O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
FARG’ONA FILIALI

“ Kompyuter injiniringi ” fakulteti
“ Kompyuter tizimlari ” kafedrasi

« KOMPYUTER ARXITEKTURASI **»**
FANIDAN

(2 kurs bakalavriatura ta’lim yo’nalishlari uchun**)**

laboratoriya mashg’ulotlari uchun
USLUBIY KO’RSATMA

FARG’ONA - 2016

Ushbu o'quv uslubiy ko'rsatma O'zbekiston Respublikasi Oliy va O'rta Maxsus ta'lim vazirligining Oliy o'quv yurtlari boshqarmasi tasdiqlagan na'munaviy o'quv dastur va kafedrada ishlab chiqilgan ishchi o'quv dastur asosida tuzilgan.

O'quv uslubiy ko'rsatma 2-kurs «Kompyuter injiniringi» *bakalavr yo'nalishi talabalari uchun mo'ljallangan bo'lib, u 36 soatga rejalashtirilgan.*

Tuzuvchilar: *dots M.Djalilov*
ass SH.Abdullaev

Uslubiy ko'rsatma Kompyuter tizimlari kafedrasida muhokama etilgan (201_ yil «__» _____ -sonli bayonnoma).

Kafedra mudiri: _____ *dots, Djalilov M.*
(imzo) *(unvoni, F.I.SH.)*

Taqrizchi: _____ *dots, _____.*
(imzo) *(unvoni, F.I.SH.)*

O'quv Uslubiy ko'rsatma TATU Farg'ona filiali uslubiy kengashida ko'rib chiqilgan. (__-son yig'ilish bayoni __ _____ 201_ yil)

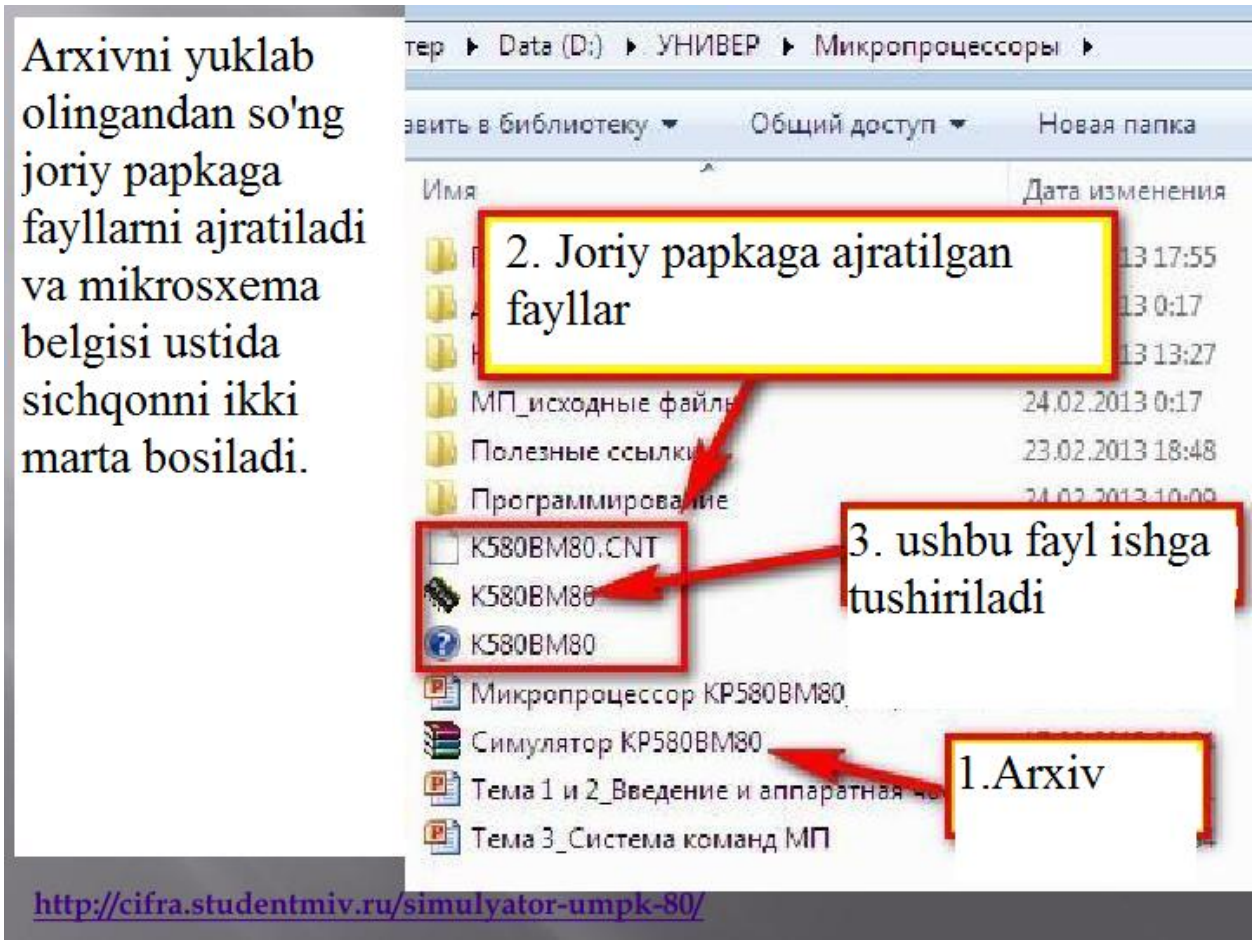
1-laboratoriya ishi

Kompyuterning bazaviy tuzilishini UMPK-80 elektron stendi yordamida o’rganish

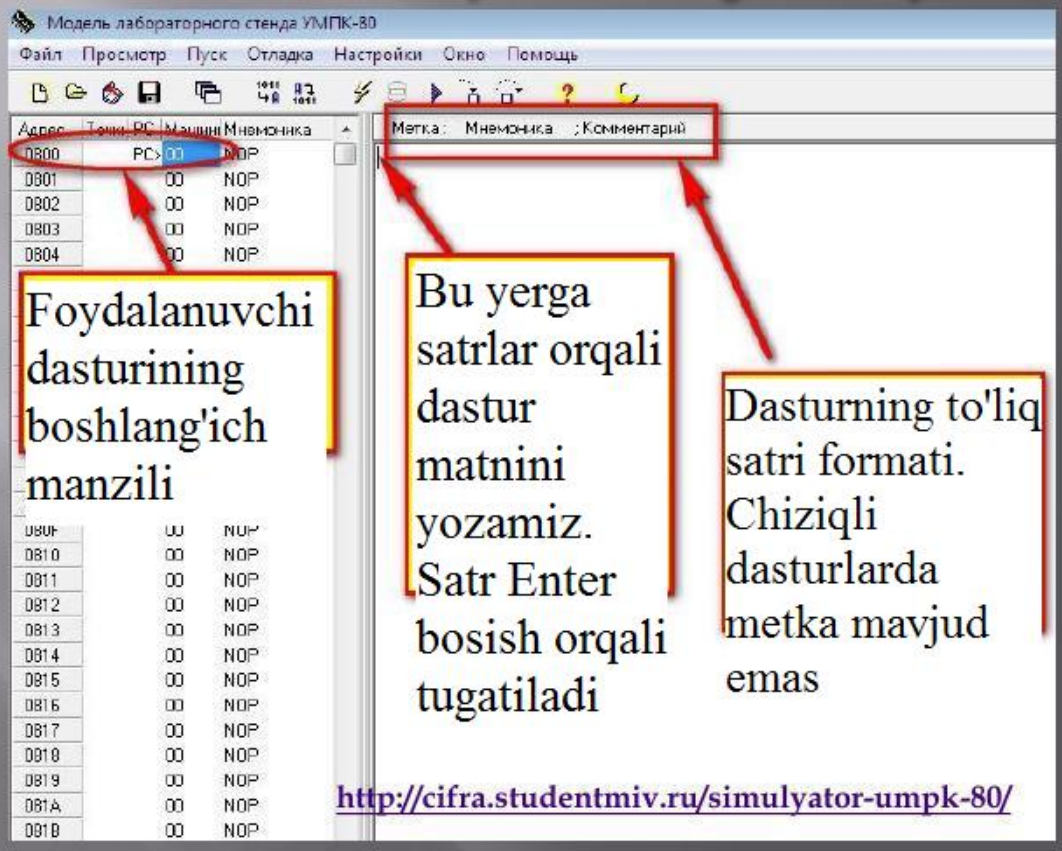
Ishdan maqsad: Kompyuterning bazaviy tuzilishini UMPK-80 elektron stendi yordamida o’rganish

NAZARIY QISM

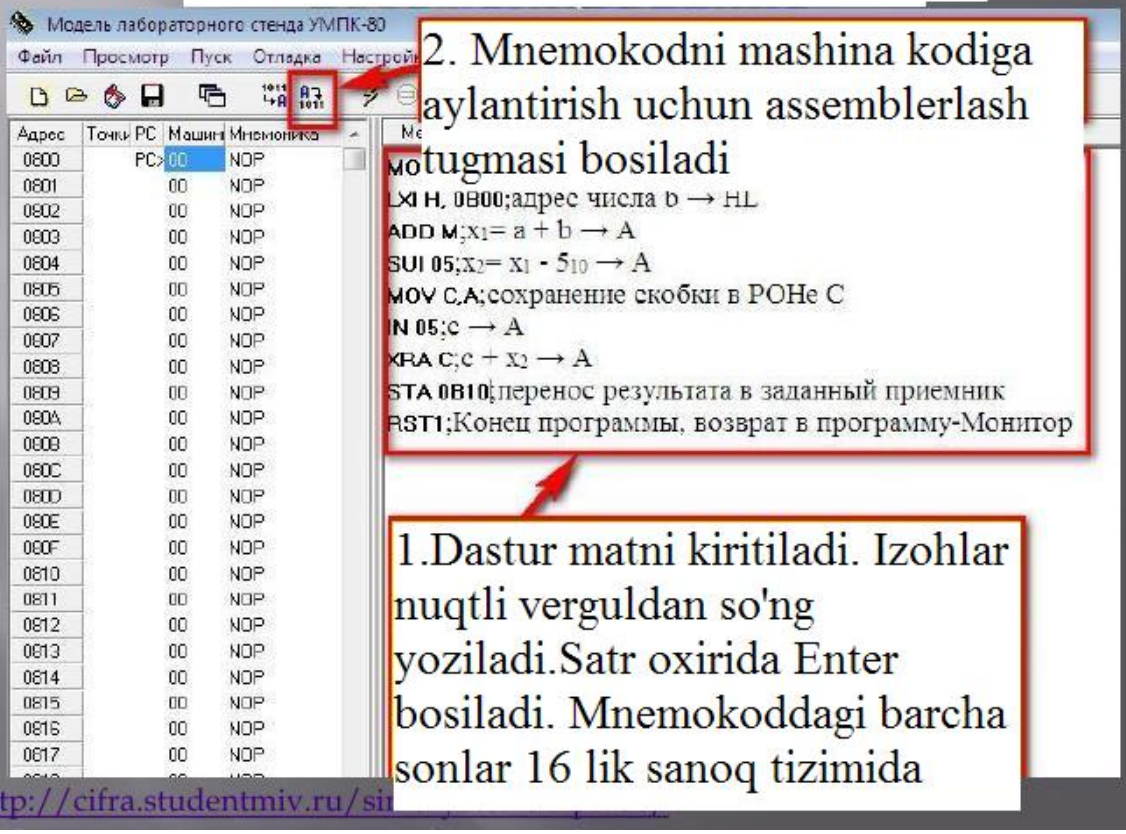
UMPK-80 stendi shaxsiy kompyuterning protsessorida dasturlarning bajarilishi hamda registrlar, xotira yacheykalarini tasavvur etish uchun qulay dastur hisoblanadi. Dasturni ishga tushirish va undan foydalanish quyida keltirilgan tartibda bo’ladi.



