KOPIUTERINE SISTEMA

kompiuteris su savo vidiniais ir periferiniais irenginiais. Programa – komandu rinkinys, kursi gali atlikti veiksmus su kompiuteriu. Asembleris – beveik masininio lygio kalba, zemo lygio programavimo kalba, atliekanti auksto lygio veiksmus.
Programa visada sudaryta is triju skyriu. Pirmas – duomenu, antras – komandu, trecias – steko segmentai. Duomenu segmente tam tikru operatoriu pagalba aprasomi duomenys, su kuriais mes dirbsim. Komandu segmentas visada apraso veiksmus, kuriuos turi atlikti programa. Komandu pavidale apraso algoritma, kuri turime realizuoti. Steko segmentas – pagalbine atmintis, naudojama tarpiniu rezultatu, adresu saugojimui. Kiekvienas segmentas prasideda specialiu operatoriumi segment ir baigiasi ends.
Operatyvine atmintis – tai baitu rinkinys, I kuri galima irasyti informacija. Bitai baite numeruojami nuo 0 iki 7 is desines. Zodis – du gretimai einantys baitai, kurio pirmasis adresas yra lyginis. Zodzio adresas yra kairiojo baito adresas. Dvigubas zodis – du gretimai einantys zodziai.

PROCESORIAUS REGISTRAI

Tai yra vieta procesoriuje. 32 skirsniu procesoriuose yra padidintos galimybes kompiuterio, bet zemesnieji rgistrai sutampa ir pagal pavadinimus ir pagal paskirti su 16 skirsniu procesoriais.
Yra 12 registru, kurie skirstomi I tris grupes: duomenu registrus, registrus – nuorodas, segmentinius registrus. Jie visi vadinami bendro naudojimo registrais.

REGISTRAI

Duomenu registrai:
AX – AH+AL – akumuliatorius
BX – BH+BL – bazinis registras
CX – CH+CL – skaitliuko registras
DX – DH+DL – duomenu registras
naudojami bet kokiam laikinam duomenu issaugojimui.
Registrai – nuorodos:
SI – saltinio registras
DI – priemejo registras
BP – bazes nuoroda
SP – steko nuoroda
SI ir DI yra ekvivalentus vienas kitam, taciau ju paskirtis yra saugoti duomenis tam tikros bazes atzvilgiu, isrenkant duomenis is operatyvines atminties. Bazes adresas nurodomas register BP arba BX. BP gali buti naudojamas kaip nuoroda dirbant su duomenimis steke, taip pat daugelyje aritmetiniu ir loginiu komandu laikinam duomenu saugojimui. SP naudojamas tik darbui su steku, visada nurodo steko virsune.
Segmentiniai registrai:
CS – komandu segmento registras
SS – steko segmento registras
DS – duomenu segmento registras
ES – papildomo duomenu segmento registras

Jie saugo pradinius programos segmentu adresus ir leidzia I juos kreiptis.
Kiti:
IP – sis registras vadinamas komandu nuoroda. Issaugo komandos, kuri turi buti atlikta sekanti, adresa. Programiskai sis registras neprieinamas, jo reiksme keicia procesorius, atsizvelgdamas I komandos ilgi.

FLAGS

veliaveliu registras, atspindintis procesoriaus busena po komandos ivykdymo Maximum gali buti 16 veliaveliu.
0-ne – CF – pernesimo veliavele. Isijungia, jei yra duomenu pernesimas (ji itakoja aritmetines operacijos, arba jei yra klaida kreipiantis I sistemines funkcijas).
2-a – PF – pariteto veliavele. Isijungia, jei 8 skirsniu rezultate yra lyginis vienetuku skaicius.
4-a – AF – papildomo pernesimo veliavele. Isijungia, jei dirbame su supakuotais duomenimis.
6-a – ZF – nulio veliavele. Isijungia, jei rezultatas yra 0 po operacijos ivykdymo.
7-a – SF – zenklo veliavele. Isijungia, jei rezultato zenklas yra neigiamas.
8-a – TF – naudojama derinimo programose, kai jas norime atlikti zingsniais. Jei ji ijungta, procesorius po kiekvienos komandos ivykdymo leidzia sustabdyti porgramos vykdyma, norint analizuoti programos darba.
9-a – IF – leidzia arba draudzia procesoriui reaguoti I pertraukimus nuo isoriniu irenginiu.
10-a – DF – naudojama programose, dirbanciomis su eilutemis, ir nurodo, kuria kryptimi turi buti perziurimi duomenys eiluteje.
11-a – OF – isijungia tuo atveju, jei yra perpildymas.

SEGMENTINIS ADRESAVIMAS

Pagrindine procesoriaus charakteristika – registru skirsniskumas (bitu skaicius registre), taip pat isoriniu adresu ir duomenu sinu(liniju) kiekis. Maksimalus sveikas skaicius, su kuriuo gali dirbti mikroprocesorius – 2 sesioliktuoju-1=65kb. Adresine atmintis turi 20 liniju (2 dvidesimtuoju minus 1=1Mb). Tam kad su 16 skirsniu adresais butu galima kreiptis I bet kuri 20 skirsniu atminties taska, procesoriuje numatytas segmentinis atminties adresavimas, kuris realizuotas 4 segmentiniu registru pagalba. Segmento adreso reiksme yra ta, kad 20 skirsniu adresas isskaiciuojamas sudedant 20 skirsniu segmento pradzios adresa, kuriame yra mineta lastele su 16 skirsniu poslinkiu nuo segmento pradzios. Pradinis segmento adresas be 4 zemiausiu bitu yra saugomas viename ir segmento registru. Si reiksme vadinama segmentiniu adresu. Kiekviena karta I vidini registra uzkraunant segmento adresa, procesorius automatiskai ji padalija is 16 ir tokiu budu bazinis adresas saugomas viename is vidiniu registru. Esant reikalui kreiptis I tam tikra lastele, procesorius prideda prie bazinio adreso lasteles poslinki nuo segmento pradzios ir taip gaunamas fizinis lasteles adresas. Daugyba is 16 padidina adresuojamu lasteliu diapazona iki 1Mb. Taigi kreipinys I bet kuria lastele vykdomas tik per segmentus, t.y. logiskas erdves uzdedamas ant reikiamu fiziniu erdviu operatyvineje atmintyje. Segmento dydis negali buti didesnis nei 64 kb. Pradinis segmento adresas yra dalintinas is 16, t.y. be jaunesniojo 16-nio skirsnio uzrasomas I viena is segmentiniu registru. Poslinkis

adresuojamas lasteles, kreipiantis I ja per kazkokia komanda, nurodomas naudojant tam tikrus adresavimo budus. Visi segmentai operatyvineje atmintyje yra isdestomi tokiu pat budu, kaip jie aprasyti programos pradiniame tekste. Pries komandu segmenta sistema patalpina papildoma (sistemini) segmenta, t.y. programos prefixa, kur patalpinta sistemine informacija, reikalinga programos vykdymui. Uzkraunant programa I operatyvine atminti abu, ES ir DS registrai, iguana ta pacia reiksme ir rodo I programos prefixo pradzia. Tai reiskia, kad is programos yra galimybe kreiptis I ta papildoma sriti operatyvineje atmintyje. Komandu segmento adresas patalpinamas I CS ir dazniausiai registro IP reiksme tampa lygi 0, nes programa pradeda darba nuo 1 komandos segmente. IP pradine reiksme formuojama is programos zymes, kuri pateikiama paskutiniame programos sakinyje. Registro SS formuojama sistemai uzkraunant programa I atminti ir lygiagreciai formuojama registro SP reiksme, kuris yra nuoroda I steko virsune. SP reiksme priklauso nuo steko dydzio ir visada lygi steko dydziui programos uzkrovimo metu. Visi programos segmentai prasideda nuo paragrafo ribos (adresai dalinami I 16). Kadangi segmento ilgis daznai nera kartotinas 16, todel tarp segmentu operatyvineje atmintyje atsiranda tarpai. Paprastai paskutine komanda int 21h (grizimas I DOS) ir programos ilgis near kartotinas 16, tas tarpas iki sekancio segmento uzpildomas 0, kurie deasembliuojant virsta komanda add.

