimo numeriai, kuriuos reikia mintinai mokėt:

- **INT** 0h = Dalybos iš nulio pertraukimas (CD 00)
- **INT** 1h = Žingsninio režimo pertraukimas (CD 01)
- **INT** 3h = Kontrolinio taško pertraukimas (CD 03) ARBA (CC)
- **INT** 4h = Perpildymo pertraukimas (CD 04) ARBA (CE)

Interrupto kvietimo mechanizmas

Pertraukimo kvietimas (INT n):

1. Pagal interrupt'o numerį (n) iš vektorių lentelės apskaičiuojamas vektoriaus absoliutus adresas
2. Į steką paeiliui padedamos SF, CS, IP registrų reikšmės
3. TF ir IF Status flag'o žymės nunulinamos
4. Perduodamas valdymas

1. **Adresas $n*4$:**
   IP_J, IP_V, CS_J, CS
2. **PUSH SF**
3. **PUSH CS**
4. **PUSH IP**
   (SP – 6)
5. **TF = 0, IF = 0**

Pertraukimo paprogramėje vyksta apdorojimas

Grįžimas iš pertraukimo (IRET):

1. Iš steko paeiliui paima IP, CS, SF (atkreipkite dėmesį, eiliškumas atvirškčias dėl steko realizacijos)
2. Grąžinamas valdymas

1. **POP IP**
2. **POP CS**
3. **POP SF**
   (SP + 6)

Pavyzdinė užduotis (1)

Duotos registrų reikšmės:

DS = FE21, SS = 5634, CS = C131, ES = 3EE3, SF = 05FF,
BP = 92A2, BX = C5D6, SI = 45FA, DI = 22F1, SP = FFE4

Duotas atminties fragmentas:

AA=00000: 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
AA=00010: 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
AA=00020: 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

Užduotis: Apskaičiuoti pertraukimo INT 5h vektoriaus absoliutųjį adresą.

Sprendimas (1)

- Gerai paskaitom sąlygą ir pažiūrim, ko mūsų prašo, mūsų prašo pertraukimo **VEKTORIAUS (!)** absoliutaus adreso.
- Vektoriaus absoliutus adresas skaičiuojamas **n*4**, kur **n** yra pertraukimo numeris.
- Kadangi pertraukimo numeris yra 5h, tai padauginam (naudodami sudėtį).

```
  5h   Ah
+ 5h + Ah
  Ah   14h
```

- Gautą atsakymą **14h** išplečiam iki 5 skaitmenų (kadangi absoliutaus adreso laukas susidaro iš 5 skaitmenų) ir gauname **00014h**, kas ir bus mūsų vektoriaus absoliutus adresas.

Ats: **00014h**

Pavyzdinė užduotis (2)

Duotos registrų reikšmės:

DS = FE21, SS = 5634, CS = C131, ES = 3EE3, SF = 05FF,
BP = 92A2, BX = C5D6, SI = 45FA, DI = 22F1, SP = FFE4

Duotas atminties fragmentas:

AA=00000: 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
AA=00010: 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
AA=00020: 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

Užduotis: Apskaičiuoti pertraukimo INT 5h apdorojimo procedūros absoliutųjį adresą.

Sprendimas (2)

- Reikia apskaičiuoti pertraukimo apdorojimo procedūros absoliutųjį adresą. Mes žinome, kad jis yra laikomas pertraukimo vektoriuje, keturiuose baituose pavidalu:
  IP_J, IP_V, CS_J, CS_V
- Kadangi INT 5h vektoriaus absoliutus adresas **00014h** (kaip gauti aprašyta praeitame sprendime), mums tereikia atmintyje tuo adresu surasti mus dominančius keturis baitus.

```
AA=00000: 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
AA=00010: 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
                      IPJ IPV CSJ CSV
```

- Susilipdome **CS** ir **IP**: **CS** = 3332h, **IP** = 3130h ir pasirašom absoliutųjį adresą pagal formulę: **AA** = **CS***10h + **IP**. **Ats**: **36450h**

Pavyzdinė užduotis (3)

Duotos registrų reikšmės:

DS = FE21, SS = 5634, CS = C131, ES = 3EE3, SF = 05FF,
BP = 92A2, BX = C5D6, SI = 45FA, DI = 22F1, SP = FFE4

Duotas atminties fragmentas:

AA=00000: 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
AA=00010: 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
AA=00020: 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

Užduotis: Kokia bus registrų SF ir SP reikšmių suma įvykdžius pertraukimo komandą INT 7h?

Sprendimas (3)

- Kadangi mums reikia apskaičiuoti SP ir SF reikšmių sumą įvykdžius pertraukimo komandą, mums reikia prisimint, kaip keičiasi jie keičiasi.
- Kviečiant pertraukimą į steką padedamos (PUSH) trys reikšmės. Tai reiškia, kad SP reikšmė mažės 6 (2 už kiekvieną PUSH).
  Duotas **SP** = FFE4, FFE4 – 6 = **FFDE** **naujas SP = FFDE**

- Kadangi IF ir TF reikšmės yra nunulinamos, tai SF‘o reikšmė keisis.
  Duotas **SF** = 05FF, išsirašom dvejetaine: 0000 0101 1111 1111
  XXXX ODIT SZXA XPXC
  Pakeičiam IF ir TF į 0 ir perrašom: 0000 0100 1111 1111
  Paverčiam į šešioliktainę: **naujas SF = 04FF**
- Sudedam **SF** ir **SP**: FFDE + 04FF = 104DD (sumos atsakymui nėra taikomi lauko apribojimai, t.y. rašom tokį atsakymą, kokį gaunam) **Ats**: **104DDh**

Užduotys

1. Atminties baituose su adresais nuo 00000 iki 000FF yra užrašytos reikšmės nuo 0 iki 255. Koks bus komandos INT 55h pertraukimo vektoriaus absoliutus adresas?
2. Apskaičiuokite kontrolinio taško pertraukimo vektoriaus absoliutų adresą.
3. Atminties baitai su adresais nuo 00000 iki 000FF užpildyti baitų reikšmėmis nuo -128 iki 127. Apskaičiuokite INT 39h pertraukimo apdorojimo procedūros absoliutų adresą
4. Atminties baitai su adresais nuo 00000 iki 000FF užpildyti baitų seka nuo -128 iki 127. Apskaičiuokite INT 15h pertraukimo procedūros IP reikšmę šešioliktaine sistema.

Užduočių atsakymai

1. **00154**
2. **0000C**
3. **6DBC4**
4. **D5D4**

Kas yra flag field‘as ?

Flag field’as (žymių laukas) – yra sveiko skaičiaus tipo kintamasis, kurio kiekvienas dvejetainis skaitmuo (bitas) pasako apie kažkokio požymio buvimą arba nebuvimą.
Alternatyvus apibrėžimas: tai yra kodavimo būdas, kada lauko bitai (pagal tam tikrą aprašą ir eiliškumą) nusako požymio buvimą arba nebuvimą.

Flag’as (žymė) – vienas konkretus požymis (jo aprašas). Kai flag‘o reikšmė **1**, tai reiškia požymio **buvimą**, kai **0** – jo **nebuvimą**

Pagrindinė Flag Field‘o paskirtis: laikyti esamą kažkokio daikto ar mechanizmo būseną, kad pagal ją būtų galima priiminėti konkrečius, nuo situacijos priklausančius sprendimus.

Status Flag‘as Intel8086 archtektūroje

- Status flag‘as (SF) yra žodžio dydžio registras, kuris savyje laiko procesoriaus būsenos požymius.
- Su SF‘o registru galima atlikti *TIK DVI OPERACIJAS*: jį padėti į steką (**PUSHF**) ir išimti iš steko (**POPF**).
- Pagal funkcionalumą SF‘o požymius galima skirstyti į:
  - **operacijų (pasako, kas būdinga paskutinei operacijai ir rezultato laukui dvejetainiame lygmenyje)**
  - **procesoriaus būsenos kontrolės (pasako, kaip veiks / elgsis procesorius tam tikrų situacijų metu)**.

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

Carry Flag (CF) požymis (poz. – 0)

Pernešimo požymis

CF požymį galima suprasti keliais būdais:

- Parodo ar įvyko pernešimas (atimties atveju – pasiskolinimas) vyriausiajame bite (CF =1, jeigu įvyko; CF =0, jei ne)
- Yra papildomas bitas iš kairės (***c*** 0000 0000)
- Parodo ar rezultatas netilpo į rezultato lauką (be ženklo) (CF =1, jei netilpo, CF =0, jei tilpo)

Pavyzdžiai:

1111 1111	1000 1000	1000 0000 **SVARBU!**
0000 0001	0110 1000	1000 0000 Po loginių komandų, CF'as
1 0000 0000	0 1111 0000	1 0000 0000 visada 0.
CF = 1	***CF = 0***	***CF = 1***

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)

Pavyzdžiai:

1111 0000	**0000 0000**
ZF = 0	*ZF = 1*

Papildomas pernešimo / pasiskolinimo požymis

Auxilliary Carry Flag (AF) požymis (poz. – 4)

AF požymis nurodo, ar įvyko pernešimas tarp jaunesniojo ir vyresniojo pusbaičio.