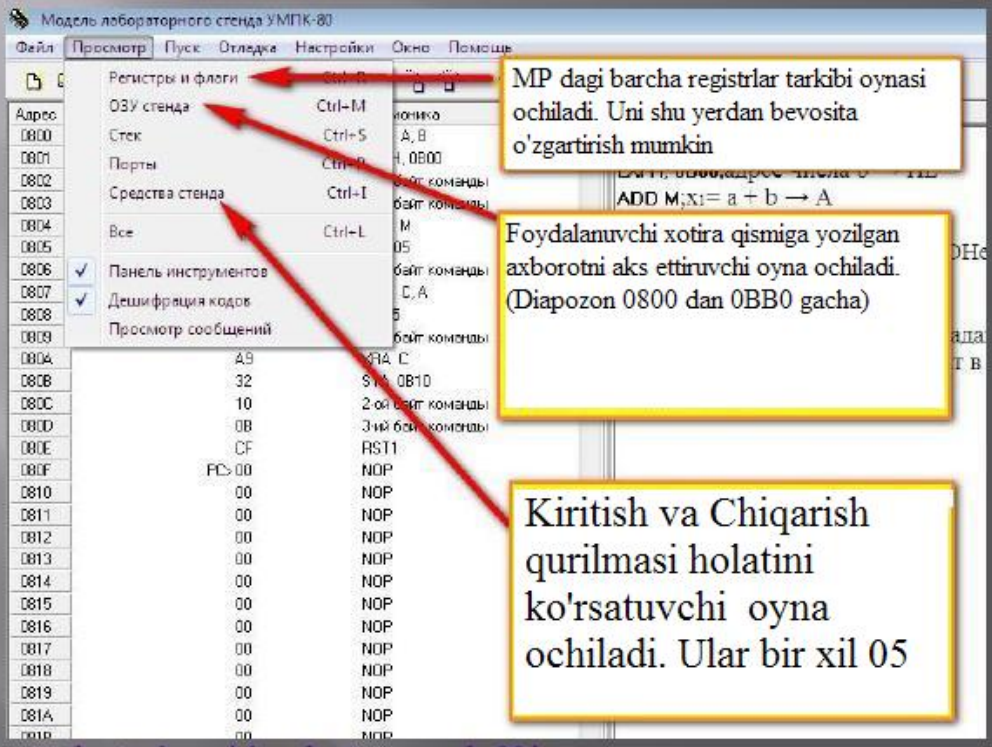
Elektron stendning ishchi ekrani



Dastur matnini yozish va assemblerlash

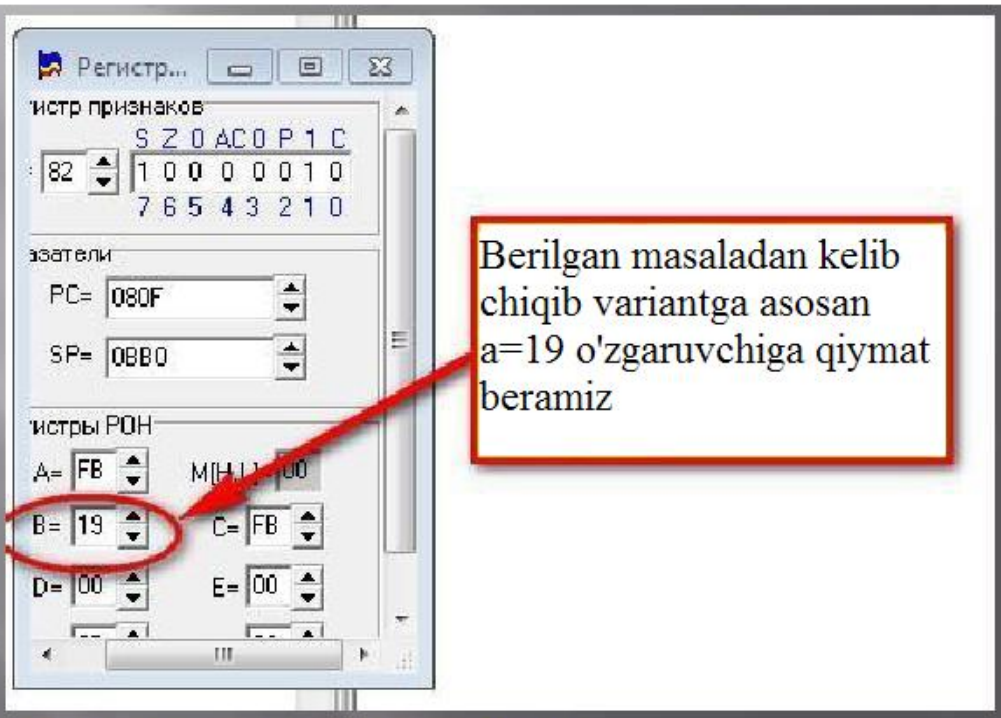


Registrlarga, xotira yacheykasiga ma'lumot kiritish va kiritish porti

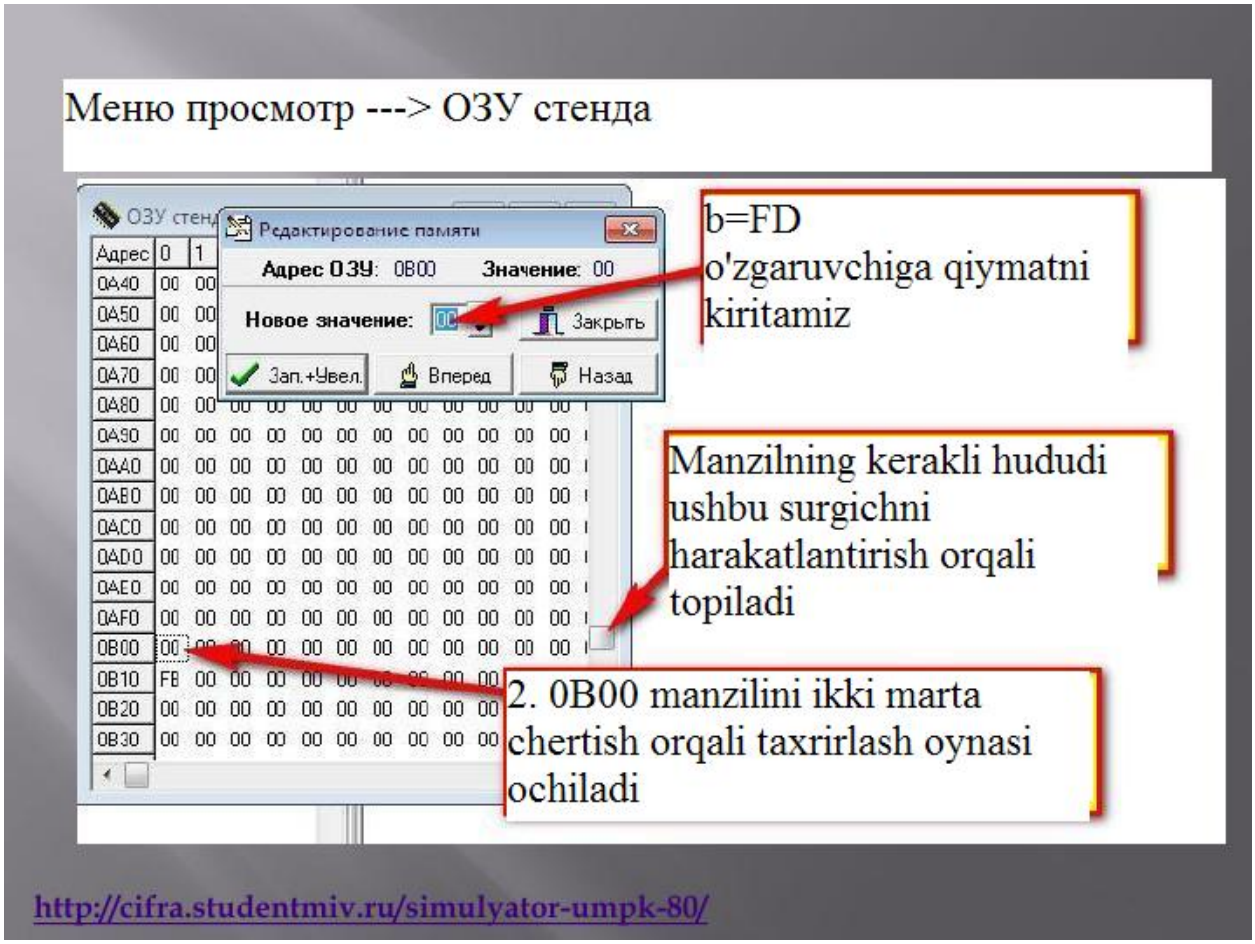


http://cifra.studentmiv.ru/simulyator-umpk-80/

Просмотр menyusi --> registrlar va bayroqlar



http://cifra.studentmiv.ru/simulyator-umpk-80/



NAZORAT SAVOLLARI

1. UMPK-80 stendining vazifasi nimadan iborat?
2. UMPK-80 stendining qanday imkoniyatlari mavjud?
3. UMPK-80 stendi menyularini sanang.
4. Registrlarni qiymatini o’zgartirish qanday usullari mavjud?

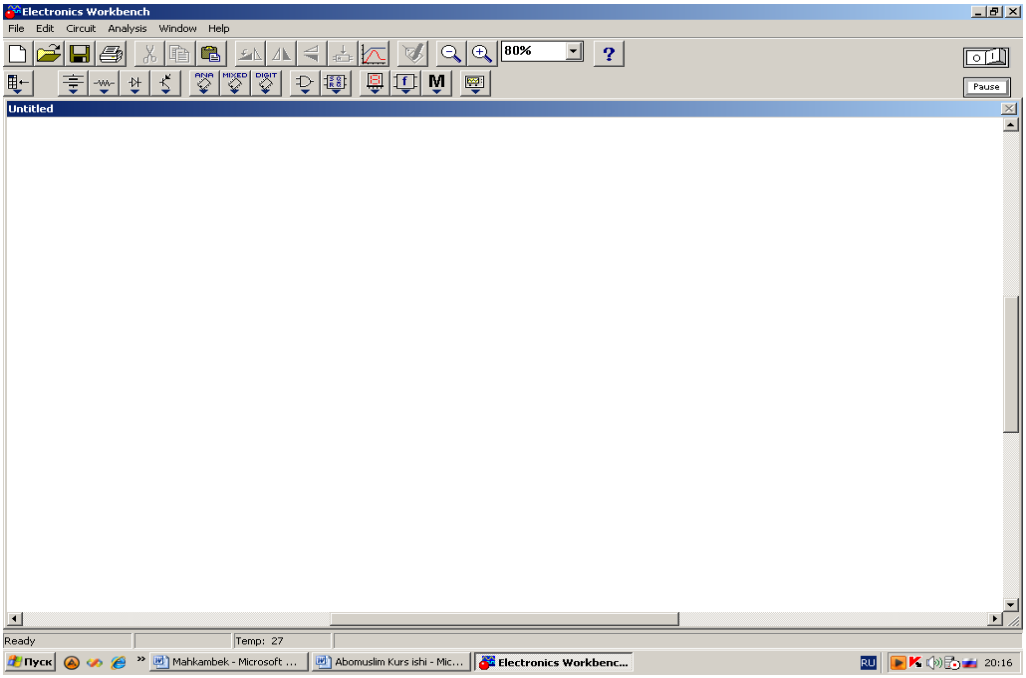
2-laboratoriya ishi

Elektronics workbench dasturidan foydalanish.

Ishdan maqsad: Elektronics workbench dasturidan foydalanib kombinatsion sxemalar qurishni o’rganish

NAZARIY QISM

Electronics Workbench dasturiga kirish uchun dastlab, Ishchi stolidan **Electronics Workbench** ob’ekti bo’lsa, ishga tushirishimiz mumkin yoki **Пуск>Программы> Electronics Workbench** ni tanlaymiz.
Electronics Workbench ishga tushiramiz u quyidagi ko’rinishga ega.



Bu dasturdan chiqish uchun fayl menyusidagi Exit (Alt-F4) tugmasi bosiladi yoki oynaning yuqori o’ng burchagidagi X tugmasi bosiladi. Dasturning asosiy oynasi 2-rasmda tasvirlangan, dastur ishlatuvchining standart oynali interfeysiga ega.
Меню қисми 6 га бўлинади.
1. File
2. Edit
3. Circuit
4. Analysis
5. Windows
6. Help
1. File bo’limi.
New — Yangi oyna ochish
Open — Tayyorlangan fayllarni ochish
Save — Ma’lumotni saqlash
Save as—Ma’lumotni qayta nom bilan saqlash

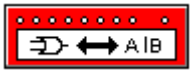
Import — Qabul qilish
Export — Jo’natish
Print — Ma’umotni chop qilish
Print setup — Print setapi
Exit — Dasturdan chiqish
2.Edit bo’limi
Cut — Bir ob’ektni qirqish
Copy — Nusxa olish
Paste — Urnatish, qo’yish
Delete — O’chirish
Select All—Blokka olish

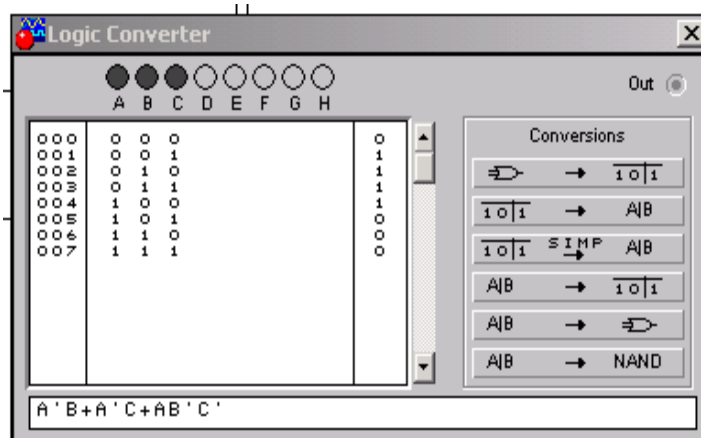
 yangi oyna ochish.

 saqlangan fayllarni ochishimiz mumkin,bu ob’ekti sichqoncha yordamida bir marta chertsak quyidagicha oyna hosil bo’ladi. Ойнадан бизга керакли бўлган Electronics Workbench

 дастурига файллар оид саqланган Mavjud очиладию kengaytma bilan sxemani .EWB saqlash

Oynadan  tugmasini tanlaymiz va ish stoliga olib o’tamiz.

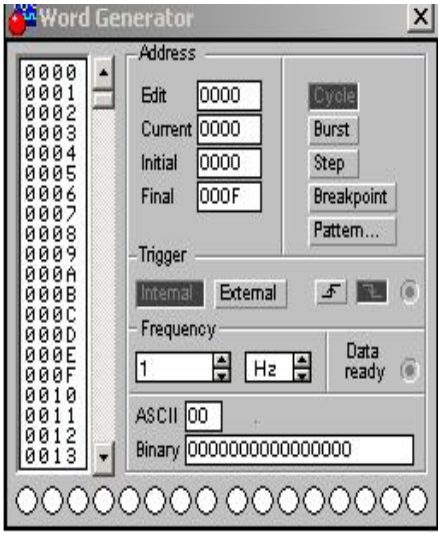
Logic Convertorni  sichqoncha yordamida 2 marotaba chertamiz. Chinlik jadvalini kiritamiz.So’ng chinlik



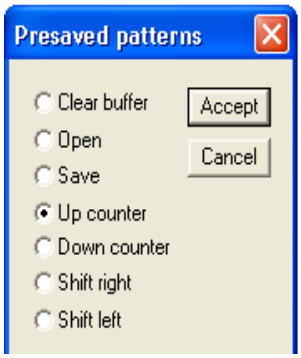
jadvalining o’ng tomonida joylashgan qatorga chin qiymatlarni kiritamiz. Oynadan  yoki  tugmalardan birini bosamiz. Natijada pastki oynada soddalashitrilgan formulaga ega bo’lamiz.