STEKO SEGMENTAS

Stekas – tai programos sritis laikinam bet kokiu duomenu saugojimui. Pagrindine steko ypatybe yra savotiskas duomenu isrinkimas, t.y. bet kuriuo metu steke yra prieinamas tik virsutinis elementas, toks kuris I steka pakliuvo paskutinis. Tik iskrovus is steko virsutini elementa galime prieiti prie sekancio. Steko elementai isdestyti operatyvineje atminty, skirtoje stekui, pradedant nuo steko dugno, t.y. nuo jo max adreso, ir uzpildoma nuosekliai mazejant steko virsunes adresui. Virsutinis prieinamo elemento adresas saugomas steko virsunes registre SP. Patalpinti bet kokia reiksme I steka push, paimti bet kokia reiksme is steko pop. Push sumazina steko nuorodos reiksme per 2 ir patalpina operanda nurodytu adresu, kuris saugomas registre SP. Isemus is steko duomenis – jis neissivalo, bet jame esantys elementai neprieinami, nes pasikeite SP reiksme, stekas laikomas tusciu.
Stekas naudojamas ir darbui su paprogramemis. Tuo atveju pati sistema naudoja steka, kad isiminti adresa, kur turi buti grazintas valdymas, paprogramei baigus darba. Net tuo atveju, jei tiesiogiai nanaudojame steko, privalom aprasyti ji programoje, nes sistema naudoja ji savo tikslams. Steke programa taip pat saugo visu registru reiksmes, kai ji eina I pertraukimo apdorojima.

KOMANDU FORMATAI

Registras – registras
OK d w 11 reg1 reg2
Komanda uzima du baitus, atlieka veiksmus su duomenimis, patalpintais registruose. Po operacijos:
Reg1:=reg1*reg2
Reg2:=reg2*reg1

OK – skaicius, kuris nusako, koks veiksmas turi buti atliktas tarp duomenu registruose (sie turi buti bendro naudojimo). W nustato operandu dydi: w=1, kai zodis, w=0, kai baitas. D=0, kai rezultatas saugomas reg2, d=1, kai rezultatas saugomas reg1.

Reg w=1	w=0	reg w=1	w=2
000	AX	AL	100 SP	AH
001	CX	CL	101 BP	CH
010	DX	DL	110 SI	DH
011	BX	BL	111 DI	BH

registras – atmintis

komandos ilgis – nuo 2 iki 4 baitu.

OK d w mod reg mem adr(0-2)

Reg:=reg*adr

Adr=adr*reg

w nustato operandu dydi: w=1, kai zodis, w=0, kai baitas. D=0, kai rezultatas saugomas atmintyje, d=1, kai rezultatas saugomas registre. Mod nusako, kiek baitu uzima operandas adr (jei mod=00, adr uzima 0, jei mod=01, tai adr uzima 1, jei mod=10, adr uzima 2 baitus). Mem nurodo adresu modifikavimo buda: a8 telpa 1 baite adresas, a16 – dviejuose. Jei komandoje adresas neuzduotas, jis skaitomas lygiu 0. jei mod=11, komanda pavirsta tipo registras – registras komanda.

Registras – tiesioginiai duomenys

Uzima nuo 3 iki 4 baitu.

OK S W 11 OK reg im(1-2)

OK1 – kokia komanda turim atlikti, rezultatas saugomas registre, w nurodo operando dydi. OK1 nurodo grupe komandu, kuriai priklauso si komanda, OK patikslina komanda. Tiesioginiai duomenys im gali uzimti 1 arba 2 baitus.

Atmintis – tiesioginiai duomenys

Ilgis – 3 – 6 baitai

OK S W 11 OK1 mem adr(0-2) im(1-2)

Adr:=adr*im

OK patikslina pacia komanda.

ADRESAVIMO BUDAI

Registrinis adresavimas

Kai komandoje nurodomi registru numeriai, kuriuose yra saugomi duomenys

MOV BP, SP

Tipo BP=SP

Tiesioginis (betarpiskas) adresavimas

Operandas gali buti arba vienas baitas, arba vienas zodis. Reiksme gali buti skaicius, adresas, ar bet koks ASCII kodas ar simboline israiska.

MOV AX, 4000h

Tiesiogine adresacija

Kai komandoje nurodomas simbolinis lasteles vardas su kuriuo turime atlikti veiksmus.

MOV DX, MAS

Jei reikia kreiptis I lastele su konkreciu baito numeriu, tai ta adresa galime nurodyti betarpiskai kaip operanda. Pries tai reikia apibrezti segmentini registra tam atminties skyriui, kur yra mineta lastele.

Bazinis ir indeksinis adresavimas

Santykinis operando adresas yra registre, kad nurodyti jog mes imame duomenis ne is registro, bet is atminties, kurios adresas nurodytas registre, tam mes ta registra uzrasome tarp lauztiniu skliaustu.

MOV AX, [BX]

Tokiam adresavimui galim naudoti tik BX ir BP registrus, jei naudojami SI ir DI registrai – gauname indeksini adresavima.

Dirbant su lasteles turiniu, jos adresas isskaiciuojamas sudedant 2 registru reiksmes, t.y. segmentinio registro turinys plius registro reiksme, kuri nurodyta komandoje kaip adresavimo budas.

Bazinis ir indeksinis adresavimas su poslinkiu

Naudojami skirtingi registrai BX, BP – baziniai, SI, DI – indeksiniai. Tarkim turim aprasyta masyva.

MOV DL, mas[BX]

Rezultate gausime registro DL reiksme, lygia BX-ojo masyvo nario reiksmei. Cia elemento adresas isskaiciuojamas, kaip suma registro BX ir simbolinio masyvo adreso, kuris sutampa su masyvo pradzia.

Bazine – indeksine adresacija

Adresas isskaiciuojamas kaip suma bazinio ir indeksinio registru reiksmiu. Galimos kombinacijos [bx] [si] arba [bx] [di] arba [bp] [si] arba [bp] [di]. Jeigu kaip bazinis registras naudojamas BX, tai segmentinis butinai DS. Bazinis BP, segmentinis butinai SS

MOV BX, [BP] [SI]

Bazine – indeksine adresacija su poslinkiu

Operando adresas isskaiciuojamas sudedant 3 reiksmes: bazinio registro, indeksinio registro ir papildoma reiksme

MOV mas[BX][SI], 10

I BX+SI+maspradzia – ataja lastele siunciamas 10.

ASEMBLERIO KALBOS PAGRINDAI

Tai programavimo kalba, leidzianti uzrasyti programa ir ja transliuoti I masinini koda. Programa uzrasoma sakiniais, kurias apraso komanda arba direktyva, operandas, komentaras.

[vardas] direktyva operandas1 [,operandas2] [;komentaras]

[zyme:] komanda operandas1 [,operandas2] [;komentaras]

vardas eina tieisog su direktyva. Zyme nurodo adresa pirmo komandos baito. Vardas atskiria viena direktyva nuo kitos, taciau direktyva gali buti ir be vardo. Transliavimo metu, varduo gali buti priskirtos tam tikros charakteristikos. Komanda ir direktyva yra mnemokodai, kurie atitinka tam tikra masinine komanda arba transliatorius is tos direktyvos suformuoja atitinkamus duomenis. Komandos ar direktyvos operandai apraso objektus, su kuriais atliekami veiksmai. Jais gali buti skaiciai, israiskos, adresai, simboliniai adresai, registru numeriai. Israiskose gali buti naudojami aritmetiniu operaciju simboliai. Is tokiu sakiniu formuojamas programos iseities kodas, kurie sudeliojami I atskirus segmentus.

Sakiniai: **1. Negalima sakinio perkelti I kita eilute. 2. negalima vienoj eilutej rasyti keliu sakiniu. 3. sakinys ne ilgesnis nei 131 simbolis. 4. komantarai gali buti rasomi bet kurioje programos vietoje, prasideda kabliataskiu. Tuscia eilute taip pat suprantama kaip komentaras.**

Operacijos kodas – butinas atributas. Tai kodas, nurodantis, koks veiksmas turi buti atliktas. Esant ne vienam operandui, vienas nuo kito jie atskiriami kableliais. Direktyva skirta duomenu aprasymui programose. Jei ji turi varda, veliau sis vardas gali buti panaudojamas komandose kaip operandas. Direktyvu vardai taip pat yra tarnybiniai zodziai, nurodantys pirmojo aprasyto baito adresa segmento pradzios atzvilgiu.
DB apraso duomenis, kurie operatyvineje atmintyje yra formuojami viename baite.
[vardas] DB [operandai] [;komentarai]
Operandai nurodo reksmes, kurios formuojamos poreatyveneje atmintyje. Esant keletaij ju jie nurodomi per kaklbeli. Pirmajame is ju pirskyriamas nurodytas vardas. Yra 2 budai formautoti baito reiksme. Su ? ir konkreti reiksme.
Operatyvineje atminty nuo to paties adreso gali prasideti skirtingo dydzio duomenys: baito zodzio dvigubo zodzio. Operandas kosntanta su reikesmemis nuo –128 iki 255. Bet kuriuo atveju atmintyje iskiriamas 1b.
Direktyva su keliais operandais tai vienas is budu naudojamas aprasyti masyva. Masyvuose vardas paparstai suteikiamas pirmam elementuj. Jei yra keli simboliais aparsome poepratyvineje atmintyje greta vienas kito, juos galima aprasyti kaip eilute. Pasikartyojimo Operatorius DUP. Naudojamas sutrumtinti duomenu aprasymuj, kaip reikia aprsaryti keliata vienodu reiksmiu greta viena kitos.
Kad galetumeme dirbti su skirtingo tipo duomenimis, naudojamas tranformavimo operatorius.
<tipas> PTR <israiska>
tipas gali buti byte, word, dword
pvz.: MOV BYTE PTR [SI], 0
pagal adresa registre SI nusius 0 byte ilgio.