Pavyzdžiai:

```
0000 1000
0000 1000
0001 0000
```

AF = 1

```
1111 1111
0000 0001
0000 0000
```

AF = 1


```
1000 0000
1001 0000
0001 0000
```

AF = 0

SVARBU!

Po loginių komandų, AF yra neapibrėžtas

Sign Flag (SF) požymis (poz. – 7)

Ženklo požymis

SF požymis parodo, kokia yra rezultato lauko vyriausiojo bito reikšmė.

Pavyzdžiai:

```
0000 0000
```

SF = 0

```
1111 1111
```

SF = 1

```
1000 0000
```

SF = 1

Overflow Flag (OF) požymis (poz. – 11)

Ką rodo OF požymis priklauso nuo paskutinės operacijos tipo:

Aritmetinė operacija: aritmetinės operacijos dėmenys yra laikomi skaičiais su ženklu. Jei rezultato laukas atitinka teisingą operacijos rezultatą, tai OF =0, jei ne – OF =1.

Pavyzdžiai:

```
0000 0101(5)          1111 1111(-1)          1000 0000(-128)
0000 1001(9)          1000 0000(-128)        1000 0000(-128)
0000 1110(14) OK!     0111 1111(127) Blogai! 0000 0000(0) Blogai!
OF = 0                OF = 1                 OF = 1
```

Postūmio operacija: jei vyksta perstumimas **per vieną bitą** ir **po perstūmimo pasikeičia ženklo bitas**, tada OF =1, jei nepasikeičia – OF =0.

Jei įvyksta perstūmimas per daugiau nei vieną bitą – OF = neapibrėžtas

Pavyzdžiai:

Right Shift 1
0000 1001
0000 0100
OF = 0

Left Shift 1
0100 0000
1000 0000
OF = 1

Left Shift 2
0010 0000
1000 0000
OF = NA!

Loginė operacija: OF =0

Procesoriaus būsenos kontrolės požymiai

Direction Flag (DF) požymis

DF parodo kaip vykdant eilutinės komandas keičiasi registrai *SI, DI*.
Jei **DF = 1** – *SI, DI* mažėja, jei **DF = 0** – *SI, DI* didėja

Interrupt Flag (IF) požymis

IF parodo ar leidžiami išoriniai maskuojami pertraukimai.
Jei **IF = 1** – *leidžiami*, jei **IF = 0** – *neleidžiami*.

Trap Flag (TF) požymis

TF parodo ar po kiekvienos įvykdytos komandos vykdomas žingsninis pertraukimas (INT 1), kuris naudojamas debuginimui

Pavyzdinis uždavinys

UŽDUOTIS:
Duotos registrų reikšmės: `DS = FE21, SS = 5634, CS = C131, ES = 3EE3, SF = 04FF, BP = 92A2, BX = C5D6, SI = 45FA, DI = 22F1, SP = FFE4`

Vykdoma baitų sudėtis: 253 + (-126). Kokia bus registro SF reikšmė, įvykdžius sudėtį?

Kaip spręsti:
1) Pasiimam SF ir išsirašom jį dvejetainiu pavidalu
2) Po išrašytu SF‘u pasirašom kiekvieno bituko reikšmę
3) Pasirašom sudėties dėmenis dvejetainiu pavidalu (baitus arba žodžius)
4) Atliekame aritmetinį veiksmą, pasirašom rezultatą.
5) Nustatome flag‘ų reikšmes pagal operaciją
6) Pasirašom naują SF‘ą, pakeisdami reikšmes, kurios pasikeitė
7) Parašom SF‘ą šešioliktainio žodžio pavidalu ir tai ir bus mūsų rezultatas ☺

Sprendžiam pavyzdinį uždavinį pagal schemą

1) Išsirašom SF: `0000 0100 1111 1111`
2) Pasirašom bitukus: `XXXX ODIT SZXA XPXC`

3) Užsirašom operacijos dėmenis dvejetainiu pavidalu:

```
 1111 1101 (253)
 1000 0010 (-126)
1|0111 1111 (127)
```

4) Atliekame sudėtį ir parašome rezultatą:
5) Nustatome flagų reikšmes (kurie keičiasi):

CF: ar įvyko pernešimas už lauko ribų? **TAIP**
PF: ar yra lyginis kiekis 1‘ukų? **NE,** jų yra 7 (nelyginis skaičius)
AF: ar įvyko pernešimas tarp pusbaičių? **NE**
ZF: ar rezultato lauke vien nuliukai? **NE**
SF: ar rezultato lauko ženklo bitas lygus 1? **NE**
OF: ar operacijos rezultatas, kur abu operandai laikomi skaičiais su ženklu, neatitinka rezultato lauko? Ar `1111 1101 (-3) + 1000 0010 (-126) =/= 127`? **TAIP, neatitinka**
6) Pasirašom naują SF: `0000 1100 0010 1011`. 7) Pasirašom 16-aine: **`0C2B`** **<- ATS**

Uždaviniai pasisprendimui

1. Duoti registrai SS=0BD2, CX=DE44, DI=17EE, ES=5F69, CS=45AC, DS=553D, BP=F0DC, SF=CBB1, BX=5D33. Apskaičiuote naują SF reikšmę įvykdžius baitų sudėties komandą dešimtainėms reikšmėms 80 ir -128
2. Registrai ES=BB52, SF=8442, DI=C493, SI=5A29, BP=340F, CX=3F5B, CS=C60D, DX=5E65, AX=353C. Apskaičiuokite naują SF reikšmę įvykdžius baitų sudėties operaciją dešimtainėms reikšmėms 78 ir -94 (8496)
3.

4*.

Registras SF =0000h.
Įvykdomos kodo eilutės:
mov al, 20d
mov bl, -34d
sub al, bl
Kokia bus registro SF reikšmė? (16-aine)

Registras SF =0000h.
Įvykdomos kodo eilutės:
mov al, 1111b
mov bl, 0101b
and al, bl
Kokia bus registro SF reikšmė? (2-aine)

Atsakymai

1. C3A0
2. 8496
3. 0015
4. 0000 0000 000**?** 0100
   AF yra neapibrėžtas, nes buvo vykdoma loginė komanda **AND**

JUMP ir CALL

SS KA 2015-11-10 Miglė

Valdymo perdavimas

Sąlyginis

Besąlyginis

IP registro reikšmė

Nuolatos rodo į sekančią komandą

Reikia žinoti kiek baitų užima programa

Arba reikia suskaičiuoti

Perdavimo tipai

Vidinis artimas

Vidinis tiesioginis

Išorinis tiesioginis

Vidinis netiesioginis

Išorinis neteisioginis

Tipas	CALL	JUMP
Vidinis artimas	-	EB
Vidinis tiesioginis	E8	E9
Išorinis tiesioginis	9A	EA
Vidinis netiesioginis	FF *010	FF *100
Išorinis netiesioginis	FF *011	FF *101

*OPK plėtinys

**CALL atveju PUSH CS PUSH IP

RET

Grįžimas iš CALL

POP IP, POP CS

Reikalingas steko išlyginimas SP:=SP+bet.op

RET tipas	Su steko išlyginimu	Be steko išlyginimo
Vidinis	C2	C3
Išorinis	CA	CB

INT n, IRET

PUSH SF, PUSH CS, PUSH IP, IF=0, TF=0, AA 4n imami 4 baitai;

POP IP, POP CS, POP SF;

IRET kodas CF

Sąlyginis valdymo perdavimas

INTO

LOOP

17 JMP

INTO

INT 4, jeigu flagas OF=1.

Operacijos kodas CE.