Electronics Workbench dasturi yordamida formulaga mos ravishda ”EMAS” , “VA”, ”YoKI” elementlari yordamida sxema yig’ishimiz uchun  tugmani bosamiz . ”EMAS” , “VA” , ”YoKI” elementlari yordamida yigilgan sxema ishchi oynada paydo bo’ladi .  tugmasini olib ishchi stolga joylashtirib generatorni ulaymiz.

Sichqoncha  yordamida 2 marotaba chertsak Word Generator oynasi ochiladi.
Oynaning to’xtash parametri kiritiladi :

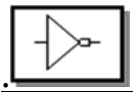


 tugmasi bosiladi . Bizga Prisaved patterns oynasi ochiladi, oynadan Up Counter bandi tanlanib Accept tugmasi bosiladi :



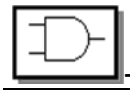
Chastotasi 1 gersga sozlanib. Indicators oynasidan is’temolchi lampa tanlanib sxemaga ulaymiz.

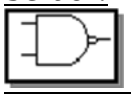
Soddalashtirilgan mantiqiy formulani realizastiya qilish

 -mantiqiy INKOR(emas) elementi hisoblanib,agar biz kiritgan “1” signalni “0” qilib beradi yoki aksincha.

 -mantiqiy YOKI elementi.Bu elementni ikkita kirishi va bitta chiqishi mavjud bo’lib, unga kiritilgan ikkita mantiqiy o’zgaruvchilarni dizunkstiyasini hosil qilib beradi.

 -mantiiy YOKI EMAS elementi.Bu element ham YoKI elementiga o’xshagan bo’lib lekin dizunkstiyasini inkorini beradi.


- mantiqiy VA elementi.Bu elementni ikkita kirishi va bitta chiqishi mavjud bo’lib,unga kiritilgan mantiqiy o’zgaruvchilarni konyuksiyasini hosil qilib beradi.

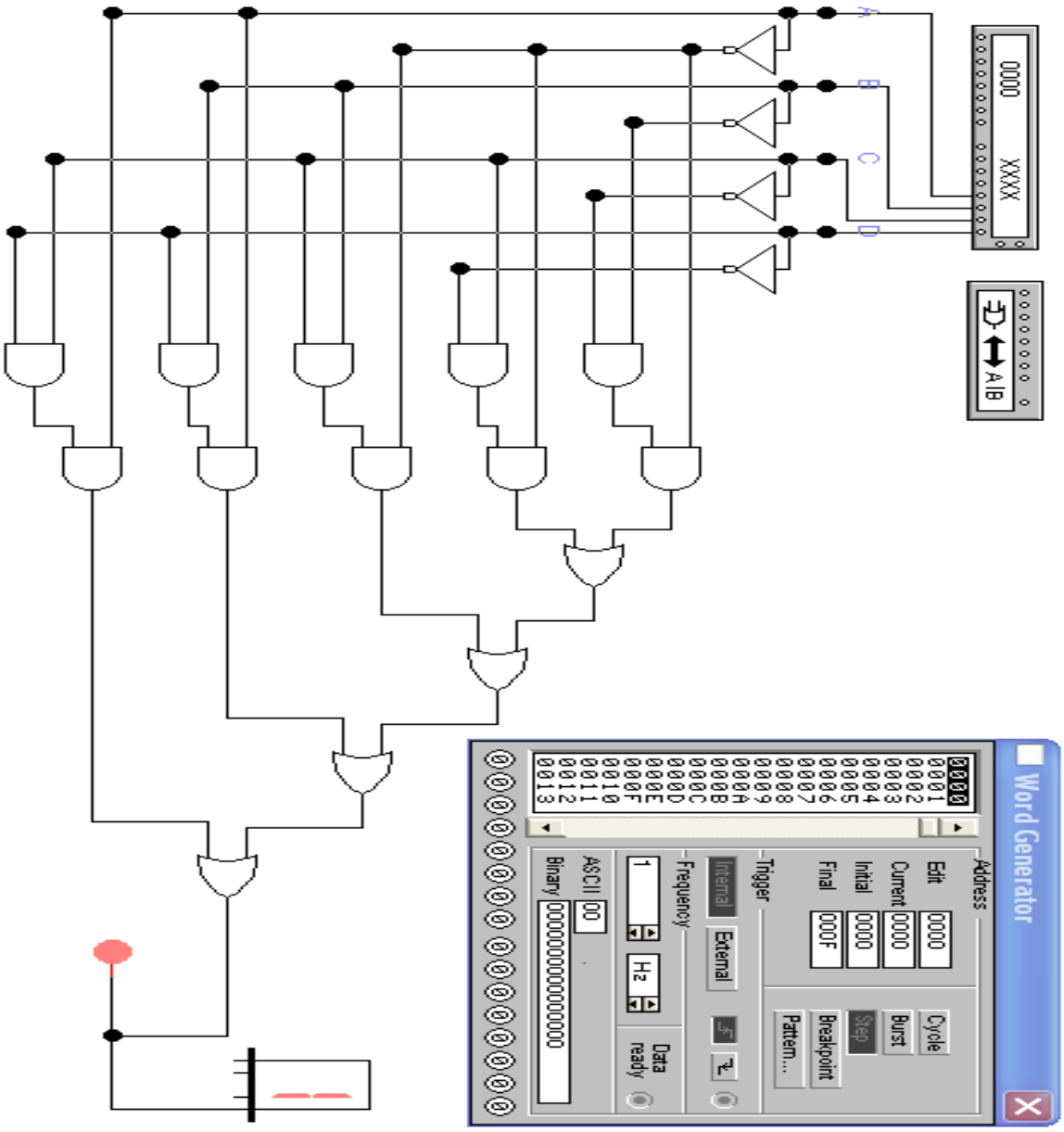

-mantiiy VA EMAS elementi.Bu element ham Va elementiga o’xshash bo’ladi lekin konyukstiyasini inkorini hosil qilib berad

AMALIY QISM

1. **EWB dasturini ishga tushiramiz**
2. **Berilgan funksiyalarni soddalashtiramiz**
3. **Sxemasi quriladi**

Electronics workbench dasturidan foydalanib sxemalarni qurish

Birinchi bazis “va”, “yoki”, “inkor” bo’yicha qurilgan sxema.

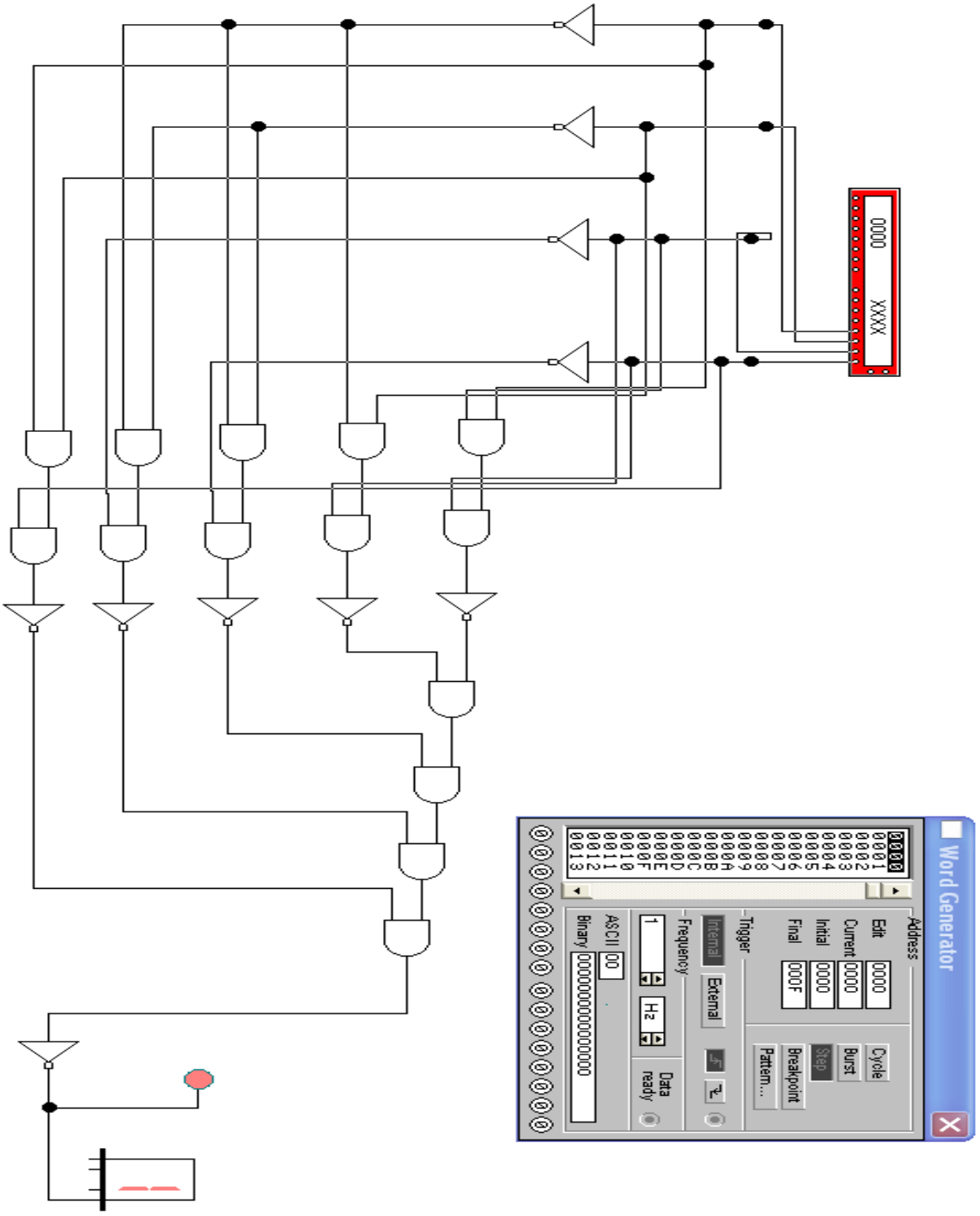


F(x)=ACD+A’BC+ABD+A’B’D’+A’B’C’

Kombinastion sxemani bunday usulda yig’ish aralash (INKOR,VA ,YOKI) usul orqali amalga oshadi. Buning uchun Electronics Workbench dasturidagi uskunalar panelidan tugmachani bosamiz.Xosil bo’lgan oynadan

 tugmani tanlab, ish stoliga olib o’tamiz va ikki marta bosamiz. Bunda Logic Convertor muloqot oynasi ochiladi.Unga chinlik jadvalini kiritamiz.Aralash kombinastion sxemani tuzish uchun  tugmani bosamiz.Natijada ekranda yuqoridagi sxema xosil bo’ladi. Sichqoncha yordamida 2 marotaba chertib Word Generator oynasini ochdim va oynani to’xtash parametrini kiritdim.  tugmasini bosib hosil bo’lgan oynadan Up Counter bandi tanlanib Accept tugmasini bosdim. Chastotani 1 gerstga sozlab Indicators oynasidan is’temolchi lampa tanlab sxemaga uladim. Indicators oynasidan is’temolchi lampa tanlab sxemaga uladim va uskunalar panelidan  tugmasini olib ishchi stolga joylashtirib generatorni uladim. Schyotchikni ulab to’g’ri ishlashini tekshirdimAralash usulda 3 ta INKOR, 4 ta VA elementi, 2 ta YoKI elementlari mavjud.

<u>Uchinchi bazis “va”, “inkor” bo’yicha qilingan sxema.</u>

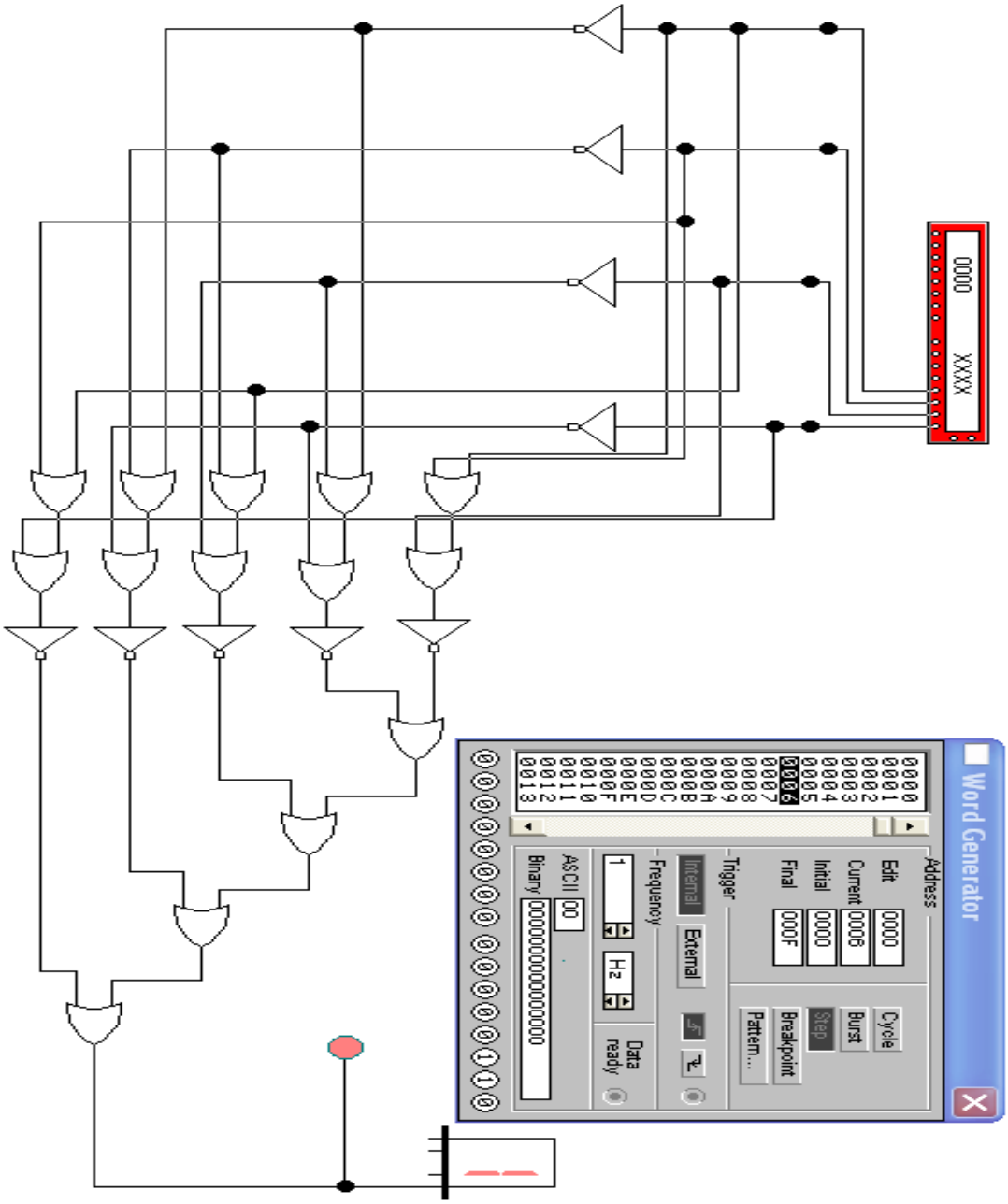


F(x)=A*C*D+A’*B*C+A*B*D+A’*B’*D’+A’*B’*C’=
=((A*C*D)’+(A’*B*C)’+(A*B*D)’+(A’*B’*D’)’+(A’*B’*C’)’)’

To’rtinchi bazis “yoki” bo’yicha qilingan sxema.