IVEDIMO / ISVEDIMO KOMANDOS

Tai darbas su ivedimo – isvedimo irenginiais. Jie skirstomi I vidinius: vidine atmintis, procesorius; ir isorinius: visi kiti like. Ivedimas ir isvedimas yra suprantami kaip apsikeitimas informacija tarp procesoriaus ir bet kurio kito irenginio. Atskaitos taskas yra procesorius, ir ivedimas yra procesas, kai duomenys is isorinio irenginio perduodami procesoriui. Isvedimas – kai duomenys is procesoriaus perduodami isoriniams irenginiams. Informacija tarp procesoriaus ir isoriniu irenginiu perduodami per portus – specialius registrus.
Wisi portai yra vienbaiciai, du gretimai senatysa portai gali buti nagrinejami kaip portas, kurio dydis yra zodis.Potencialiai portai numeruojami nuo 0 iki ffff. Su kievienu soriniu irenginiu yra susijes konkretuas portas. Ju numeriais is ankto zinomi. Isronis ireng. Gali uzrasyti infomracija arba nuskaityti ja is porto.
Ivedima atitiktu
IN AL,n arba IN AX,n Isvedimuj OUT n,AL arba OUT n,AX
Pagal komanda IN infomarija persiunciame is porto I registra.Porto numeris gali buti uzduotas skaiciais arba registre DX. Ivedimo isvedimo senarijus panasus visiems irenginiams ir vyksta per 2 portus. Vienas skirtas duomenis ir vadinimas duomenu portu, kitas ivairiai valdymo uniformacjai ir vadinmas valdymo portu. Norint oragizuoti ivedima isvedima reikia zinoti portu numerius, valdanciuosius ri atsakancsiuosiu signalus ir kitas detalias. Skirtingiems

irenginiams si informacija skirtinga, kiekvienoje programoje reikis is naujo aprasyti.

Yra suprogramuoti pertraukimo sistema, leidzianti apdoroti pertraukimus kiekvienam irenginiuj. Pertraukiant turi buti isimenami registrai CS ir IP. Jie isimenami steke, taipat isaugomos visos veliaveliu reiksmes.

DUOMENU ISVEDIMAS IS KLAVOS

Pirmiausia turi buti nustatymas max siomboliu skaicius. Tai butina norint ispeti vartotoja apie perkrovima. Kiti nenumatyti simboliai yra ignaruojami. Vienops komandos metu gali buti ivesta iki 255 simboliu. Turi buti sikirtas laukas kur komanda baigusi duomenu ivedima irasys trikraji simboliu skaiciu ir reikia nurodyti adresa, kur turi buti talpinami simboliai. Ivedimo metu naudojami reigistrai AH (jo reiksme 0AH) ir DX (nurodoma ivedimo srytis) Tada nurodomas is klavos int 21H
Ivedimas baigiamas nuspaudus ENTER (13) taipat formauojamas ivedimo srityje bet neiskaitomas I ivestu simb skaiciu.
08H leidzia ivesti viena simboli, kuris formauojamas registre AL.Siuo atveju simbolis nerodomas ekrane.

EKRENO VALDYMAS IR KURSORIAUS NUSTYMAS

Jis apibrezimas kordinatemis (eilutes ir stuleplio numeris). Ekranas turi 25X80
Kairio visrutinio kampo adresas 0000 (10) 0000h(16)
Virsutinio Desiniojo kapo adresas 0079 (10) 004Fh(16)
Kairiojo apatinio 2400(10) 1800h(16)
Desiniojo apatinio 2479 (10) 184Fh(16)
Kursoriaus nustatymuj i AH siunciame 02h, I BH 0, I DH reilutes numeri, I DL stulpelio numeri. Tuomet komanda INT 10h nustato kursoriu.
Ekrano valymas uzduodamas kaip ekrano sritis, kuria norima isvalyti, kaip kairiojo virsutinio ir desiniojo apatinio kampu koordinates. I AX sunciame 0600h, I BH 07h, I CX kairiojo virsutinio kampo koord, I DX desiniojo apatinio koord. Komanda INT 10h.

TOLIMI PERĖJIMAI

Dirbome iki šiol su vienu programiniu segmentu. Tai yra būdinga trumpom programom. Bendru atveju programoje gali būti tiek daug komandų, kad jos netelpa viename segmente. Tuo atveju programoje aprašome kelis komandinius segmentus. Pvz.

```
C1 Segment
Assume cs:C1,...
Start: Mov ax,0

Jmp Far Ptr L
C1 ENDS

C2 SEGMENT
```

ASSUME CS:C2
L: INC BX

C2 ENDS
Pabrėšime, kad kiekvieno segmento pradžioje turi būti nurodyta komanda ASSUME, kurioje segmentiniam registrui CS priskiriamas konkretus šio segmento atitikmuo. Priešingu atveju sutikęs pirmąją žymę transliatorius fiksuos klaidą.
Priminsime, kad PK komandos (kuri turi būti vykdoma) adresas užduodamas registrų CS ir IP pora . CS nurodo atminties pradžią kur yra komanda, o registre IP yra poslinkis iki tos komandos, atskaitytas nuo segmento pradžios. Bet kurio iš tų regisrtų turinio pasikeitimas yra ne kas kitas kaip perėjimas , nes keičiasi vykdomos komandos adresas.
Jei keičiasi tik IP, tai reiškia kad perdavimas yra tame pačiame segmente. Tokie perėjimai vadinami artimais arba vidiniais. Jei programa nedidelė ir visos jos komandos telpa viename segmente, tai kitokie perėjimai ir nereikalingi. Bet jei programoje yra keletas komandų segmentų, tai atsiranda poreikis organizuoti perėjimus iš vieno segmento į kitą. Tokie perėjimai vadinami tolimaisiais arba tarpsegmentiniais. Esant tokiems perdavimams keičiasi registro CS reikšmė ir registro IP reikšmė. CS nustatomas į segmento C2 pradžią , o į IP užrašomas poslinkis iki žymės šiame segmente. PK numatytos komandos, realizuojančios tokius tolimus perėjiimus, tačiau jos visos yra besąlyginio perdavimo komandos. Sąlyginio perdavimo komandos visada yra artimos. Asembleryje tokios komandose kaip mnemokodas visada naudojamas tas pats JMP, kaip ir esant artimam perdavimui, bet naudojami kiti operandų ipai. Tolimo perdavimo komandos vėlevėlių nekeičia.

Tiesioginis tolimas perdavimas JMP FAR PTR <žymė>

Čia FAR nurodo asembleriui, kad žymė tolima, kad ji yra kitame komandiniame segmente. Pagal šią komandą registras CS nustatomas į segmento pradžią, kuriame yra žymė, o į registrą IP užrašomas poslinkis iki tos žymės nuo segmento pradžios.
CS:=seg<žymė>; IP=offset <žymė>
Taip ankstesniam pavyzdyje nurodytas tplimas perėjimas į komandą, kurios žymė yra L.

Šalutinis tolimas perėjimas JMP m32
Šioje komandoje nurodomas adresas dvigubo žodžio, kuriame turi būti absoliutinis perėjimo adresas užduotas kaip pora seg:ofs, ir užrašytas "perverstam" pavidale: t.y. ofs turi būti užrašytas adresu m32, o segmento numeris adresu m32+2. Komanda perduoda valdymą pagal šį absoliutinį adresą, t.y. užrašo į registrą CS dydį seg, o į registrą IP ofs reikšmę:
CS:=[m32+2]; IP:=[m32]

Pvz.

X dd L; X: offset L, X+2 :seg L

JMP X; per4jimas 5 L (CS:=seg L, IP:=offset L)

Užrašant šalutinio perdavimo komandas reikia elgtis atsargiai. Jeigu komandoje nurodytas vardas X , kuris aprašytas iki šios komandos, tai sutikęs ją, assembleris jau žino kad X reiškia dvigubą žodį ir vadinasi teisingai supras kad šioje komandoje turima omenyje tolimas šalutinis perdavimas. Bet jei X bus aprašytas vėliau, tada assembleris, sutikęs perdavimo komandą, nežinos šiuo atveju kokio pobūdžio turimas omenyje perdavimas ar tai artimas perdavimas pagal žymę ar ar tai artimas šalutinis perdavimas, ar tai tolimas šalutinis perdavimas. Tokioje situacijoje assembleris įtaria, kad nurodytas vardas – tai esamo komandų segmento žymė ir formuoja artimo perdavimo mašininę komandą. Vėliau, kai bus nustatyta, kad šiame segmente nėra tokios žymės, assembleris fiksuos klaidą. Kad to išvengti, reikia valdymo perdavimo į priekį atveju aiškiai nurodyti, kad vardas nurodo dvigubo žodžio kntamąjį. Tam naudojame operatorių PTR.