LOOP

Veiksmai

-sumažinti CX 1;

-tikrinti sąlygą, jei tenkinama, pridėti 1B;

Komanda	Peršokimui su vieno baito poslinkiu būtina sąlyga
LOOP	gautas CX nelygus 0000
LOOPE LOOPZ	gautas CX nelygus 0000 IR flagas ZF=1
LOOPNE LOOPNZ	gautas CX nelygus 0000 IR flagas ZF=0

JMP if

KOMANDOS PAVADINIMAS	PAVADINIMO PRASMĖ	TIKRINAMA SĄLYGA
JO	Jump if Overflow	OF=1
JNO	Jump if Not Overflow	OF=0
JNAE JB JC	Jump if Not Above nor Equal Jump if Below Jump if Carry	CF=1
JAE JNB JNC	Jump if Above or Equal Jump if Not Below Jump if Not Carry	CF=0

JE JZ	Jump if Equal Jump if Zero	ZF=1
JNE JNZ	Jump if Not Equal Jump if Not Zero	ZF=0
JBE JNA	Jump if Below or Equal Jump if Not Above	CF=1 ARBA ZF=1 (bent vienas)
JA JNBE	Jump if Above Jump if Not Below nor Equal	CF=0 IR ZF=0 (reikalingi abu)
JS	Jump if Sign	SF=1
JNS	Jump if Not Sign	SF=0
JP JPE	Jump if Parity Jump if Parity Equal	PF=1
JNP JPO	Jump if Not Parity Jump if Parity Odd	PF=0

JL JNGE	Jump if Lower Jump if Not Greater nor Equal	SF nelygu OF
JGE JNL	Jump if Greater or Equal Jump if Not Lower	SF=OF
JLE JNG	Jump if Lower Or Equal Jump if Not Greater	ZF=1 arba (SF nelygu OF) (bent vienas)
JG JNLE	Jump if Greater Jump if Not Lower nor Equal	ZF=0 ir SF=OF (reikalingi abu)
JCXZ	Jump if CX Zero	CX=0

0111 0000 poslinkis – JO žymė

0111 0001 poslinkis – JNO žymė

0111 0010 poslinkis – JNAE žymė; JB žymė; JC žymė

0111 0011 poslinkis – JAE žymė; JNB žymė; JNC žymė

0111 0100 poslinkis – JE žymė; JZ žymė

0111 0101 poslinkis – JNE žymė; JNZ žymė

0111 0110 poslinkis – JBE žymė; JNA žymė

0111 0111 poslinkis – JA žymė; JNBE žymė

0111 1000 poslinkis – JS žymė

0111 1001 poslinkis – JNS žymė

0111 1010 poslinkis – JP žymė; JPE žymė

0111 1011 poslinkis – JNP žymė; JPO žymė

0111 1100 poslinkis – JL žymė; JNGE žymė

0111 1101 poslinkis – JGE žymė; JNL žymė

0111 1110 poslinkis – JLE žymė; JNG žymė

0111 1111 poslinkis – JG žymė; JNLE žymė

Sutrumpinimas

J raidė reiškia žodį Jump

N – not, nor. O – odd. Sąlygos paneigimas.

C,Z,S,P – reiškia atitinkamus flagus

A – above (skaičiuose be ženklo), G – greater (skaičiuose su ženklu) reiškia „daugiau“

B – below (skaičiuose be ženklo), L – lower (skaičiuose su ženklu) reiškia „mažiau“

E – equal – lygu

JO, JNO atveju O raidė reiškia OF flagą

1 PS2013

Įvykdžius nurodytą komandą, apskaičiuoti registrų reikšmių sumą:
AL+BL+CL+DL+IP, kai AL=03, BL=02, CL=00, DL=01, AH=00, BH=01, CH=02, DH=03, ES=0000, CS=ABCD, SS=1234, DS=FE21, SP=2222, SF=0000:
0100 E2 90 90 loop ... (0100 yra poslinkis kodo segmente)

2. IT2013

2. Įvykdžius nurodytą komandą, apskaičiuoti sekančios vykdomos komandos absoliutų adresą, kai DS=21FE, SS=5634, CS=0ADF, ES=41E3, BP=9A32, SI=FFF1, DI=22F1, AX=0003, BX=0002, CX=0000, DX=0001, SF=0000:
DCBA E0 90 90 loopne ... (DCBA yra poslinkis kodo segmente)

3. IT2013

6. Registrų reikšmės yra: SI=FFF0, DS=1234, DI=FFFF, ES=1233, registras CX=FFFF, registras SF=FF00.
Kokia bus steko viršūnės reikšmė įvykdžius procedūros tolimo iškvietimo komandą:
46DE 2E FF 59 F9 37 90 90 call cs:... (46DE yra poslinkis kodo segmente)

4.

1. CS=FFFF
9999: 9A 12 34 56 78
Koks yra kviečiamos procedūros AA?

5.

2. CS=ABCD, SS=7894, DS=2222, ES=3333, SP=0001 SI=1111 DI=7894 BP=92A2, BX=BEBE
Vykdomas kodas:
1234: 2E FF D4 90 90 90
1) Sekančios komandos AA?
2) Kviečiamos procedūros AA?

6.

Turimos registrų reikšmės yra AX=7897, BX=AA2E, CX=EE32, DX=12EE, SI=AAEE, DI=DDAA, CS=78AA, SS=DEAE, DS=7700, ES=2EAA.
Apskaičiuokite sekančios vykdomos komandos absoliutų adresą, jei vykdoma komanda...:
9090: E9 90 90 90 90 90 90 90 90 90 (9090 – poslinkis kodo segmente)

7

CS= 1234. Apskaičiuokite kitos vykdomos komandos ea ir aa, įvykdžius komandą:

0014: E9 F9 FF (0014 – poslinkis kodo segmente)

8

CS= 1234, BX= 0008. Duomenų segmento pirmieji 16 baitų atrodo taip:

```
0000: 80 81 82 83 84 85 86 87
0008: 88 89 8A 8B 8C 8D 8E 8F
```

Apskaičiuokite kitos vykdomos komandos ea ir aa, įvykdžius komandą:

0014: FF 67 05 (0014 – poslinkis kodo segmente)

9.

3. Registrų reikšmės yra: DS=FE21, SS=3456, CS=C131, ES=3EE3, BP=92A2, BX=C5D6, SI=45FA, DI=22F1, SP=FFF6. Kokia bus registro SP reikšmė, įvykdžius grįžimo iš artimos procedūros komandą:

C2 10 00

Atsakymai

1. 197
2. 18A3C
3. 46E2
4. 7B792
5. B4F72
6. 7ABC3
7. EA= 0010; AA=12350
8. EA=8E8D AA=1B1CD
9. 0008

Eilutinės komandos (1)

Eilutinės komandos skirtos operacijoms su baitų ar žodžių eilutėmis.

Eilutinės komandos dirba su eilutėmis, esančiomis DS arba ES segmentuose (nebent yra keitimo prefiksas) ir/arba akumuliatoriumi (AX/AL registru).

Eilutinės komandos gali padėti greitai pakeisti tam tikrus atminties blokus norimomis reikšmėmis.

Baito ilgio eilutė.
Dabar joje yra

Žodžio ilgio eilutė.
Dabar joje yra

Atminties gabalas

12345	11
	12
	45
	89
	BA
	CC
	A5
	57
	7E
	AC
	DC
	AA
	AA
	AA
12353	AA

Eilutinės komandos (2)

Visos eilutinės komandos mašininiame kode užima 1 baitą.

Paskutinis eilutinės komandos mašininio kodo bitas yra **w (word)** bitas – jis nurodo, ar bus dirbama su baitais (**w=0**), ar bus dirbama su žodžiais (**w=1**).

Ar eilutinė komanda dirba su baitu, ar su žodžiu, galima nuspręsti ir iš komandos pavadinimo: jei paskutinė raidė B, dirbama su baitais, jei W – su žodžiais.

Pvz.: MOVSB – gale B, reiškia bus dirbama su baitais.

Pvz.: Eilutinės komandos operacijos kodas: 1010 0101 – w=1, reiškia bus dirbama su žodžiais.

Eilutinių komandų klasifikacija

Eilutinės komandos gali būti **palyginimo** arba **duomenų persiuntimo**.