F(x)=A*C*D+A’*B*C+A*B*D+A’*B’*D’+A’*B’*C’=

=(A+C+D*)’+(A+B’+C’)’+(A’+B’+D’)’+(A+B+D)’+(A+B+C)’;



NAZORAT SAVOLLARI

1. EWB dasturi qanday ishga tushiriladi
2. Mantiqiy funksiyalarning qanday usullar yordamida ifodalash mumkin
3. MAF larini soddalashtirish usullarini ayting
4. Karno kartasi yordamida MAF larini soddalashtirishni tushuntiring

3-laboratoriya ishi

Assembler tilida dasturlash asoslari va undagi buyruqlarining tuzilishlari bilan tanishish.

Ishdan maqsad: Assembler tilida dasturlash asoslari va undagi buyruqlarining tuzilishlari bilan tanishish

NAZARIY QISM

1.KIRISH. Assembler tili haqida

Assembler tilida dasturlarni yezish uchun kompyuterning barcha tarmoklarning tuzilishlarini bilish kerak. Kompyuterning asosida bit va bayt tushunchalari mavjud. Ular yordamida kompyuterning xotirasidagi komandalar va ma'lumotlar kursatiladi.

Dasturlar mashinaviy kodlar kurinishida ma'lumotlarni aniklash segmentlaridan tashkil topgan. Bu segmentlar mashina buyruklari va adreslarini saklovchi steklardir. Arifmetik amallarni, ma'lumotlarni yuborish va adreslash uchun kompyuterda bir necha registrlar mavjud.

Birinchi kompyuterlarni yaratilishi davridan boshlab xozirgi kungacha dasturchilarni assambler tili xaqidagi qarashlarini qiziqarli tarzda kuzatish mumkin.

Qachonlardir assembler tilini yaxshi bilmasdan kompyuterni biron bir foydali ish qilishga majbur qilib bo'lmas edi. Vaqt o'tishi bilan xolat o'zgarib, kompyuter bilan muloqot qilishning zamonaviy, qulay vositalari vujudga keldi. Lekin boshqa tillardan farqli xolda, assembler tili o'lmadi.

Assembler tili - bu mashina tilining simvolik ko'rinishidir. Mashinaning quyi apparat darajasidagi hamma jarayonlar mashina tili buyruqlari yordamida amalga oshiriladi.

Bundan ko'rinadiki umumiy nomlanishdan qat'iy nazar, xar bir turdagi kompyuterning assembler tili mavjud.

Kompyuter apparat qismiga bog'liq bo'lgan muammolarni assembler tilini bilmasdan hal etish juda qiyin. Dasturchi yoki foydalanuvchi boshqa ixtiyoriy yuqori darajali dasturlash vositalardan foydalanib turli xil vazifalarni bajarishi mumkin. Kompyuter tizimining to'la zaxirasidan foydalanish uchun adreslash rejimini va mikroprotsessor buyruqlarini bilish talab etiladi.

Adreslash rejimi shunday usullarni ko'rsatadiki, uning yordamida dastur buyruqlariga va ma'lumotlarga yo'l ochiladi. Xamma yuqori darajali dasturlash tillarining kompilyatorlarida yoki translyatorlarida dasturda assembler muxitiga chiqish uchun o'zining modellari mavjud.

Assembler tilida dasturlash uchun kompyuterning ishlash tamoyilini va uning arxitekturasini bilish talab etiladi.

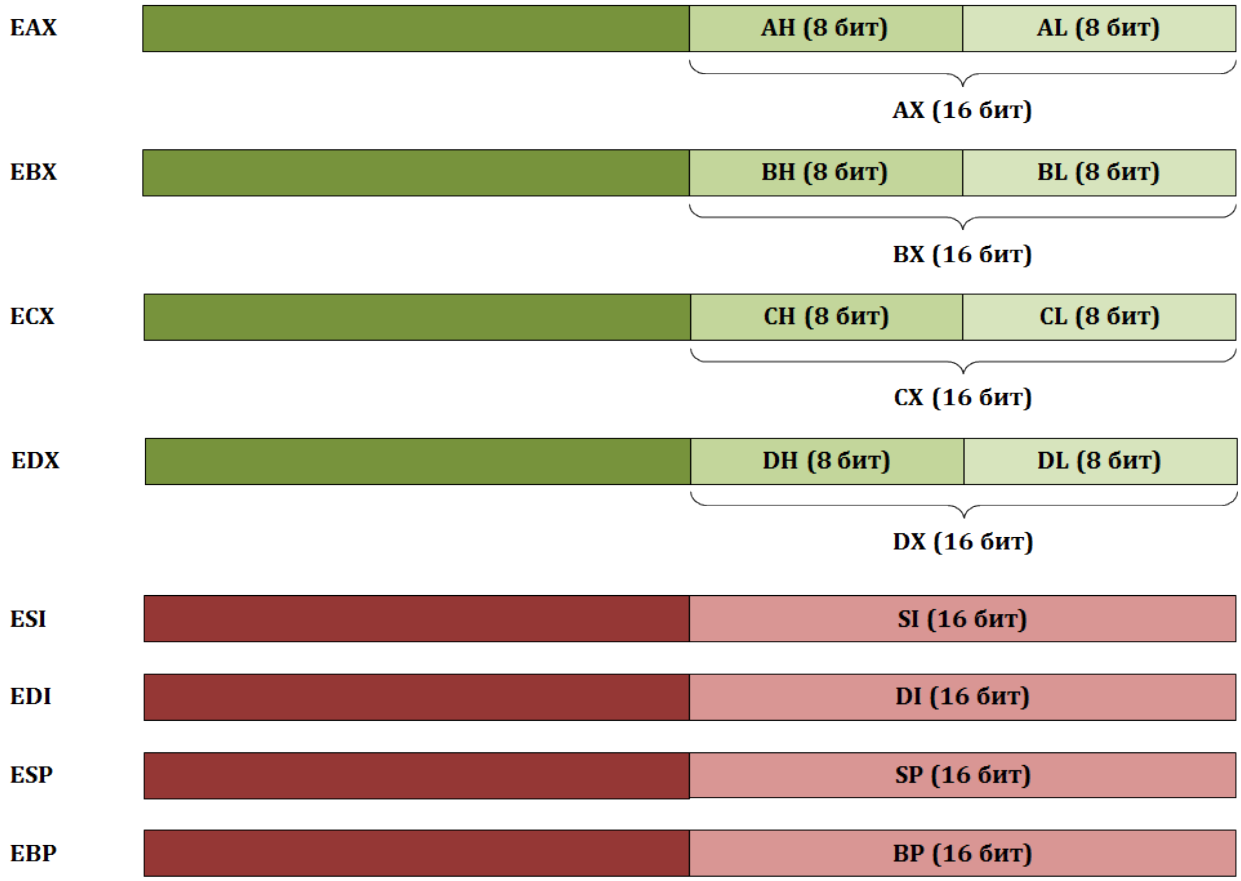
Registrlar

Registrlar – bu maxsus xotira yacheykasi bo'lib, bevosita protsessorda joylashadi. Registrlar bilan ishlash operativ xotiraga nisbatan sezilarli darajada tez bo'ladi, shuning uchun registrlar assemblerda hamda yuqori darajadagi dasturlash tillari kompilyatorlarida keng foydalaniladi.

Registrlarni quyidagi guruhlarga ajratish mumkin: umumiy foydalanish registrlari, buyruqlarni ko'rsatuvchi, bayroq registrlari va segment registrlari.

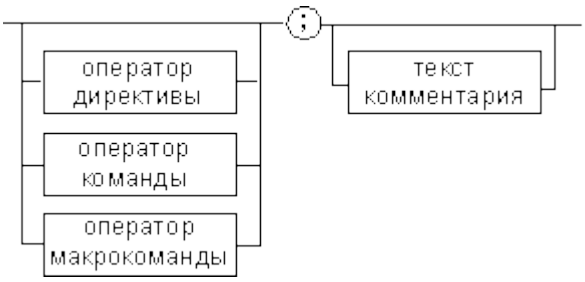
UMUMIY FOYDALANISH REGISTRLARI

Umumiy foydalanish registrlariga 8 registrdan tashkil topgan guruh kiradi. Ularni assembler tilidagi dasturlarda foydalanish mumkin. Barcha registrlar 32 bitli bo'lib, ular ikki va undan ortiq bo'laklarga bo'lishi mumkin.

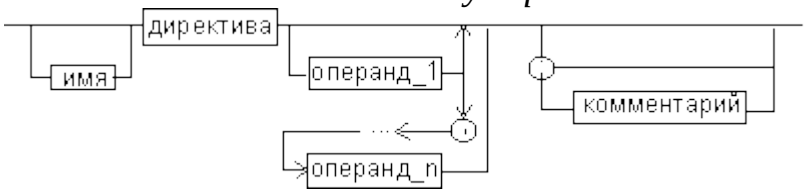


Registrlarning nomlanishi ularning vazifasidan kelib chiqqan:

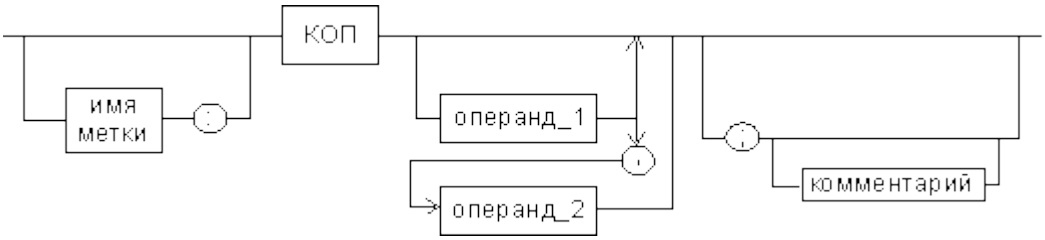
- **EAX/AX/AH/AL** (*accumulator register*) – akkumulyator;
- **EBX/BX/BH/BL** (*base register*) –baza registri;
- **ECX/CX/CH/CL** (*counter register*) – sanagich;
- **EDX/DX/DH/DL** (*data register*) – ma'lumotlar registri;
- **ESI/SI** (*source index register*) – manba indeksi;
- **EDI/DI** (*destination index register*) – qabul qiluvchi indeksi;
- **ESP/SP** (*stack pointer register*) – stekni ko'rsatuvchi registri;
- **EBP/BP** (*base pointer register*) – baza stek kadri ko'rsatuvchi registr



rasm. 1. *Assemblerda buyruqlar Formati*



Rasm. 2. *Direktivalar Formati*



Rasm. 3. *komanda va makrokomandalar Formati*

* Arifmetik operatorlar. Bular quyidagi:
unar “+” va “–”;
binar “+” i “–”;
ko’paytirish “*”;
bo’lish “/”;
bo’lishdan qoldiq olish “mod”.