JMP DWORD PTR X

Kaip buvo pastebėta, tokio pobūdžio patikslinimus reikia naudoti ir esant artimam perdavimui. Jei viską apibūdinsime viską kas susiję su besąlyginio perdavimo į priekį komandomis, tai gausime tokią taisyklę: jei X yra nuoroda į priekį, tai besąlyginio perdavimo komandą reikia užrašyti sekančiai:

JMP X	**artimas tiesioginis ilgas perdavimas**
JMP SHORT X	**artimas tiesioginis trumpas perdavimas**
JMP FAR PTR X	**tiesioginis tolimas perdavimas**
JMP WORD PTR X	**artimas šalutinis perdavimas**
JMP DWORD PTR X	**tolimas šalutinis perdavimas**

Analoginio pobūdžio patikslinimus reikia daryti ir tada, kai komandoje JMP nurodytas šalutinis kreipinys , pvz. JMP [BX] ; pagal nutylėjimą assembleris nagrinėja tokio pobūdžio komandas kaip artimą šalutinį perdavimą.
Kas liečia atvejį, kai X nuoroda atgal, tai perdavimo tipą reikia būtinai nurodyti tik vienu atveju- esant tolimam tiesioginiam perdavimui: assembleris netgi žinodamas, kad X yra žymė iš kito komandinio segmeno, vis tiek nenagrinės tos komandos
JMP X kaip tolimo perjimo. Esmė tame, kad sutikdamas bet kokią žymę, assembleris automatiškai priskiria jai tipą NEAR (artimas) ir toliau vadovaujasi šiuo tipu. (assemblerio kalboje yra direktyva LABEL, kurios pagalba galima žymei priskirt tipą FAR). Pakeisti šį tipą galima tik su operatoriumi PTR.

PAPROGRAMĖS –PROCEDŪROS

Dažnai programose, ypač dideliuose, reikia keletą kartų spręsti tą pčią užduotį ir todėl reikia daug kartų aprašyti tokią pačią komandų seką, sprendžiančią šią užduotį. Kad išvengti pasikartojančių komandų grupių, tokias komandų grupes aprašo vieną kartą ir apiformina atatinkamu būdu, o po to reikiamose vietose tiesiog perduoda valdymą į tas komandas, ir kai jos atidirba, grąžina valdymą atgal. Tokia komandų grupė, kuri sprendžia tam tikrą užduotį ir kuri organizuota tokiu būdu, kad ją galima būtų naudoti neribotą skaičių kartų ir į ją būtų galima kreiptis iš bet kurios programos vietos, vadinama paprograme.

Jos atžvilgiu kita programos dalis vadinama pagrindine programa. Lengva suprasti, kad paprogramės tai procedūrų analogas aukšto lygio programavimo kalbose. Pati paprogramė – tai analogas procedūros aprašymo, o kreipinys į paprogramę yra analogas procedūros operatoriaus. Realizacija paprogramės assembleryje yra sudėtingesnė nei procedūros aprašymas. Esmė čia, kad aukšto lygio kalbose už mus tai daro transliatoriai, o programuojant assembleryje mums niekas nepadeda, todėl čia reikia atlikti viską pačiam programuotojui.

KUR TALPINTI PAPROGRAMES?

Bendru atveju kalbant jas galima talpinti bett kur. Bet reikia prisiminti, kad pati savaime programa neturi būti vykdoma, o turi būti vykdoma tik tada, kai yra į ją kreipins. Todėl talpinti ją reikia taip, kad į ją atsitiktinai nebūtų galima pakliūti. Jei yra keletas paprogramių , tai jas galima sudėti greta. Paprastai paprogramės dedamos arba gale komandų segmento už paskutinės komandos, arba komandų segmento pradžioje prieš komandą nuo kurios programa turi pradėti veikti. Didelėse programose kartais paprogramės iškeliamos į atskirą segmentą.

KAIP APIFORMINTI PAPROGRAMES?

Bendru atveju komandų grupę, kuri bus kaip paprogramė galima niekaip neišskirti. Tačiau asembleryje priimta paprogrames apiforminti ypatingu būdu- kaip procedūrą. Tai yra tam tikru būdu naudinga. Paprogramės kaip procedūros apiforminimas atrodo taip:
<procedūros vardas> PROC <parametrai>
Procedūros kūnas
<procedūros vardas> ENDP

Kaip matome, procedūros kūnas eina po direktyvos PROC, o po jos eina direktyva ENDP. Tose abiejose direktyvose nurodomas tas pats vardas- vardas, kurį mes davėme paprogramei. Reikia atkreipti dėmesį, kad direktyvoje PROC po vardo nededame dvitaškio, bet šis vardas skaitomas žyme, kurią turi pirmoji procedūros komanda. Šį vardą galima naudoti perdavimo komandoje ir perdavimas bus vykdomas į šią komandą.
Direktyva PROC yra parametras – tai gali būti NEAR (artimas) arba FAR (tolimas). Parametro gali ir nebūti, tada priimama kad jis yra NEAR. Esant parametrui NEAR arba nesant jokiam parametrui, procedūra yra vadinama "artima", o esant FAR parametrui- "tolima". Į "artimą" procedūrą galima kreiptis tik iš to paties komandų segmento kuriame ji aprašyta, ir negalima kreiptis iš kitų komandinių segmentų, o į tolimą procedūrą galima kreiptis iš bet kokių komandinių segmentų. Assembleryje vardai ir žymės, aprašyti procedūroje, nėra lokalizuojami jos viduje, todėl jie turi būti unikalūs ir negali sutapti su kitais vardais, naudojamais programoje.

PROCEDŪRŲ IŠKVIETIMAS IR GRĮŽIMAS IŠ JŲ

Kaip iškviesti procedūras ir grįžti iš jų? Programuojant aukšto lygio kalbomis, procedūrų vykdymui pakanka tik parašyti procedūros vardą ir faktinius parametrus. Tada mūsų nejaudina kaip procedūra pradės dirbti ir kaip ji grįš į pagrindinę programą. Assembleryje visus valdymo perdavimus iš programos į procedūrą ir atvirkščiai reikia realizuoti patiems. Čia yra dvi problemos: kaip iš pagrindinės programos perduoti valdymą procedūrai ir kaip iš jos grįžti į pagrindinę programą. Pirmoji problema sprendžiama paprastai. Pakanka tik perduoti valdymą į pirmą procedūros komandą, t.y. nurodyti perdavimo komandoje procedūros vardą. Sunkiau yra realizuoti grįžimą. Esmė ta, kad į procedūrą galima kreiptis iš įvairių programos vietų, vadinasi grįžti reikia į skirtingas vietas. Pati procedūra nežino kur ji turi perduoti valdymą, kai baigs savo darbą.

PARAMETRŲ PERDAVIMAS PER REGITRUS

Aukšto lygio kalbose kad perduoti parametrus paprogramei užtenka juos tik užrašyti juos į operatorių, kuris kviečia procedūrą.
Asembleryje parametrai ir procedūros rezultato perdavimas yra kitas. Asembleryje nėra fomalaus dalinimo į procedūras ir funkcijas, Tačiau pagal turinį yra dalinama į procedūras ir funkcijas. Perduodami faktiški parametrai skirtingai. Paprasčiausias būdas yra registrai. Pagrindinėje programoje faktiniai parametrai yra užrašomi į kokius nors registrus, o procedūra toliau jais naudojasi. Analogiškai galima pasielgti ir su rezultatais, jei tokie yra. Procedūra užrašo rezultatus, o programa vėl naudojas. Kokie bus tam naudojami registrai priklauso nuo paties programuotojo.