Palyginimo komandos:

1) Palygina dviejų baitų/žodžių reikšmes
2) Atitinkamai pakeičia SF registrą
3) Koreguoja SI ir/ar DI registrus

Duomenų persiuntimo komandos:

4) Keičia atminties baitų arba AX/AL reikšmę
5) Koreguoja SI ir/ar DI registrus

Palyginimo komandos

SCASB – Scan string byte

SCASW – Scan string word

CMPSB – Compare string byte

CMPSW – Compare string word

---

*Pvz.: atliekama komanda SCAS**B**. Kas vyksta:*

1) *Paimama AL reikšmė*
2) *Paimama **baito**, esančio absoliučiu adresu ES*10h + DI, reikšmė*
3) *Atliekamas CMP (AL – baitas)*
4) *Pakeičiami atitinkami SF registro flag‘ai*

Komandos pavadinimas	**Atliekamas veiksmas**
SCASB	cmp al, es:[di]
SCASW	cmp ax, es:[di]
CMPSB	cmp ds:[si], es:[di]
CMPSW	

Duomenų persiuntimo komandos

MOVSB – Move string byte

MOVSW – Move string word

LODSB – Load string byte

LODSW – Load string word

STOSB – Store string byte

~~STOSW – Store string word~~

*Pvz.: atliekama komanda MOVS**W**. Kas vyksta:*

1) *Paimama **žodžio**, esančio absoliučiu adresu DS*10h + SI, reikšmė*
2) *Paimta reikšmė įrašoma į atminties vietą, kurios absoliutus adresas yra ES*10h + DI*
3) *Priklausomai nuo DF (Direction Flag), didinamos arba mažinamos SI ir DI registrų reikšmės*

Komandos pavadinimas	**Atliekamas veiksmas**
MOVSB	mov es:[di], ds:[si]
MOVSW	
LODSB	mov al, ds:[si]
LODSW	mov ax, ds:[si]
STOSB	mov es:[di], al
STOSW	mov es:[di], ax

SI ir DI eilutinėse komandose

Po kiekvienos eilutinės komandos yra keičiamos SI ir/arba DI reikšmės. Tačiau, jei kažkuris iš registrų komandoje nebuvo panaudotas, jo reikšmė išlieka nepakitusi.

SI ir DI keičiamos pagal šias taisykles:

- DF = 1 → reikšmė yra mažinama
- DF = 0 → reikšmė yra didinama
- Komanda dirba su žodžiais → reikšmė keičiama per 2
- Komanda dirba su baitais → reikšmė keičiama per 1

Pvz.: Vykdoma komanda SCASW, o DF=1. Kaip pasikeis SI ir DI?

1) SCASW → naudojamas tik DI registras, SI nesikeis.

2) SCASW → dirbama su žodžiais, reikšmė keisis per 2.

3) DF = 1 → reikšmė bus mažinama.

Ats.: SI nesikeis, DI:=DI-2.

Pakartojimo prefiksai

Pakartojimo prefiksai leidžia pakartoti eilutinę komandą.

Visi pakartojimo prefiksai užima vieną baitą mašininiame kode. Paskutinis baito bitas yra **z (zero)** bitas.

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

Palyginimo komandos su pakartojimo prefiksu yra kartojamos tol, kol CX!=0 ir ZF=z.

Duomenų persiuntimo komandos su pakartojimo prefiksu yra kartojamos tol, kol CX!=0.

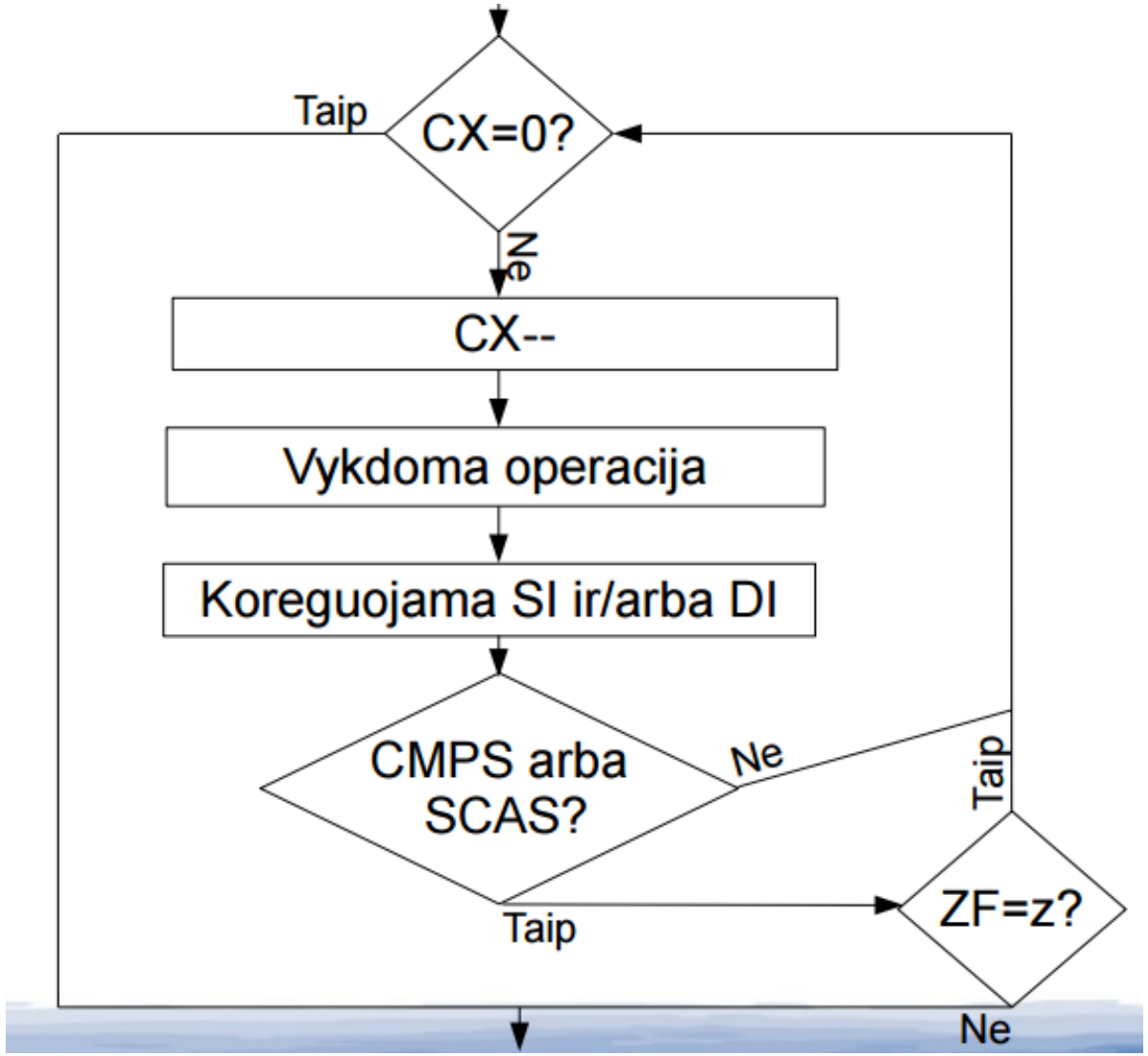
Eilutinių komandų vykdymo schema

Pvz.: CX=0005, SF=0000, SI = 1234, DI = ABCD, visų ES ir DS baitų reikšmės yra nuliai. Vykdoma komanda REPNZ CMPSW. Kokios bus SI, DI ir CX reikšmės įvykdžius komandą?

1) Ar CX = 0? Ne.
2) CX:=0005-1=0004;
3) Vykdom operaciją. Kadangi visi ES ir DS baitai užpildyti nuliais, bus atliekamas toks veiksmas: CMP 0000h, 0000h. Reiškia, ZF tampa 1 (nes atėmę gavom nulį, reiškia ZF=1).
4) Komanda SCASW → dirbam su žodžiais, dalyvauja ir SI, ir DI registrai → DF=0, reikšmės didės → SI:=1234+2=1236, DI:=ABCD+2=ABCF
5) Ar vykdoma komanda yra CMPS arba SCAS? Taip.
6) Ar ZF=z? ZF=1, z=0 →Ne.

Eilutinė komanda įvykdyta, einama vykdyti kitos komandos.

Ats.: SI=1236, DI=ABCF, CX=0004.



Pavyzdinis uždavinys (1)

Registrų reikšmės SI=0000, DI=0000, DS=1234, ES=3333, AX=BABA, CX = 0014, SF = 000F. Duomenų segmento pirmųjų 20-ies baitų reikšmės yra atitinkamai 01h, 02h, 03h, ..., 14h. Vykdoma komanda REP MOVSB. Kokia bus SI, DI ir baitų, atmintyje esančių adresais 33330...33344, reikšmių suma įvykdžius komandą?

Tikslas yra rasti sumą iš trijų skaičių:

1) SI registro reikšmės įvykdžius komandą
2) DI registro reikšmės įvykdžius komandą
3) Baitų, esančių nurodytais adresais, reikšmių sumos

Pavyzdinis uždavinys (2)

Pradėkime nuo baitų, esančių nurodytais adresais. Nusibraižykime, kaip atrodo atmintis prieš vykdant komandą.

Baitai, kurių mums reikia, vis dar nežinomi, tačiau pirmojo reikalingo baito adresas sutampa su ES segmento pradžia. Kadangi komandoje MOVSB rašoma į **ES** segmentą (MOVSB atlieka veiksmą MOV **ES**:[DI], DS:[SI]), pradėkime vykdyti komandą ir žiūrėkime, kaip keisis atminties baitai.