Assemblerda ma’lumotlarni zaxiralash va e’lon qilish direktivalari

Initsializatsiya	**Uzunligi (bit)**	**O’qilishi**
DB	8	Define Byte
DW	16	Define Word
DD	32	Define Double Word
DF	48	Define Six Bytes
DP	48	Define Six Bytes
DQ	64	Define Quarter Word
DT	80	Define Ten bytes

•

db — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 1 bayt.
Direktiva **db** da da quyidagi qiymatlarni berish mumkin:

- o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
  - belgili tipdagi sonlar uchun–128...+127;
  - belgisiz tipdagi sonlar uchun0...255;

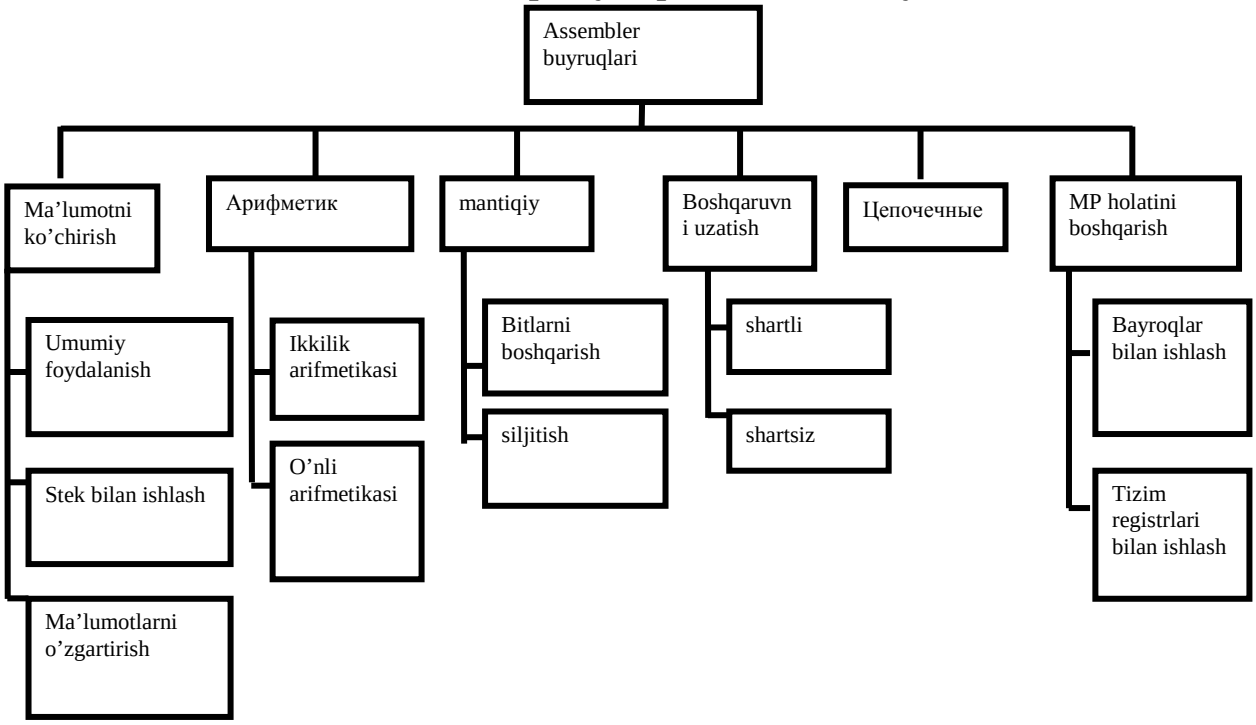
- **dw** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 2 bayt.
  Direktiva **dw** da da quyidagi qiymatlarni berish mumkin:
  - o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
    - belgili tipdagi sonlar uchun–32 768...32 767;
    - belgisiz tipdagi sonlar uchun0...65 535;

- **dd** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 4 bayt.
  Direktiva **dd** da da quyidagi qiymatlarni berish mumkin:
  - o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
    - i8086 uchun:
      - belgili tipdagi sonlar uchun–32 768...+32 767;
      - belgisiz tipdagi sonlar uchun0...65 535;
    - i386 va yuqorilar uchun:
      - belgili tipdagi sonlar uchun–2 147 483 648...+2 147 483 647;
      - belgisiz tipdagi sonlar uchun0...4 294 967 295;

- **df** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 6 bayt;
- **dp** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 6 bayt.
  Direktivami **df** i **dp** da da quyidagi qiymatlarni berish mumkin:
  - o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
    - i8086 uchun:
      - belgili tipdagi sonlar uchun–32 768...+32 767;
      - belgisiz tipdagi sonlar uchun0...65 535;
    - i386 va yuqorilar uchun:
      - belgili tipdagi sonlar uchun–2 147 483 648...+2 147 483 647;
      - belgisiz tipdagi sonlar uchun0...4 294 967 295;
-
- **dq** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 8 bayt.
  Direktiva **dq** da da quyidagi qiymatlarni berish mumkin:
  - o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
    - MP i8086 uchun:
      - belgili tipdagi sonlar uchun–32 768...+32 767;
      - belgisiz tipdagi sonlar uchun0...65 535;
    - MP i386 va yuqorilar uchun:
      - belgili tipdagi sonlar uchun–2 147 483 648...+2 147 483 647;
      - belgisiz tipdagi sonlar uchun0...4 294 967 295;

- **dt** — quyidagi o’lchamdagi ma’lumot uchun xotirani zahiralash 10 bayt. Direktiva **dt** da da quyidagi qiymatlarni berish mumkin:
    - o’zgaruvchi yoki konstantani qabul qilish qiymatlari diapozoni:
        - MP i8086 uchun:
            - belgili tipdagi sonlar uchun–32 768...+32 767;
            - belgisiz tipdagi sonlar uchun0...65 535;
        - MP i386 va yuqorilar uchun:
            - belgili tipdagi sonlar uchun–2 147 483 648...+2 147 483 647;
            - belgisiz tipdagi sonlar uchun0...4 294 967 295;

Ассемблер buyruqlari klassifikatsiyasi



Buyruqlar

Assembler tili Buyruqlari – bu mashina buyruqlarining simvolli shakli Buyruqlar quyidagi sintaksisga ega:

[<metka>:] <mnemokod> [<operandlar>] [;<izoh>]

Ma’lumotlarni ko’chirish buyruqlari:

- **MOV <operand qabul qiluvchi>, <operand manba>**
- **XCHG <operand1> ,<operand2>**

MOV – Ma’lumotlarni ko’chirishning asosiy buyrug’i

Foydalanish qoidalari:

- Xotiraning bir qismidan boshqasiga ko’chirishda ishlatilmaydi. Bunday holda umumiy foydalanish registrlaridan foydalanish kerak.
- Segment registriga xotiradan ko’chirishni amalga oshirib bo’lmaydi.

- Bir segment registrdagi ma’lumotni boshqa bir segment registriga o’tkazib bo’lmaydi
- **CS** segment registrini qabul qiluvchi sifatida ishlatib bo’lmaydi

XCHG <operand1> ,<operand2>

XCHG – ma’lumotlarni ikkitomonlama ko’chirish buyrug’i

Foydalanish:

- Operandlar bir xil tip yoki o’lchamga ega bo’lishi zarur
- Xotiraning bir qismidan boshqasiga ko’chirishda ishlatilmaydi. Bunday holda umumiy foydalanish registrlaridan foydalanish kerak

MASALAN:

```
xchg   eax , ebx
xchg   eax , dword ptr [ebx]
```

NAZORAT SAVOLLARI

1. Assembler tili qanday til?
2. Assembler tilida asosan qanday ahamiyatga ega dasturlar tuziladi?
3. Registr nima?
4. Registr turlari
5. Umumiy foydalanish registrlarini sanang
6. Qanday tiplarni bilib oldingiz?

4-laboratoriya mashg’uloti

Assembler tilida oddiy dasturlarni yozish va bajarish.

Ishdan maqsan: Assembler tilidagi buyruqlar bilan tanishish va oddiy dasturlarni yozishni o’rganish.

NAZARIY QISM

Assemblerda quyidagi umumiy maqsadli registrlar keng foydalaniladi.

1) AX registri. AX registri tallagichdir. U xamma kiritish va chiqarish operatsiyalari va ba’zi satrlarni ustidagi operatsiyalar uchun ishlatiladi. Masalan, ko’paytirish va bo’lish, siljish komandalar AX registrni ishlatishadi. Ba’zi buyruqlar, agar AX registrga murojat qilishsa, samaraliroq kodiga ega bo’ladilar:

AX: ! AH ! AL !

2) BX registri. BX registri tayanch (baza) registridir. Bu birgina, “indeks” sifatida kengaytirilgan adreslashtirish uchun umumiy maqsadli registridir.

BX: ! BH ! BL !

3) CX registri. CX registri schetchikdir. U siklarni qaytarilishini boshqarish, chapga va unga siljish operatsiyalari uchun ishlatiladi. CX registri xisoblash uchun xam ishlatiladi.

; dan so’ng izohlarni yozish mumkin

mov qayerga , qayerdan – bu joylashtirish buyrug’i

mul _d – bu ex registrni _d ga ko’paytirish buyrug’i

natija ax ga joylashadi

shl edx,16 – 16 razryadga surish buyrug’i

div cx - ax ni cx ga bo’lish buyrug’i. natija ax ga joylashadi

pop ecx - bu buyruq stekdan qiymatni oladi

sub ecx,eax – bu buyruq ecx dan eax ni ayiradi. Natija ecx ga joylashadi

ISHNI BAJARISH TARTIBI

1. **Assembler tilida dasturlash uchun qutidagi dasturlar kerak bo’ladi:**

MASM_v9.0 yoki MASM_v10.0 dasturi
natijalarni ko’rish uchun: OLLYDBG.EXE

2. MASM dasturi o’rnatilgan joyda aaa.bat nomli fayl yaratiladi

Faylga bunday nom berishdan maqsad uning barcha fayllardan yuqorida joylashishi va oson topilishini ta’minlash. Aaa.bat faylga quyidagi axborot kiritiladi.

ml /c /coff "work.asm"
link /SUBSYSTEM:CONSOLE "work.obj"

work.asm – bu kompilyatsiyalanadigan dastur nomi. Ushbu ma’lumot kiritilgach va uni saqlab dasturlashga kirishish mumkin

3. Assembler quyidagilarga ega :

- protsessor tipini aniqlash direktivasi,
- dastur boshlanish belgisi,
- dastur tanasi,
- dastur yakuni belgisi

4. Assembler tilida turli o’zgaruvchilar tipi mavjud: belgili va belgisiz formatdagi tiplar Shortlnt (signed char), Byte (unsigned char), Integer (int), Word (unsigned int) va boshqalar.

5. Quyidagi ifodani hisoblash dasturini yozsak: **a – e/b – de**, bu yerda:

a = 5;
b = 27;
c = 86;
e = 1986;
d = 1112;

aaa.bat: work.asm joylashgan yerga uni saqlaymiz,. Agar biz boshqa dasturni kompilyatsiya qilmoqchi bo’lsak unda aaa.bat ichida yozilgan work fayl nomini boshqasiga almashtirish kerak bo’ladi. Uni saqlash talab etiladi. Agar dastur sintaktik xatolarsiz bo’lsa unda exe kengaytmali fayl hosil bo’lishi kerak

6. dastur:

```
.686 ;  protsessor tipini aniqlash direktivasi

.model flat, stdcall ; chiziqli xotira modelini e’lon qilish

Option case map:none ;  Windows OT bilin moslashtirish

.data ; ma’lumotlarni aniqlash direktivasi

_a dw 5 ;  16-razryadli xotira omboriga _a nom bilan 5 sonini yozish
```

```
_b dw 27 ; yozish _b = 16h
_c dw 86 ; yozish _c = 56h
_e dw 1986 ; yozish _e = 7c2h
_d dw 1112 ; yozish _d = 458
res dw 0 ; res o’zgaruvchisini saqlash uchun xotirani zaxiralash

.code ; buyruqlar segmenti boshlanishi direktivasi

start:
mov edx,0 ; registrlarni tozalash
mov ebx,0
mov ecx,0
mov ax, _e ; ax registriga _e = 7c2h sonini joylashtiramiz
mul _d ;  _e  ni _d  ga ko’paytiramiz
SHL edx, 16 ;  16 ga ko’chirishni amalga oshiramiz

mov dx,ax
push edx ;  qiymatni stekka tashlaymiz

mov edx,0
mov ax, _e
mov cx, _b
div cx ; ax ni cx ga bo’lamiz
pop ecx ;  stekdan qiymatni olamiz
sub ecx, eax ; ayiramiz
mov ax, _a
sub eax, ecx
mov res, eax
ret ; OT boshqaruviga qaytaramiz

end start ; start nomli dasturni yakunlash
```

Nazariy topshiriqlar:

1. Qanday buyruqlarni bilib oldingiz?
2. Registr va ularni vazifalarini tushuntiring?
3. Foydalanuvchi registerlarga qaysi registrlar kiradi?
4. Segment registrlari va ularning vazifalarini tushuntiring?

5-Laboratoriya ishi

Oddiy kiritish-chiqarish qurilmalari bilan axborot almashinishni, niqoblashning dasturiy usullarini va shartli o’tishlarini tashkil qilishlarni o’rganish.

Ishdan maqsad: Oddiy kiritish-chiqarish qurilmalari bilan axborot almashinishni, niqoblashning dasturiy usullarini va shartli o’tishlarini tashkil qilishlarni o’rganish

NAZARIY QISM

O’tishlar va sikllar

Assembler tilida buyruqlarning bajarilish ketma-ketligini o’zgartirish uchun shartli va shartsiz o’tish hamda siklni boshqarish buyruqlaridan foydalaniladi. Barcha buyruqlar bayroqlarni o’zgartirmaydi

Shartsiz o’tish

shartsiz o’tish buyrug’i quyidagicha sintaksisga ega:

JMP <operand>

Operand o’tish manzilini ko’rsatadi. Ushbu manzilni ko’rsatishning ikki xil usuli mavjud. bevosita va bilvosita o’tish

Bevosita o’tish

Agar o’tish buyrug’ida o’tish kerak bo’lgan buyruq metkasi ko’rsatilgan bo’lsa, bu o’tish bevosita o’tish deyiladi.

```
  jmp   L
  ...
L: mov   eax, x
```

Bilvosita o’tish

Bilvosita o’tishda buyruq tarkibida o’tish manzili emas, balki registr yoki xotira yacheykasi ko’rsatiladi.