PARAMETRŲ PERDAVIMAS PER REIKŠMES

Pvz. Reikia išskaičiuoti c=max(a,b)+max(5,a-1), kur visi skaičiai – skaičiai su ženklu ir yra žodžio tipo. Išskaičiavimą max(x,y) aprašysim kaip procedūrą-funkciją, tuo pat susitarsime apie sekantį: pirmas parametras (x) programa perduoda per AX registrą, o kitą (y) –per registrą BX, o rezultatą (max) procedūra turi grąžinti per registrą AX. Esant tokioms sąlygoms procedūra ir pagrindinės programos fragmentas atrodys taip.

```
; procedūra : AX=max(AX,BX)     ;PAGRINDINė PROGRAMA
MAX         PROC FAR                MOV  AX,a   ;AX:=a
            CMP  AX,BX              MOV  BX,b   ;bx:=b
            JGE  MAX1               CALL MAX    ;AX:=MAX(a,b)
            MOV  AX,BX              MOV  c,AX   ;Išsaugome AX
MAX1:       RET                     MOV  AX,5   ;AX:=5
MAX  ENDP                           MOV  BX,a
                                    DEC  BX     ;BX:=a-1
                                    CALL MAX    ;AX:=MAX(5,a-1)
```

```
                                        ADD   c,AX   ;c:=max(a,b)+max(5,a-
1)
```

Parametrų perdavimas persiuntimo būdu

Tegu programoje yra kreipinys, kai kintamieji A ir B (kurie yra teigiami skaičiai) ir reikia priskirti jiems kokias tai reikšmes, tai yra persiūsti duomenis. Kad kažką įrašyti į lastelę reikia žinoti tos lastelės adresą. Todėl jei priskyrimas numatytas atlikti procedūroje, procedūrai reikia nurodyti tų lastelių adresą ir tai turi atlikti pagrindinė programa. Tokiu būdu parametro perdavimas persiuntimo būdu reiškia adreso perdavimą, kuris atitinka faktinį parametrą. Kaip perduoti tą adresą? Per registrą: pagrindinė programa užrašo į kokį nors registrą adresą faktinio parametro, o procedūra jį iš ten pasiima. Koks tai registras? Bendru atveju bet koks, bet geriau jei tai bus registras modifikatorius, t.y. BX, BP, SI, DI kadangi procedūrai reiks atlikti adresų modifikaciąTegu mūsų procedūrai D mes pasirinkome registrą BX. Tai reiškia kad iki jos darbo pradžios registre BX bus tos lastelės adresas (A arba B) kurios turinį procedūra turi pakeisti. Tokioje situacijoje pasiekti į šią lastelę galima kreiptis naudojant konstrukciją [BX].

```
; Pagrindinė programa
…
LEA BX,a     ;;BX:=Adresas a
CALL D       ; procedūra D
LEA   BX,b   ;;BX:=Adresas b
CALL D
…
```

```
        ;Procedūra
D       PROC
PUSH CX      ;išsaugom CX
        MOV  CL,4
SHR     WORD PTR [BX], CL
POP     CX      ;atstatom registrą CX
RET
D       ENDP
```

REGISTRŲ IŠSAUGOJIMAS PROCEDŪROJE

Kaip matome, procedūroje D bus naudojimas registras CL, kuriame procedūra patalpina poslinkio dydį. Ar gali procedūra keisti registro turinį? PK ne taip jau daug registrų, o jie yra labai naudojami daugelyje komandų. Didelė tikimybė, kad procedūroje ir programoje gali būti reikalinga naudoti tuos pačius registrus. Todėl reikia kad procedūra prieš naudodama registrus įsimintų jų turinį. Tai galima atlikti kai procedūros pradžioje jų turinys užrašomas saugojimui steke, ir po to procedūra registrus gali naudoti savo nuožiūra. Prieš grįtant į pagrindinę programą tuos registrus reika atstatyti perrašant į juos reikšmes iš steko. Yra viena ypatybė, netgi jei mums reikia išsaugoti tik vieną registro baitą mes turime įsiminti viso registro turinį, nes stekas gali skaityti ir rašyti duomenis tik žodžiais. Yra specialios komandos PUSHA ir POPA, kurios įsimena ir atstato visus bendro naudojimo registrus.

SUDĖTINGO TIPO PARAMETRŲ PERDAVIMAS

Tai gali būti masyvai ar struktūros. Tuo atveju geriau perduoti ne pačius duomenis, o adresą kur tie duomenys yra. Tada bus galima prieiti iš procedūros prie bet kurių nurodytų duomenų. Tai atliekama netgi tada kai nereikia tų duomenų keisti. Pvz.

```
X DB 100 Dup(?)
Y DB 25 Dup(?)
```

Reikia į DL užrašyti sumą maksimalių reikšmių iš masyvų X ir Y atatinkamai, t.y.

DL=max(X[I])+max(Y[I])

Šiuo atveju du kartus reikia išrinkti maksimalią reikšmę iš masyvų X ir Y.
Tam sudarysime procedūrą MAX. Pradinis masyvo adresas perduodamas per registrą BX, o elementų kiekis masyve per registrą CX, o rezultatas grąžinamas per registrą AL.

```
;procedūra MAX
MAX  PROC
     PUSH BX
     PUSH CX
     MOV  AL,0
MAX1: CMP [BX],AL
     JLE  MAX2
     MOV  AL,[BX]
MAX2: INC  BX
     LOOP MAX1
     POP  CX
     POP  BX
     RET
MAX  ENDP
; Pagrindinė programa
     LEA  BX,X        ;masyvo X adresas
     MOV  CX,100      ;CX yra masyvo elementų skaičius
     CALL MAX         ;AX=max(X[I])
     MOV  DL,AL            ;įsimenam AL
     LEA  BX,Y        ;masyvo Y adresas
     MOV  CX,25       ;elementų kiekis masyve
     CALL MAX         ;AX=max(Y[I])
     ADD  DL,AL
```

PARAMETRŲ PERDAVIMAS PER STEKĄ

Parametrų peradvimas per registrus yra patogus metodas ir dažnai naudojamas. Tačiau tai yra patogu, jei nėra daug parametrų. Jei parametrų daug, tai registrų gali ir neužtekti. Tuo atveju naudojame kitą parametrų perdavimo būdą t.y. jie gali būti perduoti per steką. Pagrindinė programa patalpina parametrus į steką, o procedūra juos iš ten nuskaito. Tai atlikti galima įvairiai.
Taigi procedūra P naudoja k parametrų. Sutarsime, kad prieš kreipiantis į procedūrą parametrai užrašomi į steką. Kokia tvarka? Tai nustato programuotojas. Mes manysime, kad parametrai į steką užrašomi iš kairės į dešinę. Pradžioje užrašomas 1-as elementas, po to kitas ir t.t. Jei kiekvienas parametras yra žodis, tai pagrindinės programos komandos, realizuojančios kreipimąsi į procedūrą bussekančios

```
; kreipinys į procedūrą
```

```
        push   a1
        push   a2
        …
        push   ak
        call   p
```

kai pradeda dirbti procedūra ir iš kart kyla problema kaip jai pasiimti parametrus. Tai yra problema, kai reikia pasiekti steko elementus jų neišimant iš steko. Tai galima realizuoti naudojant registrą BP, t.y. užsiūsti į šį registrą steko viršūnės adresą (registro SP reikšmę), o vėliau naudoti išraišką [BP+I]. Šiuo atveju mes gadiname registro BP reikšmę. Todėl pradžioje jįreikia išsaugoti ir tik po to į jį persiūsti SP reikšmę. Taigi procedūra turi prasidėti nuo pagrindinių komandų.

```
; pradiniai procedūros veiksmai

P       Proc
        Push   BP
        MOV    BP,SP
        …
tolimesnės procedūros komandos
```

Reikia atsiminti tvarką, kokia parametrai buvo padėti į steką. Procedūros pabaigoje turi eiti galinės komandos. Tai komandos kurios atstato registro BP reikšmę ir dar reikia išvalyti steką nuo parametrų. Steko valymą gali atlikti ir pagrindinė programa. Tam po komandos CALL P reikia įvykdyti komandą ADD SP,2*k. Bet geriau jei tai atliekama procedūroje dėl to, kad į procedūrą galima kreiptis iš įvairių programos vietų ir dėl to komandą ADD reiktų atlikti keliose pagrindinės programos vietose. Taigi procedūra turi išvalyti steką nuo parametrų ir tik tada grąžinti valdymą pagrindinei programai. Tam yra numatytas komandos RET variantas, kur operando pagalba galime nurodyti kiek baitų steke reikia išvalyti. Operando reikšmė traktuojama kaip skaičius be ženklo. Ši komanda pirmiausia iš steko pasiima grįžimo į pagrindinę programą adresą, tada steke išvaloma tiek baitų kiek yra nurodoma ir tik po to valdymas perduodamas į pagrindinę programą. Iš esmės paprasta RET yra
RET 0. (valymo sritis nurodoma baitais o ne žodžiais).

Assemblerinio programavimo kursinis darbas. Užduotys.

1. Apskaičiuokite funkcijos y=a+b-c+1 reikšmę, kai a=17, b=-3, c=2;

2. Aprašome 10 baitų ilgio tekstą atmintyje. Reikia sukeisti vietomis tame lauke 1 ir 5 baitus. Į 3 baitą pasiūsti simbolį "E" , į 6 ir 7 baitus nusiūsti reikšmes 78.

Šakotosios programos

1. Sudaryti programą funkcijos skaičiavimui
F=2a/(x+y), kai a>10
F=x-y kai a<10
a, x ir y reikia įvesti iš klaviatūros. Įvedimui naudoti paprogramę. Rezultatą atspausdinti ekrane.