Absoliutus adresas	Baito reikšmė atmintyje
00000	XXh
	…
DS pradžia 12340	01h
	02h
	…
12354	14h
	…
DS pabaiga 2233F	XXh
	…
ES pradžia 33330	XXh
33331	XXh
	…
33344	XXh
	…
ES pabaiga 4332F	XXh
	…
FFFFF	XXh

Pavyzdinis uždavinys (3)

Vykdome komandą REP MOVSB:

1) Ar CX = 0? CX=0014, reiškia ne.
2) CX:=0014-1=0013.
3) Vykdom operaciją. Atliekamas veiksmas: MOV ES:[DI], DS:[SI]. Vadinasi, baito, esančio adresu 12340 (1234*10h+0000), reikšmę įrašysime į adresą 33330 (3333*10h+0000).
4) Komandoje naudojamas ir SI, ir DI → keisis ir SI, ir DI.
   Komandoje dirbama su baitais → keisis per 1.
   DF=0 (iš SF=000F) → SI ir DI didės.
   SI:=0000+1=0001; DI:=0000+1=0001
5) Ar komanda SCAS/CMPS? Ne.

Dabar reiktų eiti vykdyti visko iš naujo, bet pažiūrėkime, kaip atrodo atmintis po vienos iteracijos.

Kas bus po dviejų iteracijų? (Baito, esančio adresu 12341 reikšmė įrašyta į 33331 adresu esantį baitą)

Kas bus po visų iteracijų? Iš viso yra vykdoma 14h (CX reikšmė) kartų, kas yra 20d kartų. Reiškia, visos mūsų žinomos reikšmės duomenų segmente yra perkopijuojamos į pirmus 20 ekstra segmento baitų.

Absoliutus adresas	Baito reikšmė atmintyje
00000	XXh
	...
DS pradžia 12340	01h
	02h
	...
12354	14h
	...
DS pabaiga 2233F	XXh
	...
ES pradžia 33330	01h
33331	02h
	...
33344	14h
	...
ES pabaiga 4332F	XXh
	...
FFFFF	XXh

Pavyzdinis uždavinys (4)

Prisimenam sąlygą: mums reikia baitų, esančių adresais 33330...33344, reikšmių sumos. Įvykdę komandą, jau turime šias reikšmes. Sudedam jas (nepamirštam, kad tai šešioliktainiai skaičiai):

01+02+03+...+09+0A+0B+...+13+14=D2h

Trūksta: SI ir DI reikšmių.

Anksčiau išsiaiškinom, kad ir SI, ir DI didėja per vieną (nes DF=0, dirbam su baitais, naudojami abu registrai). Kadangi eilutinę komandą vykdom 14h kartų, reikia 14h kartų pridėti 01h:

SI:=0000+0014=0014h

DI:=0000+0014=0014h

Sudedam viską: D2+14+14=FAh

Ats.: FAh

Absoliutus adresas	Baito reikšmė atmintyje
00000	XXh
	...
DS pradžia 12340	01h
	02h
	...
12354	14h
	...
DS pabaiga 2233F	XXh
	...
ES pradžia 33330	01h
33331	02h
	...
33344	14h
	...
ES pabaiga 4332F	XXh
	...
FFFFF	XXh

Uždaviniai

1. Registrų SI ir DI reikšmės yra 000A, registras CX=0002, registras SF=0C00. Kokia bus registrų SI ir DI reikšmių suma, įvykdžius komandą: REP STOSW?
2. Registrų SI ir DI reikšmės yra ABCD, registras CX=0000, registras SF=0000. Kokia bus registrų SI ir DI reikšmių suma, įvykdžius komandą: REP LODSW?
3. DI=FFFF, SI=ABBA, CX=0010, SF=FFFF, AX=1234. Visas ekstra segmentas užpildytas baitais, kurių reikšmės 34h. Vykdoma komanda REPNE SCASB. Kokia bus AX ir DI registrų reikšmių suma įvykdžius komandą?
4. SF=FFFF, SI=0013, DI=0009, CX=0003.
   Pirmieji 32 DS baitų atrodo taip:
   0000: ---abcdefghijk—
   0010:---ABCDEFGHIJK—
   Pirmieji 32 ES baitų atrodo taip:
   0000: lmnoprstuv------
   0010: ------LMNOPRSTUV
   Kaip atrodys pateikti DS ir ES fragmentai ir kokios bus CX, SI ir DI reikšmės įvykdžius REPNE MOVSW ?

Atsakymai

1. 0010h
2. 1579h
3. 11232h
4. 0000: lmnop----AB-----
   0010: ------LMNOPRSTUV

Skaičiaus formatai. Sveikieji skaičiai (1)

Dažniausiai programavime naudojami skaičiai – sveikieji ir realieji. Sveikieji skaičiai dažniausiai saugomi *short*, *int* arba *long* duomenų tipuose (žinant, kad skaičius niekada nebus neigiamas ir norint praplėst reikšmių diapazoną, galima naudoti *unsigned*

Tipas	Baitų kiekis	Galimos dešimtainės reikšmės
Short	2	Nuo -32768 iki 32767
Int	4	Nuo -2147483648 iki 2147483647
Long	8	Nuo -9223372036854775808 iki 9223372036854775807

Pastaba: baitų kiekis gali skirtis nuo programavimo kalbos ir esamos architektūros.

```
short fantastiskasShortas;
```

```
int klasiskasIntegeris;
```

```
long ispudingasLongas;
```

```
uint bezenklisPasakiskasIntas;
```

Skaičiaus formatai. Sveikieji skaičiai (2)

Kaip saugoti neigiamus skaičius?

Skaičiaus be ženklo ir skaičiaus su ženklu formatais. Du skirtingi skaičiai (neigiamas ir teigiamas) gali atrodyti identiškai (pasivertus juos į bitus), tačiau **nuo to, kaip interpretuosime, priklauso, koks tai skaičius.**

$1001\ 1100_2 = 156_{10}$ (jei tai skaičius be ženklo)

$1001\ 1100_2 = -100_{10}$ (jei tai skaičius su ženklu)

Skaičiaus formatai. Realieji skaičiai

- Kaip saugoti realiuosius skaičius?

  Mums patogu trupmeninę dalį atskirti užrašius kablelį. Užrašę 13,37, mes suprantam, kad tai skaičius $13\frac{37}{100}$.

  Kompiuteriui „užrašyti“ kablelį nėra taip paprasta (įsivaizduojam skaičiaus bitinę išraišką). Ką daryt?

Išsaugoti realiesiems skaičiams buvo sugalvoti *floating point (slankaus kablelio)* formatai. Mūsų architektūroje, 8087 koprocesorius palaiko IEEE-754 standartą.

```
float karaliskasFloatas;
```

```
double patsGeriausiasDablas;
```

Floating point formatai (1)

Single-precision floating-point format / Trumpas realus formatas (4 baitai)

S (1b)	Charakteristika (8b)	Mantisė (23b)

Double-precision floating-point format / Ilgas realus formatas (8 baitai)

S (1b)	Charakteristika (11b)	Mantisė (52b)

Vidinis realus formatas (10 baitų)

S (1b)	Charakteristika (15b)	i bitas (1b)*	Mantisė (63 b)

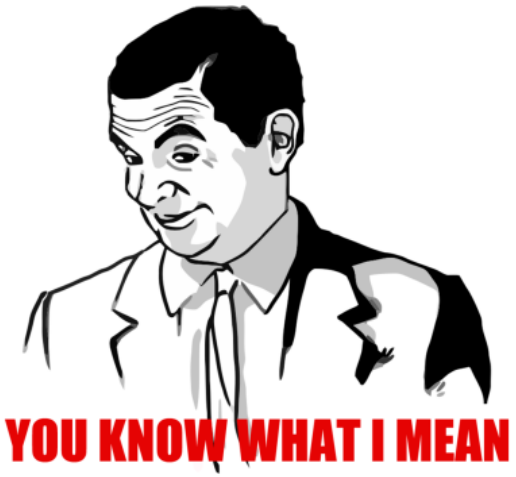
** i bitas parodo, koks skaičius eina prieš mantisę*

	Ženklo bitas	Charakteristika	i bitas	Mantisė
Trumpas realus	1 bitas	8b (eilė + 127_{10})	NĖRA	23 bitai
Ilgas realus		11b (eilė + 1023_{10})		52 bitai
Vidinis realus		15b (eilė + 16383_{10})	1 bitas	63 bitai

Normalizuota forma: **$(-1)^S * 2^{eilė} * 1,$mantisė (i bitas = 1)**

Nenormalizuota forma: **$(-1)^S * 2^{eilė} *$ i,mantisė**

Kaip atsiminti eiliškumą pagal dydį:
Iš pradžių būna trumpas, po to ilgas, o po to jau ir vidinis...