Ko’rsatilgan registr yoki xotira yacheykasi mutloq o’tish manzili sifatida qaraladi. Bilvosita o’tish o’tish manzili faqatgina dastur ishlash vaqtida aniq bo’lgan hollarda foydalaniladi

```
jmp   ebx
```

Taqqoslash va shartli o’tish buyruqlari

Shartli o’tish buyruqlari ma’lum bir shartning rost bo’lgan hollarida bajariladigan o’tishdir. Shartning rostligi bayroqning qiymati orqali tekshiriladi. Shuning uchun bevosita shartli o’tish buyrug’idan oldin bayroqlar qiymatini shakllantiruvchi taqqoslash buyrug’i qo’yiladi:

CMP <operand$_1$>, <operand$_2$>

Taqqoslash buyrug’i SUB buyrug’i bilan ekvivalent bo’lib, faqatgina ayirma hech qayerga yozilmaydi. CMP buyrug’ining vazifasi bayroqlarni o’rnatish va tashlab yuborishdan iborat.

Shartli o’tish buyruqlari yetarlicha ko’p bo’lib, lekin ular hammasi bir shaklda yoziladi:

Jxx <metka>

Barcha shartli o’tish buyruqlarini uch guruhga ajratish mumkin. **Birinchi guruhga** odatda taqqoslash buyrug’idan keyin qo’yiladigan shartli o’tish buyruqlari kiradi.

Mnemokod	nomi	*CMP op$_1$, op$_2$ buyrug’idan so’ng o’tish sharti*	Bayroqlar qiymati	Qo’llanilishi
JE	Agar teng bo’lsa o’tish	op1 = op2	ZF = 1	Barcha sonlar uchun
JNE	Agar teng bo’lmasa o’tish	op1 ≠ op2	ZF = 0	
JL/JNGE	Agar kichik bo’lsa o’tish	op1 < op2	SF ≠ OF	Belgili sonlar uchun
JLE/JNG	Agar kichik yoki teng bo’lsa o’tish	op1 ≤ op2	SF ≠ OF ili ZF = 1	
JG/JNLE	Agar katta bo’lsa o’tish	op1 > op2	SF = OF i ZF = 0	
JGE/JNL	Agar katta yoki teng bo’lsa o’tish	op1 ≥ op2	SF = OF	
JB/JNAE	Past bo’lsa o’tish	op1 < op2	CF = 1	Belgisiz sonlar uchun
JBE/JNA	past yoki teng bo’lsa o’tish	op1 ≤ op2	CF = 1 ili ZF = 1	
JA/JNBE	Baland bo’lsa o’tish	op1 > op2	CF = 0 i ZF = 0	
JAE/JNB	Baland yoki teng bo’lsa o’tish	op1 ≥ op2	CF = 0	

Mnomonik belgilanishi	Original termini	Tarjimasi	Operand tipi
E	Equal	Teng	Ixtiyoriy
N	Not	Yo’q	Ixtiyoriy
G	Greater	Katta	Ishorali sonlar
L	Less	Kichik	Ishorali sonlar
A	Above	Yuqori(katta)	Ishorasiz sonlar
B	Below	Quyi(kichik)	Ishorasiz sonlar

MISOL: Ikkita x va y o’zgaruvchilar berilgan. Z o’zgaruvchiga ularning kattasini o’zlashtirish talab etilsin.

```
  mov   eax, x
  cmp   eax, y
  jge/jae L                ; JGE ni belgili sonlar uchun va JAE ni belgisiz sonlar uchun ishlatamiz
  mov   eax, y
L: mov   z, eax
```

Shartli o’tish buyruqlarining **Ikkinchi** guruhiga u yoki bu bayroqning qiymatiga bog’liq holda ishlaydigan shartli o’tish buyruqlari kiradi

Mnemokod	O’tish sharti	Mnemokod	O’tish sharti
JZ	ZF = 1	JNZ	ZF = 0
JS	SF = 1	JNS	SF = 0
JC	CF = 1	JNC	CF = 0
JO	OF = 1	JNO	OF = 0
JP	PF = 1	JNP	PF = 0

MISOL: ***a***, ***b*** *va* ***c*** – belgisiz 1 baytli o’zgaruvchilar bo’lsin. ***c*** =***a*** * ***a*** + ***b*** ***ni hisoblash talab etiladi***, agar natija 1 bayt chegarasidan o’tib ketsa, boshqaruvni ERROR metkasiga berilsin.

```
mov   al, a
mul   al
jc    ERROR
add   al, b
jc    ERROR
mov   c, al
```

Shartli o’tish buyruqlarining **uchinchi guruhiga** ikkita buyruq kiradi. Ular bayroqlar qiymatini tekshirmaydi, balki ECX yoki CX registrlar qiymatlarini tekshiradi:

JCXZ <metka> ; CX registr qiymati 0 bo’lsa o’tish
JECXZ <metka> ; ECX registr qiymati 0 bo’lsa o’tish

Biroq bu buyruq etarli darajada uzoq bajariladi. 0 bilan taqqoslash va Oddiy shartli o’tish buyrug’idan foydalanish samaralidir

O’tish buyruqlari yordamida har qanday tarmoq va sikllarni amalga oshirish mumkin.

; if (x > 0) *S*

```
   cmp   x, 0
   jle   L
   ...                  ; S
L:
```

; if (x) *S1* else *S2*

```
   cmp   x, 0
   je    L1
   ...                  ; S1
   jmp   L2
L1: ...                 ; S2
L2:
```

; if (a > 0 && b > 0) *S*

```
   cmp   a, 0
   jle   L
   cmp   b, 0
   jle   L
   ...                  ; S
L:
```

; if (a > 0 || b > 0) *S*

```
   cmp   a, 0
   jg    L1
   cmp   b, 0
   jle   L2
L1: ...                 ; S
L2:
```

; if (a > 0 || b > 0 && c > 0) *S*

```
   cmp   a, 0
   jg    L1
   cmp   b, 0
   jle   L2
   cmp   c, 0
   jle   L2
L1: ...                 ; S
L2:
```

; while (x > 0) do *S*

```
L1: cmp x, 0
    jle L2
    ...                 ; S
    jmp L1
L2:
```

; do *S* while (x > 0)

```
L:  ...                 ; S
    cmp x, 0
    jg L
```

ISHNI BAJARISH TARTIBI

1. Kompyuterdan masm32 dasturini ishga tushiramiz
2. Quyida keltirilgan misolni dasturini tuzamiz
3. MS wordda dasturlash natijalari asosida laboratoriya ishini tahrirlash

Nazorat savollari

1. Taqqoslash buyrug’idan qanday foydalaniladi?
2. CMP buyrug’ining vazifasi va sintaksisi qanday?
3. JMP buyrug’ining vazifasi va sintaksisi qanday?
4. JG buyrug’ining vazifasi va sintaksisi qanday?
5. JLE buyrug’ining vazifasi va sintaksisi qanday?
6. Shartli o’tish buyruqlari guruhlari nechta va ular qanday farqlanadi?

6-Laboratoriya ishi

Dastur osti dasturlarini yozish, ularga murojaat qilish, va dasturlarni yaratishda stekdan foydalanish usullari

Ishdan maqsad: Dastur osti dasturlarini yozish, ularga murojaat qilish, va dasturlarni yaratishda stekdan foydalanish usullarini o’rganish

NAZARIY QISM

Stek bilan ishlash buyruqlari

Stek bilan ishlash bevosita proseduralarga aloqadorlikka ega. Stek proseduraning lokal ma’lumotlarini saqlash va parametrlarini uzatish uchun foydalaniladi.

Stek bilan ishlashda faqatgina ikkita amal bajariladi. Ma’lumotlarni qo’yish va olish.

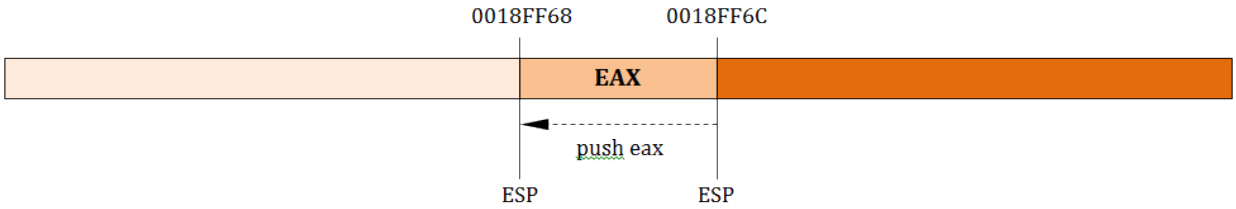
Har bir amal uchun qanday ma’lumotlar bilan ishlashiga qarab har xil bir nechta buyruqlar mavjud.

Ma’lumotlarni stekka qo’yish uchun PUSH buyrug’idan foydalaniladi:

PUSH <operand>

Operand registr, xotira yacheykasi yoki bevosita operand bo’lishi mumkin.

Operand o’lchami 2 yoki 4 bit bo’lishi kerak. Operand stekning yuqori qismiga joylashtiriladi va ESP registrning qiymati operand o’lchamiga qisqaradi.



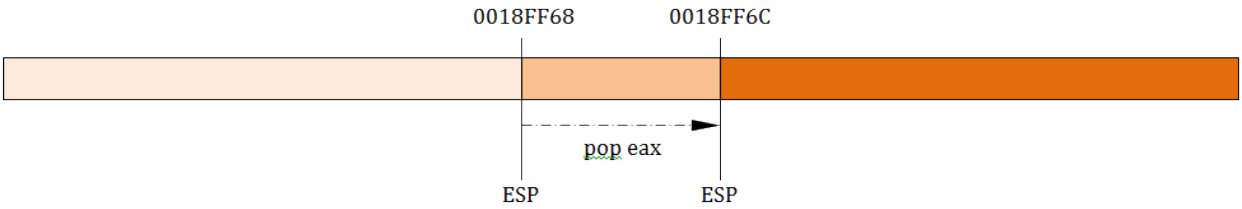
Stekdan ma’lumotni olish uchun POP buyrug’idan foydalaniladi:

POP <operand>

Operand registr, xotira yacheykasi bo’lishi mumkin.

Operand o’lchami 2 yoki 4 bit bo’lishi kerak.

Razmer operanda doljen быть 2 ili 4 bayta. Operandning o’lchamiga bog’liq holda stekning yuqorisidan 2 yoki 4 bayt olinadi va ko’rsatilgan registr yoki xotira yacheykasiga joylashtiriladi. ESP registrning qiymati operand o’lchamiga mos holda ortadi



Ushbu asosiy buyruqlardan tashqari umumiy foydalanish registrlaridagi ma’lumotni stekka saqlash va stekdan qayta tiklash imkonini beruvchi dasturlar hamda bayroq registrlaridagi ma’lumotni stekka saqlash va stekdan qayta tiklashni bajaruvchi buyruqlar mavjud.

PUSHA
PUSHAD

PUSHA buyrug’i AX, CX, DX, BX, SP, BP, SI, DI registrlardagi ma’lumotni stekka saqlaydi. *PUSHAD buyrug’i* EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI registrlardagi ma’lumotni stekka saqlaydi.

POPA
POPAD

Ushbu buyruqlar avvalgilariga aksincha vazifa bajaradi. (E)DI, (E)SI, (E)BP, (E)SP, (E)BX, (E)DX, (E)CX, (E)AX registrlarning qiymatlarini stekdan qayta tiklaydi.

PUSHF
PUSHFD

PUSHF buyrug’I bayroq registrlarining kichik 16 bitini stekka saqlaydi.
PUSHFD buyrug’I bayroq registrining barcha 32 bitini stekka saqlaydi
POPF
POPFD

POPF buyrug’I bayroq registrining kichik 16 bitini stekdan qaytaradi.
POPFD buyrug’I bayroq registrining barcha 32 bitini stekdan qaytaradi.

Protsedura Sintaksisi

Assembler tilida proseduraning ifodalanishi quyidagicha shaklga ega:

<protsedura nomi> *PROC*
<protsedura tanasi>
<protsedura nomi> *ENDP*
Prosedura nomidan keyin ikki nuqta yo’qligiga qaramay bu nom proseduraning birinchi buyrug’ini anglatuvchi metka hisoblanadi.

Assembler tilida prosedurada ifodalangan nomlar va metkalar prosedura ichida lokallashmaydi, shuning uchun ularni unikal bo'lishi talab etiladi.

Prosedurani assembler tilidagi dasturga joylashtirish quyidagicha amalga oshiriladi.

Proseduraning buyruqlari o'z-o'zicha ishlamasdan qachonki unga murojaat etilganda ishlashi uchun prosedurani kod seksiyasi boshlanishida, ya'ni .code direktivasidan so'ng yoki kod seksiyasi so'ngida *ExitProcess* funksiyasi chaqirilgandan so'ng joylashtiriladi.

Prosedurani chaqirish va proseduradan qaytarish

Prosedurani chaqirish-bu mohiyatan boshqaruvni proseduraning birinchi buyrug'iga uzatishdir

Boshqaruvni uzatish uchun prosedura nomi sifatida berilgan metkaga shartsiz o'tish orqali amalga oshirish mumkin

Xatto **proc** va **endp** direktivalaridan foydalanish ham shart emas, balki oddiy ikki nuqtali metkani ***ExitProcess*** *funksiyasi chaqirilgandan so'ng yozish kifoya*. Lekin odatda bunday qilinmaydi.

Assembler tili buyruqlar tizimi prosedurani chaqirish va proseduradan qaytarish uchun maxsus buyruqlarga ega

CALL <protsedura nomi> ; protsedurani chaqirish
RET ; protseduradan qaytarish

CALL buyrug'i

```
.686
.model flat, stdcall
option casemap: none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib
.code
program:
  call  Procedure
  push  0
  call  ExitProcess
Procedure proc
  ret
Procedure endp

end program
```

protsedura parametrlarini uzatish

Prosedura parametrlarni uzatish uchun bir nechta usullari mavjud

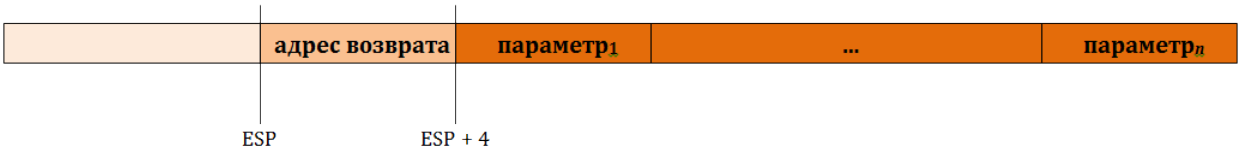
1. *Parametrlar registr orqali uzatilishi mumkin.*

Agar prosedura kam sonli parametrlarni qabul qiladigan bo'lsa, unda parametrlarni registr orqali uzatish mumkin.

push <parametr$_n$>
...
push <parametr$_1$>
call Procedure

Yuqorida keltirilgan misolda stekka bir nechta parametrlarni qo'yish va prosedura chaqiriladi.