Ciklai

Peradresavimo ciklai

1. Turime paeiliui įrašytus šimtą skaičių. Reikia rasti šių skaičių sumą.

2. Sudaryti programą, kuri suskaičiuotų kiek yra raidžių "A" masyve T1 ir kiek yra raidžių "B" masyve T2. Masyvo pabaiga žymima simboliu "*" .

Ciklai cikle

3. Turime masyvą, kurio struktūra

Grupės Numeris	Studento Pavardė	Egzaminų Pažymiai	Pažymių Vidurkis

Reikia suskaičiuoti bendrą vidurkį, Čia nežinomas studentų skaičius. Reikia nustatyti pagal masyvo pabaigą. Masyvo pabaiga yra '*'**

Indėnų uždavinys
1627 metais indėnai pardavė Manheteno salą už 24 dolerius. Kiek jie turėtų pinigų 1999 metais, jei šiuos dolerius būtų padėją į banką ir bankas mokėtų 3 procentų metines palūkanas.
Jei pradinė suma yra S0 ir kasmet ji padidėja p procentų, tai po n metų gaunama suma Sn apskaičiuojama iš formulės

$$Sn=S0(1+p)^n$$
Šiuo atyveju S0=24, n=1999-1627=372, p=0.03.

Sumos skaičiavimo algoritmas
1) Skaitliukas =n t.y. kiek metų buvo skaičiuojamos palūkanos
Galutinė suma S

2) S=S0 t.y Suma nuo kurios skaičiuojamos palūkanos, Kiekvienais metais ši suma yra padidėjusi ankstesnių metų atžvilgiu suskaičiuotomis palūkanomis
3) S=S(1+p) T.y. suskaičiuojame sumą su palūkanomis ir ji bus išeities skaičius kitų metų skaičiavimams (aišku jei indėlininkas neatsiims šių palūkanų ar visų pinigų)

Šiam punktui vykdyti organizuojam ciklą tiek kartų kiek metų reikia skaičiuoti palūkanas.