Floating point formatai (2)

- S (Sign) bitas – parodo, koks yra skaičiaus ženklas.
  Skaičius neigiamas → Sign TRUE → S=1
  Skaičius teigiamas → Sign FALSE →S=0
- Charakteristikos skaičiavimas:
  - Kablelį turime „nustumti“ taip, kad jis būtų po vyriausio vieneto
  - Eilė: per kiek pozicijų „stumsime“ kablelį. Jei stumsime į kairę pusę – eilė teigiama, ~~jeigu į dešinę pusę – eilė neigiama~~.

    Pvz.: Verčiam 75,45 į floating point formatą. Pradedam dirbti su sveikąją dalimi: pasivertę 75 į dvejetainį pavidalą, gaunam 1001001,trupmeninė_dalis. Pastūmus kablelį: 1,001001trupmeninė_dalis. Stūmėm į kairę per 6 pozicijas → eilė = +6.

Floating point formatai (3)

- Nebūtina atsiminti 127 (7Fh), 1023 (3FFh) ir 16383 (3FFFh). Užtenka atsiminti, kiek bitų užima charakteristika (8, 11 arba 15) ir surašyt į vyriausią poziciją nulį, o visas kitas – vienetus.
  Pvz.: Verčiame skaičių į ilgą realų formatą. Vadinasi, charakteristiką užims 11 bitų. Reiškia, charakteristika, dar nepridėjus eilės, bus tokia: 011 1111 1111
- Nebūtina atsiminti, kiek bitų užima mantisė. Užtenka žinoti, kiek bitų išvis užima formatas ir kiek bitų užima ženklo, charakteristikos ir i bitas (jei jis yra). Visi likę bitai bus skirti mantisei.
  Pvz.: verčiam skaičių į vidinį realų formatą. Vadinasi, ženklo, charakteristikos ir i bitai užims 17 bitų (1+15+1). Reiškia, mantisei liks 63 bitai (80 – 17 = 63).
- [Insert your_trick here]

Vertimas į floating point formatą (1)

Turime skaičių -155,48. Norime paversti jį į 4 baitų floating point formatą, o atsakymą užrašyti šešioliktainėje sistemoje.

Skaičius neigiamas → Sign bitas = 1.

Įsiminę sign bitą, toliau dirbame tik su teigiamu skaičiumi (155,48). Verčiant atskirai dirbame su sveikąja ir trupmenine dalimis.

Sveikoji dalis:

- Pasiverčiame sveikąją dalį į dvejetainį pavidalą. $155_{10} = 10011011_2$
- Po vyriausio vieneto yra 7 bitai → jau turime 7 mantisės bitus.
- Mantisė iš viso sudaryta iš 23 bitų. Reiškia, trūksta dar 16-kos (23-7=16).

Vertimas į floating point formatą (2)

Trupmeninė dalis:

- Skaičiaus trupmeninę dalį (0,48) reikia versti į dvejetainę. Mum reikia versti tol, kol gausime 16 bitų (nes tik tiek trūksta iki pilnos mantisės). *Pastaba: iš kuo mažiau baitų sudarytas floating-point formatas, tuo mažesnis jo tikslumas.*
- Vertimo procesas:
  - Dauginam trupmeninę dalį iš 2
  - Pasiimam sveikąją dalį (ją įrašysim į mantisę)
  - Toliau dirbam tik su trupmenine dalim

Vertimas į floating point formatą (3)

0,48*2=0,96	0,68*2=1,36	0,88*2=1,76	0,08*2=0,16
0,96*2=1,92	0,36*2=0,72	0,76*2=1,52	0,16*2=0,32
0,92*2=1,84	0,72*2=1,44	0,52*2=1,04	0,32*2=0,64
0,84*2=1,68	0,44*2=0,88	0,04*2=0,08	0,64*2=1,28

- Kitas būdas versti – greitesnis, bet pavojingesnis. Šiuo būdu trupmeninę dalį dauginam iš 16, pasiimam sveikąją dalį, ją paverčiam į dvejetainę formą ir turim reikiamus bitukus.

0,48*16=7,68	0,68*16=10,88	0,88*16=14,08	0,08*16=1,28
$7_{10}=0111_2$	$10_{10}=1010_2$	$14_{10}=1110_2$	$1_{10}=0001_2$

Turime:

- S bitas = 1
- Eilė = +7, nes kablelį stumiam per 7 vietas į kairę (1001 1011,011... → 1,0011011011...
- Charakteristika = 127+7 = 134 = 1000 0110
- Mantisė sudaryta iš 23 bitų (sudaryta iš 7 bitų iš sveikosios dalies ir 16 bitų iš trupmeninės dalies)

1100	0011	0001	1011	0111	1010	1110	0001
C	**3**	**1**	**B**	**7**	**A**	**E**	**1**

Ats.: C31B7AE1h

Pastebėjimai

- Verčiant trupmeninę dalį pastebėjus besikartojantį pattern‘ą (ciklą), galima dalį ~~dauginimo praleisti~~.
  Pvz.: Versdami 5,2 į dvejetainę formą, gaunam 101, (0011). Normalizuota forma: 1,01(0011). Reiškia, į mantisę įrašę 01, toliau rašysime 00110011... tol, kol neužpildysime visos likusios mantisės.
- Verčiant skaičių, kurio modulis mažesnis už 1, kablelis stumsis į dešinę ir eilė bus neigiama.
  Pvz.: Verčiam -0,04 į 4 baitų floating point formatą.
  $0{,}04_{10}$ = 0,00000111110…$_2$. Stumiam kablelį iki vyriausio vieneto ir gaunam normalizuota formą: 0,000001,11110. Pastūmėm per 6 vietas į dešinę → charakteristika = 127 – 6 = 121.
- Mūsų sprendžiamuose uždaviniuose laikysime, kad mantisė nukerpama ir jos neapvalinsime.
- Pasitikrinimui naudokite interneto IEEE-754 konverterius (pvz. šitą).

Vertimas iš floating point formato

Verčiant iš floating point formato į dešimtainę sistemą, reikia pritaikyti atvirkščią algoritmą prieš tai aprašytam.

- Šešioliktainiai skaičiai verčiam į dvejetainius skaičius
- Išsitraukti ženklo, charakteristikos, (i) ir mantisės reikšmes
- Pasirašyti 1, mantisė
- Charakteristika pasiverčiam į dešimtainį skaičių. Atėmus 7F/3FF/3FFF, gauname eilę.
- Atitinkamai patraukti kablelį į reikiamą pusę per eilės reikšmę
- Gautą skaičių versti į dešimtainę sistemą (*pastaba: jis nebūtinai gausis toks pats, koks buvo prieš vertimą*)
- Pasirašyti atitinkamą ženklą (nusprendus iš ženklo bito)
- Vuolia

Uždaviniai

1. Užrašykite dešimtainį skaičių -33,33 slankaus kablelio formatu 8 baituose šešioliktaine sistema.
2. Užrašykite dešimtainį skaičių 0,00003 slankaus kablelio formatu 4 baituose šešioliktaine sistema.
3. Užrašykite dešimtainį skaičių -13,2 slankaus kablelio vidiniu realiu formatu.
4. Užrašykite dešimtainį skaičių 67,67 slankaus kablelio formatu 4 baituose šešioliktaine sistema.

Atsakymai

1. C040 AA3D 70A3 D70A
2. 37FB A882
3. C002 D333 3333 3333 3333
4. 4287 570A

Kas yra supakuoti / išpakuoti skaičiai?