CALL buyrug'i qaytarish manzilini stekka joylash vazifasini ham bajaradi. Ushbu holatda stekning holati proseduraning birinchi buyrug'i bajarilishidan oldin quyidagicha ko'rinishda bo'ladi



; stdcall chaqiruvlarga ruhsat berish orqali prosedura parametrlarini uzatish va proseduradan qaytarish

```
.686
.model flat, stdcall
option casemap: none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
  x dd 0
  y dd 4

.code
program:
   push  y                    ; 2 ta 4 baytli parametrlarni stekka joylash
   push  x
   call  Procedure

   push  0
   call  ExitProcess
```

```
Procedure proc
  ret  8                ; buyruqda stekdan 8 bayt joy bo’shatish kerakligini
ko’rsatamiz
Procedure endp

end program
```

; *cdecl* chaqiruvlarga ruhsat berish orqali prosedura parametrlarini uzatish va proseduradan qaytarish

```
.686
.model flat, c
option casemap: none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
  x dd 0
  y dd 4

.code
program:
  push  y               ; 2 ta 4 baytli parametrlarni stekka joylash
  push  x
  call  Procedure
  add   esp, 8                  ; 8 bayt stekni bo’shatamiz
  push  0
  call  ExitProcess

Procedure proc
  ret                   ;  parametrlarsiz qaytarish buyrug’idan foydalanamiz
Procedure endp

end program
```

prosedura natijasini uzatish

Prosedura natijasini uzatish uchun odatda EAX registridan foydalaniladi. 8 baytdan katta qiymatlar uchun EDX:EAX registrlar juftidan foydalaniladi.

```
.686
.model flat, c
option casemap: none

include \masm32\include\windows.inc
```

```
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
  a dd 76
  b dd -8
  d dd ?

.code
program:
   push  b                  ;  stekka parametrni joylaymiz
   push  a
   call  Procedure
   add   esp, 8                     ; 8 bayt stekni bo'shatamiz
steka
   mov   d, eax                     ; d = a – b

   push  0
   call  ExitProcess

Procedure proc
   mov   eax, [esp + 4]             ; EAX registriga birinchi parametrni yozamiz
   mov   edx, [esp + 8]     ; EDX registriga ikkinchi parametrni yozamiz
   sub   eax, edx           ; EAX registrda parametrlar ayirmasini olamiz
   ret
Procedure endp

end program
```

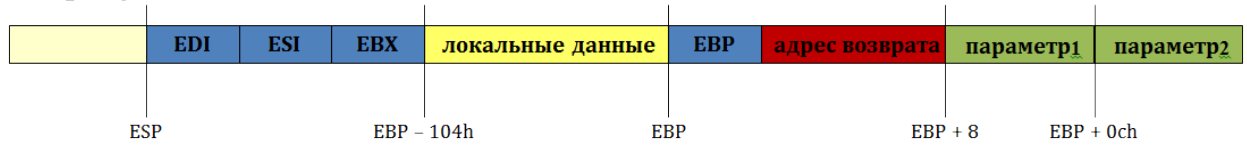
; Stek orqali parametrlarni uzatish va natijani adres bo'yicha qaytarish

```
.686
.model flat, c
option casemap: none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib
.data
  a dd 76
  b dd -8
  d dd ?
.code
program:
   push  offset d           ;  natija  yozilishi  kerak  o'zgaruvchi  adresini  stekka
joylaymiz
   push  b
```

```
  push  a
  call  Procedure
  add   esp, 12              ;  Stekni 12 baytini bo’shatamiz
  push  0
  call  ExitProcess

Procedure proc
  mov   eax, [esp + 4]             ; EAX registriga birinchi parametrni yozamiz
  mov   edx, [esp + 8]     ; EDX registriga ikkinchi parametrni yozamiz
  sub   eax, edx           ; EAX registrda parametrlar ayirmasini olamiz
  mov   edx, [esp + 12]    ; uchinchi parametr-natija adresini EDX regitrga yozamiz
  mov   [edx], eax         ;  EDX registrga adres bo’yicha natijani yozamiz
  ret
Procedure endp

end program
```



ì

ISHNI BAJARISH TARTIBI

1. Kompyuterdan masm32 dasturini ishga tushiramiz
2. Quyida keltirilgan misolni dasturini tuzamiz
3. MS wordda dasturlash natijalari asosida laboratoriya ishini tahrirlash

Nazorat savollari

1. Prosedurani chaqirish buyrug’idan qanday foydalaniladi?
2. PUSH buyrug’ining vazifasi va sintaksisi qanday?
3. CALL buyrug’ining vazifasi va sintaksisi qanday?
4. POP buyrug’ining vazifasi va sintaksisi qanday?
5. ExitProcess buyrug’ining vazifasi va sintaksisi qanday?

7-Laboratoriya ishi

Qo’shish va ayrish amallarini bajarish dasturlari.

Ishdan maqsad: Qo’shish va ayirish amallarini bajarish buyruqlarini, ularning bajarilish jarayonlarida yuzaga keladigan muammolarni izohlash va ularni yechish yo’llarini o’rganish

NAZARIY QISM

QO’SHISH VA AYIRISH BUYRUQLARI

Qo’shish va ayirish buyruqlari barchaga ma’lum bo’lgan arifmetik amallarni bajaradi.

Ushbu amallarning bajarilishida sonlarni kompyuter xotirasida ifodalanishi bilan bog’liq bo’lgan o’ziga xos tomonlarni hisobga olish zarur bo’ladi

ADD <operand$_1$>, <operand$_2$>
SUB <operand$_1$>, <operand$_2$>

ADD buyrug’i berilgan operandlarni yig’indisini hisoblaydi va natijani birinchi operand o’rniga yozadi. SUB buyrug’i esa birinchi operanddan ikkinchi operandni ayiradi va natijani birinchi operand or’niga yozadi.

Operandlar bir xil o’lchamga ega bo’lishi zarur. Agar birinchi operand registr bo’lsa, unda ikkinchi operand registr bo’lishi yoki xotira yacheykasi yoki bevosita operand bo’lishi ham mumkin. Agar birinchi operand xotira yacheykasi bo’lsa, unda ikkinchi operand registr yoki bevosita operand bo’lishi mumkin

Belgili va belgisiz sonlarning har qanday o’lchamlari ustida qo’shish va ayirish amallarini bajarish mumkin. Ushbu buyruqlar AF, CF, OF, PF, SF va ZF bayroqlarini o’zgartiradi.

```
a dd 45d
b dd -32d
c dd ?

mov   eax, a
add   eax, b
mov   c, eax          ; c = a + b
```

inkrement va *dekrement* buyruqlari o’z operandini bittaga oshiradi yoki kamaytiradi

INC <operand>
DEC <operand>

Operand sifatida registr yoki har qanday o’lchamdagi xotira yacheykasi berilishi mumkin. inkrement va dekrement buyruqlari qo’shish va ayirish buyruqlariga nisbatan kamroq joy egallashi bilan foydalidir

inc eax

Arifmetik amallar qatoriga belgini (ishora)o'zgartirish buyrug'ini ham kiritish mumkin:

NEG <operand>

operand sifatida regitr yoki har qanday o'lchamdagi xotira yacheykasi berilishi mumkin.

```
mov   ax, 1
neg   ax              ; AX = -1 = ffffh
mov   bl, -128
neg   bl              ; BL = -128, OF = 1
```

ISHNI BAJARISH TARTIBI

```
.486
.model flat, stdcall
Option casemap:none
.data
a dd 45d
b dd -32d
.data?
c dd ?
.code
Start:
mov   eax, a
add   eax, b
mov   c, eax          ; c = a + b
end start
```

1. Kompyuterdan masm32 dasturini ishga tushiramiz
2. Yuqorida keltirilgan misollarni dasturini tuzamiz
3. MS wordda dasturlash natijalari asosida laboratoriya ishini tahrirlash

Nazorat savollari

1. Arifmetik amallardan qaysilarini bilib oldingiz?
2. ADD buyrug'ining vazifasi va sintaksisi qanday?
3. SUB buyrug'ining vazifasi va sintaksisi qanday?
4. DEC buyrug'ining vazifasi va sintaksisi qanday?
5. INC buyrug'ining vazifasi va sintaksisi qanday?
6. NEG buyrug'ining vazifasi va sintaksisi qanday?

8-Laboratoriya ishi

Ko’paytirish va bo’lish amallarini bajarish dasturlari.

Ishdan maqsad: Ko’paytirish va bo’lish amallarini bajarish buyruqlarini, ko’paytirish va bo’lish jarayonlarida yuzaga keladigan muammolarni izohlash va ularni yechish yo’llarini o’rganish

NAZARIY QISM

Tiplarni o’zgartirish (**Preobrazovaniya**). **CBW** (Convert Byte to Word) – mazkur komanda AL dagi bayt ma’lumtoni AX ga 16 bitli so’z ma’lumotiga o’zgartiradi **CWD** (Convert Word to Double) – mazkur komanda AX registridagi vord tipidagi 16 bitli ma’lumotni 32 bitli ikkilik vord (devord) tipidagi ma’lumotga o’zgartiradi. Bunda 32 bitli sonning katta razryadi DX ga, kichik razryadi esa AX ga yoziladi **CWDE** (Convert Word to Double Extended) – mazkur komanda AX dagi 16 bitli ma’lumotni YEAX registriga 32 bitli ma’lumotga o’zgartirib yozadi **CDQ** (Convert Double Word to Quarter Word) – mazkur komanda EAX registridagi 32 bitli ma’lumotni 64 bitli ma’lumotga konvertatsiya qiladi. Bunda hosil bo’lgan 64 bitli sonning katta razryadi (katta 32 biti) EDX registriga, kichik razryadi (kichik 32 biti) esa EAX registriga yoziladi

Belgi (znak) siz ko’paytirish komandalari: MUL <ko’paytiriluvchi>

Birinchi ko’paytiriluvchi	**Ikkinchi ko’paytiriluvchi**	**Natija**
Bayt	AL	AX -> AL – kichik qism AH – katta qism
So’z	AX	DX:AX -> AX – kichik qism DX – katta qism
Ikkita so’z	EAX	EDX:EAX-> EAX – kichik qism EDX – katta qism

Ko’paytirish operatsiyasi bajarilganida natija ikki qismdan iborat o’ladi va operandlarning o’lchamidan kelib chiqqan holda ikki o’ringa joylashadi – kichik qismi ikkinchi ko’paytiriluvchi o’rniga va qo’shimcha registrlar (AH, DX, EDX) ga. Natijaning qaerga joylashganligini aniqlash uchun perenos flagi CF va perepolneniya flagi OF ishlatiladi. 1) Agar natijaning katta qismi=0 bo’lsa, u holda operatsiya bajarilganidan so’ng CF=0 i OF = 0 bo’ladi. 2) Agar CF=1 yoki OF = 1 bo’lsa, u holda bu shuni anglatidiki, operatsiya natijasi kichik qism (mladщiy chastь) chegarasidan oshib ketgan va keyingi holatlarda buni e’tiborga olish kerak. 3) Ishorali ko’paytirish komandasi 4) IMUL ko’paytiriluvchi1 5) Bitta operandli ishorali ko’paytirish – bunda faqat bitta ko’paytiriluvchini ko’rsatish talab etiladi. Bu huddi yuqoridagi MUL kabi bo’lib, natijaning qaysi registrga yoki registrlarga

yozilishi ko’paytiriluvchilarning o’lchamiga bog’liq bo’ladi. 6) IMUL ko’paytiriluvchi1, ko’paytiriluvchi 7) Ikki operandli ishorali ko’paytirish – Bunda ikkala operand o’razo ko’paytirilib, natija birinchi operandga o’zlashtiriladi. 8) IMUL natija, ko’paytiriluvchi1, ko’paytiriluvchi 9) Uch operandli ishorali ko’paytirish – bunda ko’paytiriluvchi1 va ko’paytiriluvchi2 operandlari o’razo ko’paytirilib, birinchi natija operandiga o’zlashtiriladi. Bunda uchinchi operand bayt, slovo yoki ikkita so’z o’lchamida bo’lishi mumkin.

DIV делитель

Результатом команды деления являются значения частного и остатка

Делимое	Делитель	Частное	Остаток
Слово (16 бит) в регистре AX	Байт в регистре или ячейке памяти	Байт в регистре AL	Байт в регистре AH
Двойное слово (32 бит) в DX – старшая часть в AX – младшая часть	Слово (16 бит) в регистре или в ячейке памяти	Слово (16 бит) в регистре AX	Слово (16 бит) в регистре DX
Учетверенное слово (64 бит) в EDX – старшая часть в EAX – младшая часть	Двойное слово (32 бит) в регистре или в ячейке памяти	Двойное слово (32 бит) в регистре EAX	Двойное слово (32 бит) в регистре EDX

IDIV делитель

Результатом команды деления со знаком являются значения частного и остатка

Делимое	Делитель	Частное	Остаток
Слово (16 бит) в регистре AX	Байт в регистре или ячейке памяти	Байт в регистре AL	Байт в регистре AH
Двойное слово (32 бит) в DX – старшая часть в AX – младшая часть	Слово (16 бит) в регистре или в ячейке памяти	Слово (16 бит) в регистре AX	Слово (16 бит) в регистре DX
Учетверенное слово (64 бит) в EDX – старшая часть в EAX – младшая часть	Двойное слово (32 бит) в регистре или в ячейке памяти	Двойное слово (32 бит) в регистре EAX	Двойное слово (32 бит) в регистре EDX

ISHNI BAJARISH TARTIBI

4. Kompyuterdan masm32 dasturini ishga tushiramiz
5. Quyida keltirilgan misolni dasturini tuzamiz
6. MS wordda dasturlash natijalari asosida laboratoriya ishini tahrirlash

Misol: Assembler dasturlash tilida quyidagi berilgan qiymatlar uchun (c+a)/2+(a*2+d)/b ni hisoblash dasturini tuzish: a – db tipidagi o’zgaruvchi, b – dw tipidagi o’zgaruvchi, c – dd tipidagi o’zgaruvchi, d – db tipidagi o’zgaruvchi.