Atsiskaitymui pateikti programų listingus popieriuje.
Programas turėti diskelyje, kad būtų galima pademonstruoti jų darbą.
Vykdymo metu programa turi pradėti darbą nuo autoriaus pavardės ir grupės numerio išvedimo ekrane.

```
;-steko segmentas-
stekas segment stack
      db 256 dup (?)
stekas ends

;-duomenu segmentas-
duom segment
      a dw 17      ; \
      b dw -5      ;  | - priskiriami duomenys kintamiesiems
      c dw 2       ; /
    ten db 10
    ats db 4 dup (?),'$'
    aut db 'Kursinio darbo uzduotis Nr.1',13,10
       db 'Atliko Aurimas Mameniskis, II-01',13,10
       db '2003 01 17',13,10,'$'
    abc db 13,10,'a = 17'
       db 13,10,'b = -5'
       db 13,10,'c =  2',13,10,'$'
    txt db 'a+b-c+1=',7,'$'
duom ends

;-programos segmentas-
program segment
 assume cs:program, ds:duom, ss:stekas

 start:
      mov ax,duom   ;
      mov ds,ax     ;i ds segmenta uzkraunami duomenys
      mov ax, 0002h
```

```
        int 10h
        ;-isvedame pranesima-
        mov ah,09h
        lea dx,aut
        int 21h

        mov ah, 09h
        lea dx, abc
        int 21h

        mov ah, 09h
        lea dx, txt
        int 21h
;-atliekame veiksmus-
        xor ax,ax       ; isvalome al registra
        add ax,a        ; al:=a
        add ax,b        ; al:=a+b
        sub ax,c        ; al:=a+b-c atimame c reiksme
        add ax,1        ; al:=a+b-c+1

        lea si, ats+1
          mov ah,0

                ;Tai darome, kad normaliai isvestu atsakyma
ciklas:
        div ten          ;daliname is desimt
        add ah, 30h
        mov [si],ah       ;masyvo skaitliukas
        dec si           ;mazinam skaitliuka vienetu
        mov ah, 0         ;isvalom ah:=0
        loop ciklas       ;griztame i ciklo pradzia

        mov ah, 09h
        lea dx, ats  ;atspausdina atsakyma
        int 21h      ;doso interuptas

        ;-grazina i dos'a-
        mov ah,4ch
        int 21h

program ends
end start
```

```
;STEKO SEGMENTAS
stekas segment stack
        db 256 dup(?)
stekas ends
;DUOMENU SEGMENTAS
duomenys segment
          prad db 13,10,' Pradinis tekstas: ','$'
        result db 'Pakeistas tekstas: ','$'
          zodis db 'KLAVIATURA', '$'
          sk db 10
          aut  db 'Kursinio darbo uzduotis Nr. 2', 13, 10
               db 'Atliko: Aurimas Mameniskis, II-0/1', 13, 10
               db '2003 01 17',13,10, '$'
          eil db 10, 13, '$'
          ats db 10 dup (?), '$'
          pab db 13,10,13,10,'Programos pabaiga.', 13, 10, '$'
duomenys ends

;PROGRAMOS SEGMENTAS
programa segment
       assume cs:programa, ds:duomenys, ss:stekas
       start: mov ax, duomenys
              mov ds, ax

;EKRANO ISVALYMAS
          mov ax,0002h
          int 10h


;PRANESIMU ISVEDIMAS
          lea dx, aut
          mov ah, 09h
          int 21h

          lea dx, prad
          mov ah, 09h
          int 21h

          lea dx, zodis
          mov ah, 09h
          int 21h
```

```
;TRANSORMAVIMAS
        lea si, zodis

        mov ah, [SI+4]
        xchg ah, [SI]      ;sukeiciame 1 ir 5 baito reiksmes
        mov [SI+4], ah

        mov bh, 'E'
        mov bl, 78

        mov [SI+2], bh     ;i 3 baita irasome 'E'
        mov [SI+5], bl     ;i 6 baita irasome 78
        mov [SI+6], bl     ;i 7 baita irasome 78

;ATSAKYMO ISVEDIMAS I EKRANA
        lea dx, eil
        mov ah, 09h
        int 21h

        lea dx, result
        mov ah, 09h
        int 21h

        lea dx, zodis
        mov ah, 09h
        int 21h

        lea dx, pab
        mov ah, 09h
        int 21h

;GRYZIMAS I DOS
        mov ah, 07h
        int 21h
       mov ah, 4ch
       int 21h

programa ends
end start
```

```
;   f=2*a/(x+y), kai a > 10;
;   f=x-y,       kai a < 10.

;-------------------------------------------------------
stekas segment stack
        db 256 dup (0)
stekas ends
;-------------------------------------------------------
duomenys segment
 autorius db 'Kursinio darbo uzduotis Nr.3:',10,13
          db 10,13,'Atliko: Aurimas Mameniskis, II-0/1'
          db 10,13,'2003 01 17',10,13,'$'
       ten db 10
         a db ?
         x db ?
         y db ?
        pr db 'Tokiu atveju funkcija neapibrezta, nes a=10 ','$'
  klaida1 db 10,13,'Ivyko perpildymas ivedinejant skaiciu','$'
  klaida2 db 10,13,'Ivyko perpildymas atliekant daugyba ','$'
  klaida3 db 10,13,'Ivyko perpildymas atliekant sudeti ','$'
  klaida4 db 10,13,'Klaida! Trupmenos vardiklis lygus nuliui ','$'
   ived_a db 10,13,'Iveskite a: ','$'
   ived_x db 10,13,'Iveskite x: ','$'
   ived_y db 10,13,'Iveskite y: ','$'
      ats db 10,13,'Atsakymas:     .   ','$'
 pran_pab db 'Programos pabaiga.', 10, 13, '$'

duomenys ends
;-----------------------------------------------
programa segment
        assume cs:programa,ds:duomenys,ss:stekas
start:
        jmp prog_pradzia
iv proc
        xor ax,ax      ; nunulinam ax
        xor bx,bx      ; nunulinam bx
        xor cx,cx      ; nunulinam cx
        mov ah,08h     ; ivedinesime pagal 08h komanda
        int 21h
        cmp al,13      ; tikrinam ar ne ENTER
        je pab         ; jei TRUE tai sokam pab
        cmp al,'-'     ; ar ivesta su - zenklu
        je zen         ; TRUE tai lekiam i zen zyme
```

```
        jmp plus      ; FALSE tai i plus
ppp:
        mov ah,08h
        int 21h
        cmp al,13
        je pab
plus:
        cmp al,'0'    ; tikrinam ar ivestas skaicius
        jb ppp        ; FALSE
        cmp al,'9'    ; '0'-'9' tai reziai
        ja ppp
        mov dl,al
        mov ah,02h
        int 21h
        sub al,30h
        mov dl,al
        mov al,bl
        imul ten
        jo error
        mov bl,al
        add bl,dl
        jo error
        jmp ppp
zen:
        mov dl,'-'
        mov ah,02h
        int 21h
min:
        mov ah,08h
        int 21h
        cmp al,13
        je pab
        cmp al,'0'
        jb min
        cmp al,'9'
        ja min
        mov dl,al
        mov ah,02h
        int 21h
        sub al,30h
        neg al
        mov dl,al
        mov al,bl
```

```
        imul ten
        jo error
        mov bl,al
        add bl,dl
        jo error
        jmp min
error:
        mov ah,09h
        lea dx,klaida1
        int 21h
        jmp pabaiga
pab:
        ret
iv endp

prog_pradzia:
        mov ax,duomenys   ; uzkrauname i ds
        mov ds,ax         ; duomenis

        mov ax, 0002h     ;nuvalome ekrana
          int 10h

        xor di,di         ; nunulinam di
        mov ah,09h        ; isvedame pranesima apie autoriu
        lea dx,autorius
        int 21h
        mov ah,09h        ; pranesimas apie ivedima a reiksme
        lea dx,ived_a
        int 21h
        call iv           ; kvieciame iv procedura
        mov a,bl          ; a reiksmei briskiriame bl
        mov ah,09h        ; pranesimas apie ivedima x reiksme
        lea dx,ived_x
        int 21h
        call iv           ; kvieciame iv procedura
        mov x,bl          ; x reiksmei priskiriame bl
        mov ah,09h        ; pranesimas apie ivedama y reiksme
        lea dx,ived_y
        int 21h
        call iv           ; kvieciama iv procedura
        mov y,bl          ; y reiksmei priskiriam bl
        cmp a,10          ; lyginam a su 10
        jg r1             ; jei a>10
```

```
        cmp a,10          ; lyginam a su 10
        jl r2             ; jei a<10
        cmp a,10
        je pran           ; jei a=10 varom i pran
r1:
        mov al,2          ; al:=2
        imul a            ; al:=2*a
        jo error2         ; jei overflow tai error2
        mov dl,x          ; dl:=x
        add dl,y          ; dl:=y
        jo error3         ; ar ne perpildymas
        jz error4         ; ar ne nulis
        js su_zenklu      ; ar neigiamas
at:
        idiv dl           ;2a/(x+y)
        cmp di,1          ; ar d=1
        jne xx            ; jei nelygu varom xx
        mov ats[13],'-'
xx:
        jmp spau
        jmp pabaiga
su_zenklu:
        neg dl
        mov di,1
        jmp at
r2:
        xor ax,ax       ; nunulinam ax
        xor bx,bx       ; nunulinam bx
        mov si,17       ; si:=17
        mov al,x        ; al:=x
        mov bl,y        ; bl:=y
        neg bl          ; bl:=-y
        add al,bl       ; al:=x+(-y)
        jo error3       ; jei overflow
        jns spaux
        neg al
        mov ats[13],'-'
        jmp spaux
        jmp pabaiga

spaux:
        div ten         ; dalinam is 10
        add ah,30h      ; prie ah+30
```

```
        mov ats[si],ah
        xor ah,ah
        dec si          ; si:=si-1
        cmp al,0        ; al=0
        je pabaiga      ; TRUE jei al=0 tai pabaiga
        jmp spaux       ; FALSE sokam i spaux

error2:                 ; jei ivestas skaicius per didelis
        mov ah,09h
        lea dx,klaida2
        int 21h
        jmp griztam_i_dosa

error3:                 ; jei sudauginus ivyko perpildymas
        mov ah,09h
        lea dx,klaida3
        int 21h
        jmp griztam_i_dosa

error4:                 ; ar ne nulis
        mov ah,09h
        lea dx,klaida4
        int 21h
        jmp griztam_i_dosa

pran:                   ; atspauzdiname kai a=10
        mov ah,09h
        lea dx,pr
        int 21h
        jmp pabaiga

spau:
        mov si,16
        mov cx,3
        mov di,18
        xor bx,bx     ; nunulinam bx
        mov bh,ah     ; isimename liekana
        xor ah,ah     ; nunulinam liekana
des:
        div ten
        add ah,30h
        mov ats[si],ah
        xor ah,ah
```

```
        dec si
        cmp al,0
        je ppab
        jmp des


ppab:
        xor ax,ax
        mov al,bh
        imul ten
        idiv dl
        mov bh,ah
        xor ah,ah
tol:
        div ten
        add ah,30h
        mov ats[di],ah
        xor ah,ah
        inc di
        cmp al,0
        jne tol
        loop ppab

pabaiga:
          mov ah, 09h
        lea dx,ats
        int 21h

;KURSORIAUS NUKELIMAS I APACIA
          mov ah, 02h
          mov bh, 00h
          mov dh, 15h
          mov dl, 00h
          int 10h
          xor dx,dx
          lea dx, pran_pab
          mov ah, 09h
          int 21h
griztam_i_dosa:
        mov ah,4ch
        int 21h
programa ends
end start
```

```
;STEKO SEGMENTAS
stekas segment stack
        db 256 dup(?)
stekas ends

cr equ 13
lf equ 10

;DUOMENU SEGMENTAS
duomenys segment
        aut  db 'Kursinio darbo uzduotis Nr. 4', cr, lf
             db 'Atliko: Aurimas Mameniskis, II-0/1', cr, lf
             db '2003 01 18',cr,lf,lf, '$'
   skaiciai db 100 dup (?)
      suma dw ?, '$'