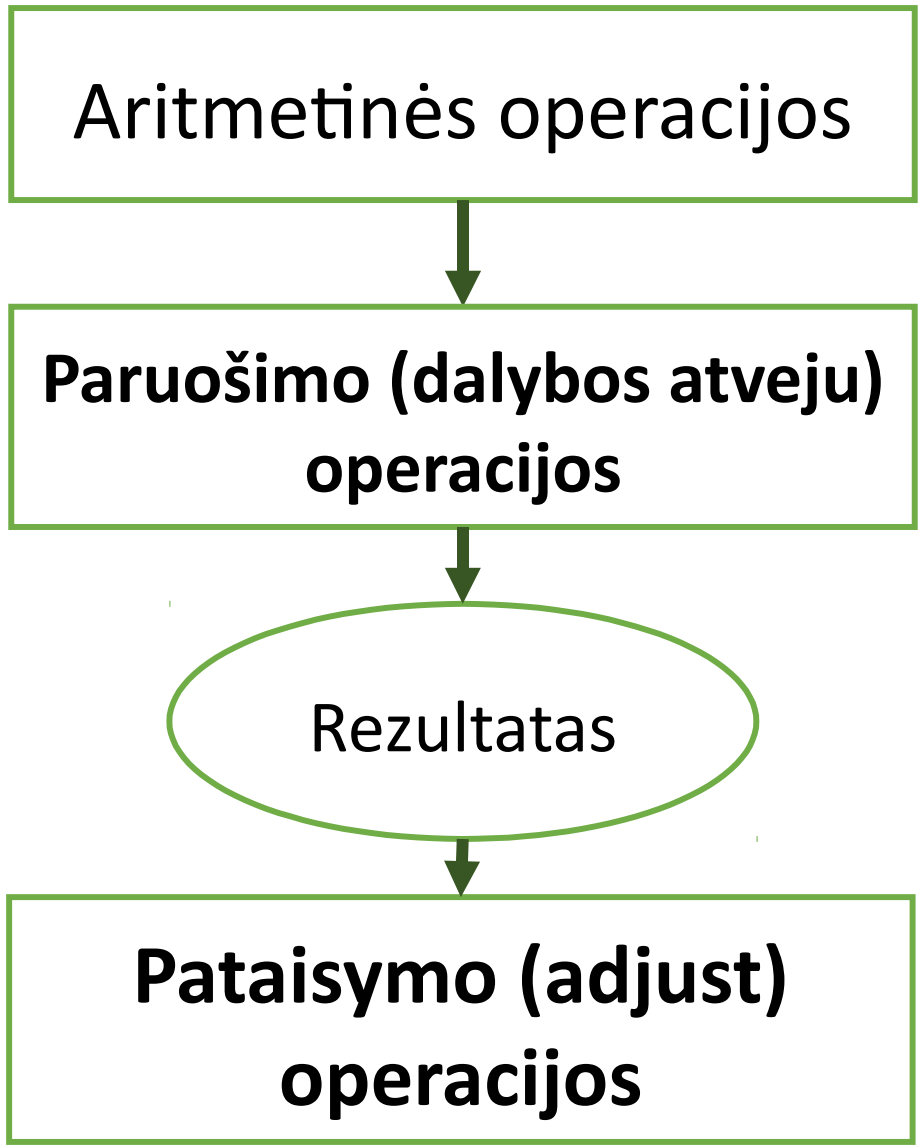
Išpakuotų ir supakuotų skaičių formatai yra tokie skaičių formatai, kai viename baite yra ***TIESIOGIAI*** laikomi dešimtainiai skaitmenys.

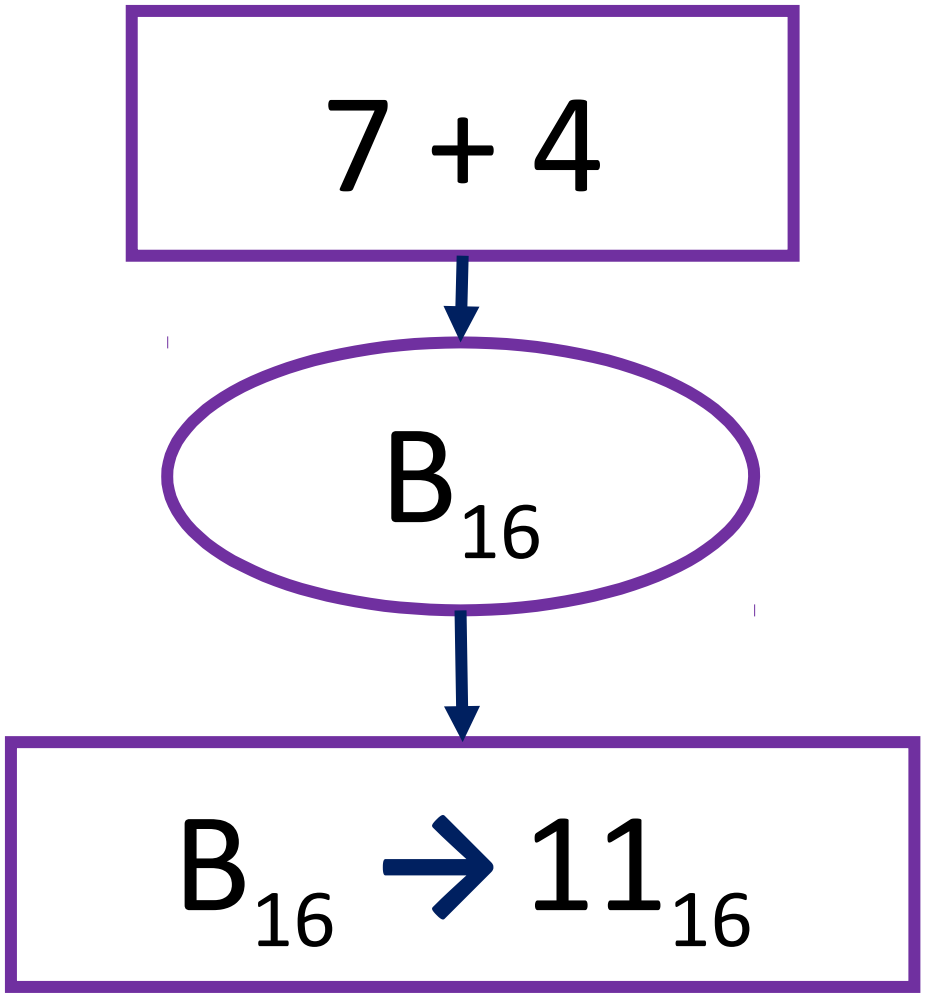
Išpakuotų skaičių formato atveju viename baite, jaunesniame pusbaityje, yra laikomas vienas dešimtainis skaitmuo **(00, 01, 02 ,..., 09)**

Supakuotų skaičių formato atveju viename baite yra laikomi DU dešimtainiai skaitmenys **(00, 01, ...10, 11, 12, ...,98, 99)**

Po aritmetinių operacijų, kuriose dalyvavo šių formatų skaičiai buvo vykdomos vadinamos „pataisymo“ (ang. *Adjust*) operacijos, kurios užtikrina, kad aritmetinės operacijos rezultatas **taip pat būtų dešimtainis skaičius**.

Kaip jie realizuojami assembleryje?





Komandos darbui su Iš/Supakuotais skaičiais

Assembleris darbui su išpakuotais bei supakuotais skaičiais turi 6 komandas:

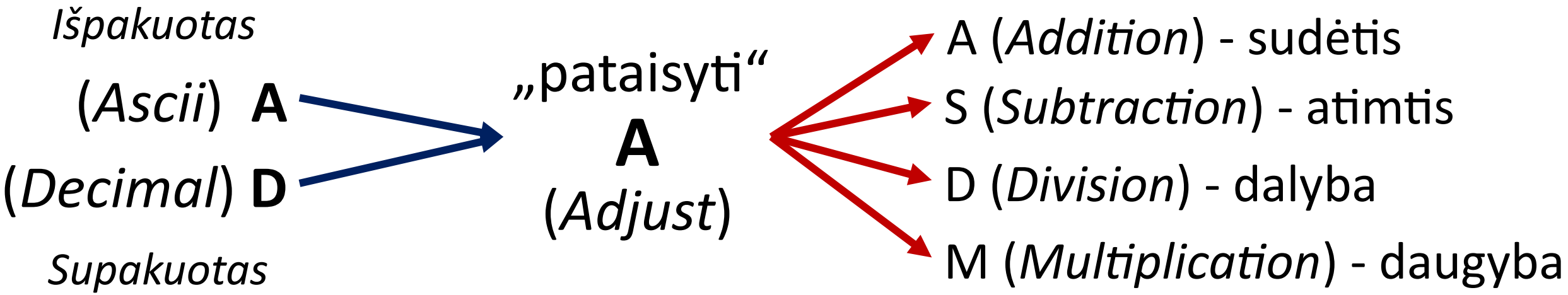
Su **išpakuotais**:

- ***AAA*** (komanda naudojama po sudėties)
- ***AAS*** (komanda naudojama po atimties)
- ***AAM*** (komanda naudojama po daugybos)
- ***AAD*** (komanda naudojama **PRIEŠ** dalybą)

Su **supakuotais**:

- ***DAA*** (komanda naudojama po sudėties)
- ***DAS*** (komanda naudojama po atimties)

Kaip iššifruoti „pataisymo“ komandas



AAA / AAS (Ascii Adjust for Addition / Subtraction)

If *((AL* ***and*** *0Fh) >9)* ***or*** *(arba) (AF=1))*
then

AL := AL +| - 6

AH:= AH +| - 1

AF:= 1

CF:= 1

else

AF:=0

CF:=0

endif

AL:=AL and 0Fh

Jei AAA – tai +

Jei AAS – tai -

Atkreipkite dėmesį!

and – loginė operacija

or – arba sąlyga

AAM

(Ascii Adjust for Multiplication)

$$AH := AL \; div \; 10_{10}$$
$$AL := AL \; mod \; 10_{10}$$

AAD

(Ascii Adjust for Division)

$$AL := AH * 10_{10} + AL$$
$$AH := 0$$

DAA / DAS (Decimal Adjust for Addition / Subtraction)

IF *(((AL* ***and*** *0Fh) >9)* ***or*** *(arba) (AF = 1))* ***then***

$AL := AL \pm 6$

$AF := 1$

endif

IF *((AL>9Fh)* ***or*** *(CF=1))* ***then***

$AL := AL \pm 60h$

$CF := 1$

endif

Jei DAA – tai +

Jei DAS – tai -

Atkreipkite dėmesį!

or – arba sąlyga

Pastaba!