Yechish:

.DATA; (c+a)/2+(a*2+d)/b ni hisoblashga tegishli ma’lumotlarni e’lon qilamiz

```
 d_a db -110; a o’zgaruvchini e’lon qilamiz
 d_b dw -90; b o’zgaruvchini e’lon qilamiz
 d_c dd -32000; c o’zgaruvchini e’lon qilamiz
 d_d db -120; d o’zgaruvchini e’lon qilamiz
 d_rez dd 0; (c+a)/2 qiymatining vaqtinchalik o’zgaruvchisini e’lon qilamiz
 d_rez2 dw 0; a*2 qiymatining vaqtinchalik o’zgaruvchisini e’lon qilamiz
 d_rez3 dw 0; (a*2+d) qiymatining vaqtinchalik o’zgaruvchisini e’lon qilamiz;
        .CODE
Main:
mov eax,0 ; eax registrini bo’shatish (obnulenie)
mov edx,0 ; edx registrini bo’shatish (obnulenie)
mov al,d_a ; a operandni AL ga o’zlashtiramz
mov ebx,d_c ; s operandni EBX ga o’zlash-z
PrintText"(c+a)/2+(a*2+d)/b" masala berlishini ekranga chiqaramiz
cbw ; AL registridagi qiymatni AX ga kengaytiramiz
cwde ; AX dagini EAX ga kengaytiramiz
PrintDec eax ; EAX dagi a qiymatni ekranga chiqaramiz
PrintDec ebx ; EBX dagi s qiymatni ekranga chiqaramiz
add eax,ebx ; EAX=EAX+EBX
PrintDec eax ; natijani ekranga chiqaramiz
mov ebx, 0 ; EBX registrni bo’shatish (obnulenie)
mov ebx,2 ; EBX ga 2 qiymatni o’zlashtiramiz
idiv ebx ; (c+a)/2 – ishorali bo’lishni amalga oshirish
PrintDec ax ; bo’lish operatsiyasi natijasini ekranga chiq.
cwde ; AX dagini EAX ga kengaytirish
mov d_rez, eax ; d_rez=(c+a)/2 o’zlashtirish
PrintDec d_rez ; (c+a)/2 natijani ekranga chiqarish
mov al, d_a ; a ni AL ga o’zlashtirish
mov bl, 2 ; VL ga 2 qiymatni o’zlashtirish
imul bl ; a*2 ko’paytmani ishorali ko’paytirish
PrintDec ax ko’paytirish natijasini ekranga chiqarish
mov d_rez2, ax ; a*2 ko’paytma natijasini AX dan d_rez2 ga yozish
mov eax, 0 ; eax ni bo’shatish
mov al, d_d ; d ni AL ga yozish
cbw ; AL dagi qiymatni AX ga kengaytirish
add ax, d_rez2 ; a*2+d – qo’shishni amalga oshirish
PrintDec ax; natijani ekranga chiqarish
mov d_rez3, ax ; a*2+d natijasini AX dan d_rez3 o’zgaruvchisiga yozish
mov eax, 0 ; eax ni bo’shatish
mov ax, d_rez3 ; d_rez3= (a*2+d) qiymatni AX ga yozish
cwd ; AX dagi 16 bitni 32 bitga DX/AX ga kengaytiramiz
idiv d_b ; (a*2+d)/b – ishorali bo’lishni amalga oshirish
PrintDec eax ; natijani ekranga chiqarish
add eax, d_rez ; (c+a)/2+(a*2+d)/b – ikkala qismn qo’shish
PrintDec eax ; umumiy natijani ekranga chiqarish
```

Nazorat savollari

7. Tiplarni o’zgartirish buyrug’idan qanday foydalaniladi?
8. MUL buyrug’ining vazifasi va sintaksisi qanday?
9. MUL buyrug’ining vazifasi va sintaksisi qanday?
10. DIV buyrug’ining vazifasi va sintaksisi qanday?
11. IDIV buyrug’ining vazifasi va sintaksisi qanday?
12. Bo’lish va ko’paytirishda yuzaga kelishi mumkin bo’lgan hollarni izohlang

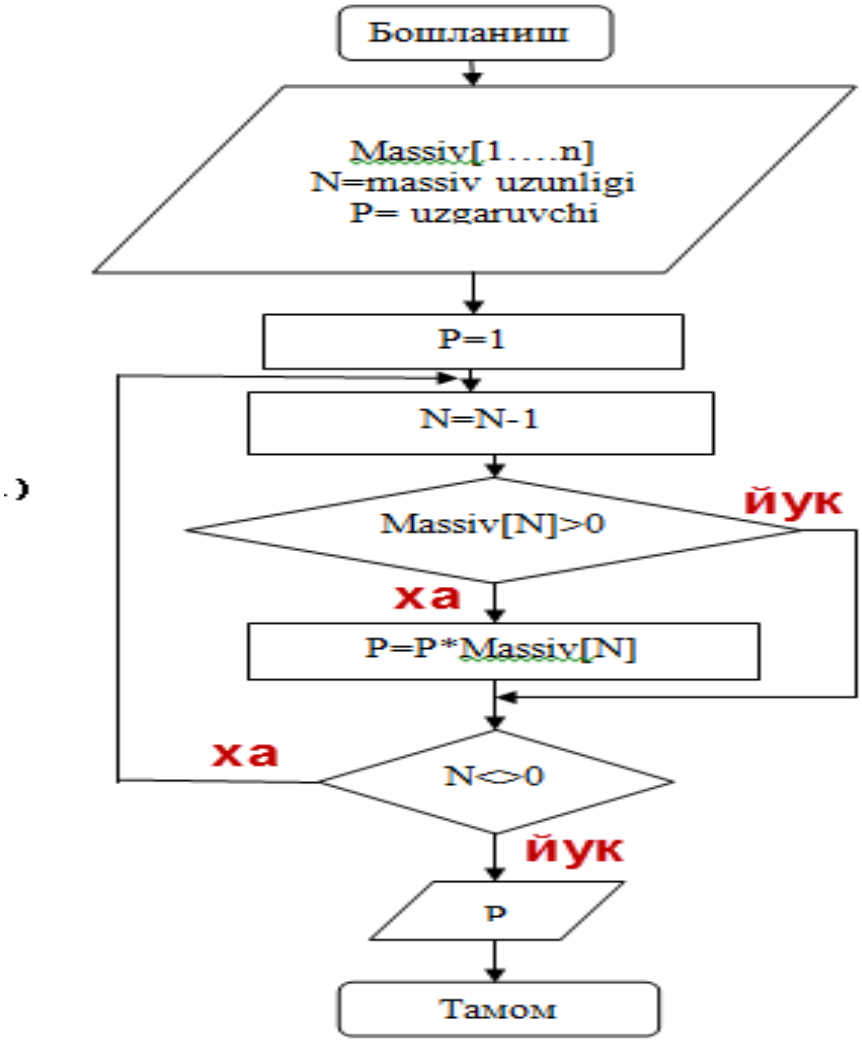
9-laboratoriya ishi

Maxsus funksiyalarni hisoblash dasturlari.

Ishdan maqsad: Assembler tilida maxsus funktsiyalarni hisoblash dasturlari tuzishni o'rganish

NAZARIY QISM

Misol: Berilgan massivning noldan katta elementlarini ko'paytmasini hisoblash algoritmi va assemblerda dasturini tuzish.




```
        ;================================================
    .DATA
       ket_k dd  5, 25, -5, 0, -2, -1, -125, -65, 100  ;  uzunligi 9 ga teng
       n_uzunlik dd  9
;================================================
.CODE
Main:
   mov ebx,0 ; registrni  tozalash
   mov ecx,0
   mov ebx,1
   lea esi,ket_k   ; massivning  boshlangich adresi
```

```
  mov ecx,n_uzunlik
m1:
  mov eax,0
  mov eax, [esi]
  add esi,4   ; 4*8(qadam biti)=32 bit
    dec ecx
  cmp eax,0
  jg m2  ; ishorani  etiborga  olib taqqoslash (yani ZF va SF ni tekshiradi)
  jmp m3
m2:
  mul ebx
  mov ebx,eax
m3:
    cmp ecx,0
    ja m1
  PrintText  'Musbat  sonlar kupaytmasi P = '
  PrintDec ebx
    invoke ExitProcess,0
END Main
```

CreateFile –bu funktsiya faylni yaratish yoki ochish uchun ishlatiladi. Funktsiyani chaqirishning qisqartirilgan formati: invoke CreateFile, adres_imeni_fayla, rejim_dostupa, 0, 0, rejim_otkrtiya, atribut_fayla, 0. Parametrlari: **adres_imeni_fayla** – fayl nomini ko'rsatuvchi, nol bilan tugovchi satrga ko'rsatkich. **rejim_dostupa** –faylga ruxsat olish rejimini ko'rsatadi (rejim dostupa k faylu)

Qiymat	**Ma'nosi (Tavsifi)**
GENERIC_READ	Fayldan o'qishga ruxsat berishni ko'rsatadi
GENERIC_WRITE	Faylga yozishga ruxsat berishni ko'rsatadi

rejim otkrtiya – fayl ochilgandagi harakatni ko'rsatadi:

Qiymat	**Ma'nosi (Tavsifi)**
CREATE_NEW	Yangi fayl yaratadi. Agar fayl mavjud bo'lsa, funktsiya xatolik qaytaradi
CREATE_ALWAYS	Yangi fayl yaratadi. Agar fayl mavjud bo'lsa, funktsiya fayl o'lchamini nollaydi va fayl atributlarini qayta yozadi.
OPEN_EXISTING	Mavjud faylni ochadi. Agar fayl mavjud bo'lmasa xatolik qaytaradi.
OPEN_ALWAYS	Mavjud faylni ochadi. Agar fayl mavjud bo'lmasa, funktsiya yangi fayl yaratadi
TRUNCATE_EXISTING	Mavjud faylni ochadi va o'lchamini nollaydi. Agar fayl mavjud bo'lmasa xatolik qaytaradi.

ReadFile. Bu funktsiya fayl ko’rsatkichining joriy o’rnidan boshlab, xotira bufferiga ochilgan fayldan ma’lumotlarni o’qish uchun foydalaniladi. Funktsiyani chaqirishning qisqartirilgan formati: invoke ReadFile, ukazatel_fayla, adres_bufera, kolichestvo_baytov_chteniya, adres_kol-va_prochtennx_baytov,0.
WriteFile. Bu funktsiya fayl ko’rsatkichining joriy o’rnidan boshlab, xotira bufferidan faylga ma’lumotlarni yozish uchun foydalaniladi. Funktsiyani chaqirishning qisqartirilgan formati: invoke WriteFile, ukazatel _ fayla, adres _ bufera, kolichestvo _ baytov _ zapisi, adres _ kolichestva _ zapisannx _ baytov,0.
CloseHandle - bu funktsiya faylni yopish uchun foydalainladi. Funktsiyani chaqirish formati: invoke CloseHandle, fayl_ko’rsatkichi

NAZORAT SAVOLLARI

1. **Fayllar bilan ishlash jarayonini izohlab bering**
2. **CreateFile funksiyasi** qanday vazifa bajaradi?
3. **CreateFile funksiyasini qo’llash sintaksisi** qanday?
4. **ReadFile** funksiyasidan nima uchun va qanday foydalaniladi?
5. **WriteFile** funksiyasidan nima uchun va qanday foydalaniladi?
6. **CloseHandle -** funksiyasidan nima uchun va qanday foydalaniladi?

Adabiyotlar:

1. Микропроцессоры 1-2-3 под ред. проф. Л.Н. Преснухина. М. 1986
2. Алексеенко и др. Проектирование радиоэлектронной аппаратуры на микропроцессорах. М. 1984
3. Проектирование микропроцессорных измерителных приборов и систем. В.Д. Циделко и др. Киев, Техника 1984.
4. В.В. Майоров. Гаврилов А.И. Практический курс програмирования МП систем. М. Машиностроение 1989.
5. С. Т. Хвош и др. Микропроцессоры и микро ЭВМ в системах автоматического упраления.
6. Безуглов Д.А., Калиенко И.В. Цифровые устройства и микропроцессоры.- Ростовн/Д.: Феникс,2006.
7. Логические и арифметические основы и принципы работы ЭВМ.: www.интуит.ру
8. Ғаниев С.К., Каримов М.М., Мамбетов Н.М. Ҳисоблаш системаларининг информацион асослари. Олий ўқув юрт.студ. учун дарслик. -Тошкент.: ТДТУ, 2002.
9. Савелев А.Я. Основы информатики. Учеб. Для вузов. –М Изд-во МГТУим. Н.Э Баумана, 2001.
10. Жмакин А.П. Архитектура ЭВМ.Уч.пособие. Санкт-Петербург, БХВ-Петербург, 2006

Internet resurslar

1. **Google.com**
2. **Yandex.ru**
3. **Intuit.ru**
4. **Ziyonet.uz**