050631 – ″Komputer muhəndisliyi″ ixtisas?

″Sistem proqramlaşdırılması″ fənni

V.H. Səlimov

vagif_salimov@yahoo.com

http:// vagif5.tripod.com

1. Sual

Məlumdur ki, muasir komputerlər murəkkəb proqram-texniki quruluşa malikdir. Məlumdᵊr ki butun texniki elementləri muvafiq proqram vasitələrinin koməyilə işləyirlər.

Komputerlərin işləməsini təmin edən proqramlar əməliyyat sistemi (ƏS) və ya sadəcə sistem adlandᵊrᵊlᵊr.

ƏS-lər muasir komputerlərin tərkib hissəsidir və proqram təminat? ilə məşğul olan firmalar tərəfindən istehsal olunur.

Bir qayda olaraq, muasir ƏS-lər kifayət qədər qabaqcᵊl konfiqurasiya etmə və ya başqa sozlə muxtəlif istifadəcilərin tələbatᵊna uyğun sazlama imkanlar? var. Bununla yanaşı, bəzi hallarda ƏS-lərin imkanlarᵊn? genişləndirməsi məqsədilə əlavə proqram vasitələrinin yaradᵊlmas? və ya olanlarᵊn korreksiyas? zərurəti yaranᵊr.

ƏS-lərin təkmiləşdirilməsi ilə bağlı və ona yaxᵊn məsələlərin həlli ilə bağlı proqramlaşdırması **sistem proqramlaşdırılması adlandᵊrᵊlᵊr.**

Sistem proqramlarda proqramciya komputerlərin texniki qurğuları haqda lazᵊmi qədər dərin bilik tələb olunur.

Muasir alqoritmik proqramlaşdırma dillərinin imkanlar? bu kimi məsələlərin həllində kifayət qədər məhduddur.Onlarᵊn sᵊrasᵊnda, ən cox isə **Basic** dilinin son versiyalarᵊnda fiziki səviyyədə yaddaşla işləmək ucun kifayət qədər effektiv vasitələr var, hansᵊlar ki, sistem məsələlərin lazᵊmi qədər geniş sahəsini həll etməyə imkan verir.

Sistem məsələlərinin proqramlaşdırılmasının daha geniş imkanlarᵊn? Assembler dili təqdim edir. Mahiyyətinə gorə Assembler dili maşın dilini təmsil edir, yəni mnemonik formalizm əsasᵊnda butun əmrlər sistemini əhatə edir.

Assembler dili **maşın - yonəmli** dildir, belə ki, o, uyğun tip kompyuterlərin texniki xarakteristikalarᵊn? əhəmiyyətli dərəcədə nəzərə alᵊr.

Şübhəsiz, Assembler dilində proqramçılığın işlənməsi və sazlanması yuksək səviyyəli alqoritmik dillərdə proqramlaşdırmadan əhəmiyyətli dərəcədə **murəkkəbdir.** Uzun muddət Assembler sistem proqramlaşdırılmasının yeganə vasitəsi kimi qalırdı. Assembler dilində proqramlaşdırmanın sistem məsələlərinin həlli ucun xususi dilin –C dilinin yaranmasına gətirib cıxardı.

Gostərilən dil kifayət qədər muvəffəqiyyətlə Assembler tərəfindən təqdim olunan «fiziki» səviyyədə işləmə imkanları olan yuksək səviyyəli alqoritmik dillərin «dostyana» xarakterini ozundə birləşdirir.

C dilinin yaranmasına baxmayaraq, Assembler dili əvvəlki kimi sistem proqramlarının işlənməsi ucun əsas vasitələrdən biri olaraq qalır. C# C dillərin ailəsini son numaindəsinə baxılır, və muxtəlif sistem problemlərin həllin numunələrə diqqət etirilib.

2. 80x86 mikroprosessorlarıhaqda umumi məlumat

Baxmayraq ki komputerlər və onların əsas qurğusu olan prosesorlar (**CPU**) ildən ilə təkmilləşir, onların arxitekturası (ümümi quruluşu) demək olar sabit qalır. Məllumdur ki muassir fərdi komputerlərin əsasını **İNTEL** firmasının istehsalı olan **80x86** prosesorlar? **(8086,80286,80386,80486,80586, P1,P2,P3,P4, Core ...)** təşkil edir. Bu prosesorların əsasını **8086** 16 bitlik (16b) prosesorlar təşkil edir. Məlumdur ki **8086,80286** prosesorlar 16bitlik dir (yəni bir başa 16 bitlik ədədlərlə işləyir və 4kb yaddaşı əhatə edən ünvanlarla işləyə bilir). **80386** başlayraq prosesorlar 32 bitlik oldular və son modellər Core 2 duo və ondan sonra qələnlər **64bit** prosesorlardır. Bu dəyşiliqlər sozsuz assembler dilinin strukturuna muyyən təsir edib buna baxmayaraq əsas əməliyaatlar oz quvvəsində qalir.