       des dw 10
       ats db cr,lf,lf,'Skaiciu suma yra: ', '$'
       pab db cr,lf,lf,'Programos pabaiga.', cr, lf, '$'
duomenys ends

;Programos segmetas
programa segment
      assume cs:programa, ds:duomenys, ss:stekas
      start: mov ax, duomenys
             mov ds, ax

;EKRANO ISVALYMAS
       mov ax, 0002h
         int 10h

;PRANESIMU ISVEDIMAS
        lea dx, aut
       mov ah, 09h
       int 21h

       mov cx, 100
       lea bx, skaiciai       ;adreso persiuntimas i registra
       xor dx, dx
       xor ax, ax

;SKAICIU ISVEDIMAS
        cikl_sk:
```

```
            mov al, cl     ;skaicius
            mov [BX], al
            add dx, ax
            call sk_isv
            call tarpas     ;tarpo spausdinimas
            inc bx

        loop cikl_sk

        mov ax, dx
        push ax
;SKAICIU SUMA
          lea dx, ats
        mov ah, 09h
        int 21h
        pop ax
        call sk_isv

          lea dx, pab
          mov ah, 09h
        int 21h

;GRYZIMAS I DOS
          mov ah, 07h
          int 21h
        mov ah, 4Ch
        int 21h

;---Skaiciu isvedimo aprasymas---
sk_isv:
        push ax
        push cx
        push dx

        xor cx, cx
        sk_isv_1:
            xor dx, dx
            div des
            add dx, 30h
            push dx
            inc cx
            cmp ax, 0
        jnz sk_isv_1
```

```
        mov ah, 02h
        sk_isv_2:
                pop dx
                int 21h
        loop sk_isv_2

        pop dx
        pop cx
        pop ax
ret

;TARPAS
tarpas:
        push ax
        push dx
        mov ah, 02h
        mov dl, ' '
        int 21h
        pop dx
        pop ax
ret

programa ends
        end start




;kiek raidziu "A" masyve t1
;kiek raidziu "B" masyve t2

stekas segment stack
          db 256 dup (?)
stekas ends

duomenys segment
vardas db 'Kursinio darbo uzduotis Nr. 5',13,10
       db 'Atliko: Aurimas Mameniskis',13,10
       db 'Grupe: II-0/1      2003 01 18',13,10,10
       db 'Raidziu A pirmajame tekste yra $'
t1 db 'k','o','p','A','s','A','u','A','s','A','*'
t2 db 'B','B','d','B','B','B','l','j','*'
```

```
tekst2 db 13,10,'Raidziu B antrajame tekste yra $'
duomenys ends

programa segment
 assume cs:programa, ds:duomenys, ss:stekas

start:
        mov             ax,duomenys
        mov             ds,ax
        mov ax, 0002h
        int 10h
;vardo isvedimas
      mov ah,09h
      lea dx,vardas
      int 21h
;skaiciavimas

      mov bh,0       ;skaitliukas masyvui t1
      mov bl,0       ;skaitl.mas.t2
      mov  si,0

cikl:   mov ch,t1[si]
      cmp ch,'*'
      je  fin        ;jeigu lygu
      cmp ch,'A'
      jne net        ;jei nelygu
      inc bh
      net:
         inc si
      jmp cikl       ;besaliginis perejimas
fin:
      mov si,0

cikl1:
      mov ch,t2[si]
      cmp ch,'*'
      je  fin1       ;jeigu lygu
      cmp ch,'B'
      jne net1       ;jei nelygu
      inc bl
      net1:
          inc si
      jmp cikl1      ;besaliginis perejimas
```

```
fin1:
;pirmos reiksmes isvedimas
     mov ah,02h
     mov dl,bh
     add dl,30h
     int 21h
;antras pranesimas ir reiksme
     mov ah,09h
     lea dx,tekst2
     int 21h

     mov ah,02h
     mov dl,bl
     add dl,30h
     int 21h

        mov             ah,4Ch
        int     21h



programa ends
end start



;UZDUOTIS 3.2: suskaiciuoti bendra balu vidurki.
;Masyvo pabaiga yra '***'.

;Steko segmentas
stekas segment stack
        db 256 dup(?)
stekas ends

cr equ 13
lf equ 10

;Duomenu segmentas
duomenys segment
        info db 'Kursinio darbo uzduotis Nr.3.2', cr, lf
            db 'Autorius: Aurimas Mameniskis, II-0/1', cr, lf, lf, '$'
       grp db 'Gr.:', '$'
```

```
        vard db 'Vardas:', '$'
        paz db 'Pazymiai:', '$'
        vid db 'Pazymiu suma:', lf, cr, lf, cr, '$'
        vardai db 1, '    Zigmas      ', 6, 5, 9,  8,  7,  9,  7, 6,  8,  8, 0
             db 2, '    Ignas       ', 6, 8, 5,  9,  8,  3,  9, 8,  5,  8, 0
             db 3, '    Paulius     ', 8, 6, 8,  6,  5,  6,  6, 5,  4,  8, 0
             db 4, '    Egle        ', 9, 7, 8,  8,  8,  4,  9, 9,  8, 10, 0
             db 5, '    Jonas       ', 8, 6, 8,  6,  5,  6,  6, 5,  4,  8, 0
             db 6, '    Vita        ', 9, 7, 8,  8,  8,  4,  9, 9,  8, 10, 0
             db "***"
        des dw 10
        eil db cr, lf, '$'
        pranb db 'Irasu nera!!!', cr, lf, '$'
        prang db lf, cr, 'Bendras vidurkis yra: ','$'
        suma dw ?
          pab db 'Pogramos pabaiga!!! Paspauskite bet kuri mygtuka...', cr,
lf, '$'
duomenys ends

;Programos segmetas
programa segment
      assume CS:programa, DS:duomenys, SS:stekas
      start: mov AX, duomenys
           mov DS, AX

;---Ekrano nuvalymas-----
         mov AX, 0002h
         int 10h

;---Kursoriaus nustatymas---
         mov AH, 02
         mov BH, 00h
         mov DH, 00h
         mov DL, 00h
         int 10h

         lea DX, info
         call isv

;---Pranesimu isvedimas---
         mov AH, 02        ;nustatome kursoriu
         mov BX, 00
         mov DH, 5  ;y
```

```
        mov DL, 0  ;x
        int 10h

        lea DX, grp
        call isv

        mov AH, 02        ;nustatome kursoriu
        mov BH, 00
        mov DH, 5
        mov DL, 6
        int 10h

        lea DX, vard
        call isv

        mov AH, 02        ;nustatome kursoriu
        mov BH, 00
        mov DH, 5
        mov DL, 19
        int 10h

        lea DX, paz
        call isv

        mov AH, 02        ;nustatome kursoriu
        mov BH, 00
        mov DH, 5
        mov DL, 42
        int 10h

        lea DX, vid
        call isv

      lea BX, vardai     ;isveda visa masyva
      xor CX, CX

;---Ieskome masyvo pabaigos pagal '***'---
tikrina:
      push BX
      cmp byte ptr [BX], '*'   ;tipo patikslinimas per ptr
      jne toliau               ;op1<>op2 persoka toliau
      inc BX
      cmp byte ptr [BX], '*'
```

```
        jne toliau
        inc BX
        cmp byte ptr [BX], '*'
        jne toliau
        pop BX
        jmp pabaiga

toliau:
        pop BX
        xor AX, AX
        mov AL, [BX]
        call sk_isv
        call tarp_isv
        inc BX
        push CX
        mov CX, 16
        mov DX, BX
        call eil_isv
        pop CX
        call tarp_isv
        add BX, 16

        push CX

        mov CX, 10
        xor AX, AX

;---Pazymiu isvedimas---
        cikl_pazymiai:
                        push AX
                       mov AL, [BX]
                        call sk_isv
                        call tarp_isv      ;tarpeliai tarp pazimiu
                        pop AX
                        add AL, [BX]
                        inc bx
        loop cikl_pazymiai

;---Sulyginam vidurkiu spausdinimo vietas---
        call tarp_isv             ;visus pastumiame
          call tarp_isv
        call tarp_isv
        call tarp_isv
```

```
        call tarp_isv
        call tarp_isv
        call tarp_isv
        call sk_isv
        call eilute

         mov [BX], AL

         mov CX, suma
         add AX, CX
         mov suma, AX

         inc BX

         pop CX
         inc CX
jmp tikrina

pabaiga:
        cmp CX, 0
        je blogai
        lea DX, prang
        call isv
        mov AX, 10
        mul CL
        mov CX, AX
        mov AX, suma
        xor DX, DX
        div CX
        call sk_isv
        call kablys
        mov AX, DX
        mul des
        div CX
        call sk_isv
        jmp gerai

blogai:
        lea DX, pranb
        call isv

gerai:
;---Kursoriaus nustatymas---
```

```
        mov AH, 02
        mov BH, 00h
        mov DH, 15h
        mov DL, 00h
        int 10h

        lea DX, pab
        call isv

;---Isejimas i DOS---
        mov AH, 07h
        int 21h
      mov AH, 4Ch
      int 21h

;---Skaiciu atspausdinimo aprasmas---
sk_isv:
      push AX
      push CX
      push DX
      xor CX, CX

      sk_isv_1:
            xor DX, DX
            div des
            add DX, 30h
            push DX
            inc CX
            cmp AX, 0
      jnz sk_isv_1

      mov AH, 02h             ;spausdina simboli
      sk_isv_2:
          pop DX
          int 21h
      loop sk_isv_2

      pop DX
      pop CX
      pop AX
ret

;---Atspausdina eilute---
```

```
eil_isv:
	push AX
	push BX
	push CX
	push DX

	mov AH, 02h

	eil_isv_1:
		mov DL, [BX]
		int 21h
		inc BX
	loop eil_isv_1

	pop DX
	pop CX
	pop BX
	pop AX
ret

;---Tarpu isvedimas---
tarp_isv:
	push AX
	push DX
	mov AH, 02h
	mov DL, ' '
	int 21h
	pop DX
	pop AX
ret

;---Atspausdina kablelius---
kablys:
	push AX
	push DX
	mov AH, 02
	mov DL, ','
	int 21h
	pop DX
	pop AX
ret

;---Nukelia i kita eilute---
```

```
eilute:
        push AX
        push DX
        mov AH, 09h
        lea DX, eil
        int 21h
        pop DX
        pop AX
ret

;---Pranesimo isvedimo aprasymas---
isv:
    push AX
    mov AH, 09h
    int 21h
    pop AX
ret

programa ends
         end start



stekas segment stack
         db 256 dup(?)
stekas ends

duomenys segment
                crlf db 13,10,'$'
          procent dq 1.03
    galut_suma dd 0
             suma dq 24.0
     skaitliukas dw 375
           simtas dq 100.0
           desimt dd 10
                ats db 11 dup(?),'$'
                aut db 'Autorius Aurimas Mameniskis II01',13,10,10,'$'
                dol db ' doleriu...',13,10,'$'
              pran db 'indenai po tiek metu gaus labai daug pinigu',13,10,'$'
duomenys ends

programa segment
```

```
        assume CS:programa, DS:duomenys, SS:stekas
        .386
start:
        mov AX, duomenys
       mov DS, AX
       mov ax, 0002h
       int 10h

       mov cx, skaitliukas
        fld suma
        jcxz toliau

ciklas:
        fmul procent
        loop ciklas

toliau:
        fmul simtas
        fistp galut_suma
        mov eax, galut_suma
        mov edx, 0
        mov si, 10

desimtaine:
        div desimt
        add dl,30h
        mov ats[si], dl
        dec si
        xor dx,dx
        cmp al, 0
        je pabaiga
        cmp si, 8
        je kablelis
        jmp desimtaine

kablelis:
        mov ats[si],2ch
        dec si
        jmp desimtaine

pabaiga:
        mov ah, 09h
        lea dx, aut
```

```
        int 21h

        mov ah, 09h
        lea dx, pran
        int 21h

        mov ah, 09h
        lea dx, crlf
        int 21h

        mov ah, 09h
        lea dx, ats
        int 21h

        mov ah, 09h
        lea dx, dol
        int 21h

        mov ah, 4ch
        int 21h

programa ends
end start
```