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

Uždaviniams reikia mokėti

- Suprasti, kas yra išpakuotas ar supakuotas skaičius (kuo skiriasi)
- Mokėti visas „pataisymo“ (adjust) komandas, jų algoritmus (kaip jos taikomos) ir jų taikymų specifiką.

Uždavinys 1

Duotos reikšmės:
AX =0102, BX =0205, CX =ACDC, DX =ABBA, SF =0000

Įvykdomos komandos:

ADD AL, BL
AAA

Kokios bus AX ir AF, CF reikšmės?

Sprendimas 1

Peržiūrim sąlygą ir vykdom iš eilės komandas:

ADD *AL, BL* AL = 02h BL = 05h

AL = 02+05 = 07h *Atliekam aritmetinį veiksmą ir nusistatom AF ir CF flag'us.*

AAA (Nuosekliai tikrinam pagal schemą ir sužinom ką reikia daryti)

- (AL **and** 0Fh) => 07h **and** 0Fh = 07h
- 07h > 9 ? **NE** ***arba*** AF = 1 ? **NE**
- **NE** arba **NE = NE** **=> AF = 0, CF = 0; AL = AL and 0Fh;**

Vykdom veiksmus:

AL = AL *and* 0Fh **=>**

07h *and* 0Fh **=> AL = 07h** (pagal schemą pačiam gale įvykdomas *AL:=AL and 0Fh)*

ATS.: AF = 0, CF = 0, AX = 0107h

Uždavinys 2

Duotos reikšmės:
AX =0102, BX =0205, CX =ACDC, DX =ABBA, SF =0000

Įvykdomos komandos:

SUB AL, BL
AAS

Kokios bus AX ir AF, CF reikšmės?

```
0000 0000 0000 0000
XXXX ODIT SZXA XPXC
```

Sprendimas 2

Peržiūrim sąlygą ir vykdom iš eilės komandas:

SUB AL, BL AL = 02h BL = 05h
AL = 02-05 = 0FDh

AAS (Nuosekliai tikrinam pagal schemą ir sužinom ką reikia daryti)

- (AL **and** 0Fh) => FDh **and** 0Fh = 07h
- FDh > 9 ? **TAIP** ***arba*** AF = 1 ? **TAIP**
- **TAIP** arba **TAIP = TAIP =>** **AF = 1, CF = 1; AL = AL - 6; AH = AH – 1;**

Vykdom veiksmus:

1) **AL = F7h; AH = 00h**
2) **AL and 0Fh => AL = 07h** (pagal schemą pačiam gale įvykdomas $AL{:=}AL\ and\ 0Fh)$

ATS.: AF = 1, CF = 1, AX = 0007h

Uždavinys 3

Duotos reikšmės:
AX =ABBA, BX =ACDC, CX =1111, DX =2222, SF =0000

Įvykdomos komandos:

SUB AL, BL
DAS

Kokios bus AX ir AF, CF reikšmės?

Sprendimas 3

Peržiūrim sąlygą ir vykdom iš eilės komandas:

SUB AL, BL AL = BAh BL = DCh
AL = BA-DC = 0DEh

AAS (Nuosekliai tikrinam pagal schemą ir sužinom ką reikia daryti)

- (AL **and** 0Fh) => DEh **and** 0Fh = 0Eh
- 0Eh > 9 ? **TAIP** ***arba*** AF = 1 ? **TAIP**
- **TAIP** arba **TAIP = TAIP** => **1) AF = 1; AL = AL – 6;**

- DEh > 9F ? **NE** ***arba*** CF = 1 ? **TAIP**
- **NE** arba **TAIP = TAIP** => **2) CF = 1; AL = AL – 60;**

Vykdom veiksmus:

```
1)  AL = D8h; AF = 1;
2)AL = 78h;  CF = 1;
```

ATS.: AF = 1, CF = 1, AX = AB78h

Užduotys patiems spręstis

1. Duotos reikšmės: AX = 0102, BX = 0205, CX = ACDC, DX = ABBA, SF = 1111
   Kokia bus BX reikšmė įvykdžius komandas: SUB al, bl bei AAS
2. Duotos reikšmės: AX = 6969, BX = ACDC, CX = ABBA, DX = 0420, SF = 0010
   Koks bus AX reikšmė įvykdžius komandą: AAA
3. Duotos reikšmės: AX = ACDC, BX = ABBA, CX = 0420, DX = F1FA, SF = FFFF
   Koks bus AX reikšmė įvykdžius komandas: ADD al,bl bei DAA

Atsakymai patiems tikrintis

1. 0205h
2. 6A0Fh
3. AC42h

JMP, CALL

1. AX=0003 BX=0000 CX=0001 DX=0000. Įvykdžius nurodytą komandą, koks bus sekančios vykdomos komandos efektyvus adresas?
   FFFA: EB A1 JMP nb (FFFA yra poslinkis kodo segmente)
2. Įvykdžius nurodytą komandą, apskaičiuoti sekančios vykdomos komandos absoliutų adresą, kai AX=0003, BX=0002, CX=0001, DX=0000, SF=0000.
   FFFA: EA 80 90 00 90 90 jmp far ptr label (FFFA yra poslinkis kodo segmente)
3. Registrai AX=0001, BX=0002, CX=0003, DX=0004, SF=1111. Apskaičiuoti valdymo perdavimo adresą, kai duota tokia kodo dalis:
   71EA: E8 F1 B2 call number (71EA yra poslinkis kodo segmente)
4. Registras SS=ABCD, SP=0002, BP=AF00, CX=0010. Kokia bus registro SP reikšmė šešioliktaineje sistemoje įvykdžius išorinę komandą CALL?

Eilutinės komandos

5. Registrų SI ir DI reikšmės yra ABCD, registras CX=FFFF, registras SF=0000. Kokia bus registrų SI ir DI reikšmių suma, įvykdžius komandą REP LODSW?

6. Registrų reikšmės yra SI=FFFE, DS=1234, DI=FFFC, ES=1234, CX=7FFF, SF=FF00. Duomenų segmento baito su adresu FFFE reikšmė yra 01h, o baito su adresu FFFF reikšmė yra 02h. Kokia bus duomenų segmento visų baitų reikšmių suma, įvykdžius komandą REP MOVSW?

Pertraukimai

7. Atminties baitai su adresais nuo 00000 iki 000FF užpildyti reikšmėmis nuo -128 iki 127. Užrašykite INT 1D pertraukimo apdorojimo procedūros absoliutų adresą.

8. Atminties žodžiai su adresais 00000 iki 0001FE užpildyti reikšmėmis nuo 255 iki 0. Apskaičiuokite INT 37h pertraukimo apdorojimo procedūros vektoriaus absoliutaus adreso bei tos procedūros IP ir CS reikšmių sumą.

Status Flag

9. Registras SF= FFFF. Kokia bus SF reikšmė, atliekant baitų sudėtį dešimtainiams skaičiams 255 ir 1.

10. Registras SF= FFFF. Kokia bus SF reikšmė, atliekant baitų sudėtį dešimtainiams skaičiams -128 ir -128

MPL

11. Pasiųskite dešimtainę reikšmę -48 į X registrą per dvi mikrokomandas.

12. Užrašykite dvi mikrokomandas MPL kalba, kurios į registrą MBR, nenaudodamos konstantinių registrų, užrašo skaičių -4.

Atsakymai

1. FF9Dh
2. 99080h
3. 24DEh
4. FFFEh
5. 15798h
6. 18000h
7. 07554h
8. 1FDh
9. F77Fh
10. FF6Fh

11. X=15;MBR=LEFT_SHIFT(COM(1)+COM(1);
    MBR=LEFT_SHIFT(COM(X)+MBR);
12. MBR=MBR+COM(MBR);
    MBR=LEFT_SHIFT(MBR+MBR);