Məlumdur ki Ms Dos-da yaddaşın **real rejim** modeli istifadə olunur MS Windows-da 3 rejim istifadə oluna bilər: **real**, qorunan, virtual. Bunnan bağlı

assembler dilinə yeni operatorlar əlavə olunub,onlar yaddaş modelini və prosesorum modelini elan edir. Ms Windows sistemin yaradılması ilə bağlı assemblerin yeni **Win32** versiyası yaradılıb, burada **GUİ** standartindan istifadə etmək imkanı var və bunun üçün APİ (***application program interface***) sistemindən istifadə olunur. **APİ** – Ms Windows da olan standart proqramlardan istifadə olmaqla proqramlaşdırma texnologiyası. Buna baxmayraq assembler dilində əsasən **konsol** proqramları (GUİ-sız) yazılır.

3. Sual **80286 prosesorun arxitekturası?**

Əvvəl 80x86 prosesorların əsasını təşkil edən **80286** prosesorun arxitekturasını nəzərdən kecirdək.

Prosesorda daxilində **14 dənə - 16** bitlik registrlərdə yerləşir. Məlumdur ki **registrlər əməliyyat** yaddaş növünə aid olan xususi cox surətli yaddaşdır

Bunlardan 12 registr umumi təyyinatlı registrləri adlanır, biri-əmrlər unvanı və digəri-vəziyyət registri.

Umumi təyinatlı registrləri 4 registr olmaqla 3 qrupa bolunur:

a) verilənlər registrləri 4

b) indeks və gostəricilər registrləri 4

c) seqment registrləri. 4

8 bitlik (bayt) və ya 16 bitlik (soz) ilə olmasından asılı olaraq verilənlər registrlərinə 4 16 bitlik və ya 8 8 bitlik registr kimi baxmaq olar. Birinci halda registrlər **AX, BX, CX, DX** adlandırılır. Bu registrlər uyğun olaraq 8 8 bitlik registrlərdən ibarətdir: AL, AH, BL, BH, CL, CH, DL, DH. Burada L və H 16 bitlik registrlərin kicik (low) və boyuk (high) baytları mənasını verir. Məsələn, AL və AH uyğun olaraq AX registrinin kicik və boyuk baytlarını əmələ gətirir. Butun bu registrlərdən proqramlaşdırmada istifadə etmək olar. Umumi təyinatlı

registrlərdən butun 8 və ya 16 bitlik riyazi və ya məntiqi əmrlərdə operand kimi istifadə etmək olar.

Bəzi əməliyyatlarʔn yerinə yetirilməsində bu registrlərin xususi təyinatlarʔ olur:

AX-proseslərin daxiletmə/xaricetmə əməliyyatlarʔnda istifadə olunur (İN, OUT əmrləri ilə).

BX-verilənlərin ünvanlaşdırılmasında baza registri kimi istifadə olunur. Bu, yeganə umumi təyinatlʔ registrdir ki, hansʔ ki, unvanlarʔn hesablanmasʔnda istifadə olunur.

CX-bəzi əməliyyatlarʔn yerinə yetirilməsində sayğac kimi istifadə olunur:

– dovrlərin təşkili;

– sətrlərin emalʔ;

Butun reqistrlərin haqqʔnda məlumat cədvəldə verilir

	15 0	7 0	7 0	
1	AX	AH	AL	Akkumlyator
2	BX	BH	BL	Baza registri
3	CX	CH	CL	Sayğac
4	DX	DH	DL	Verilənlər registri :

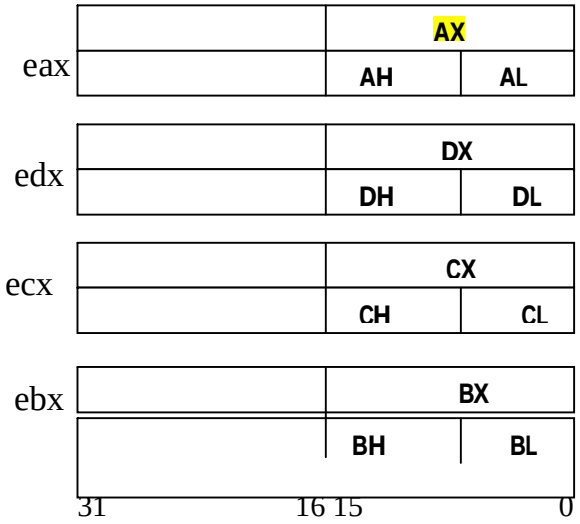
5	SP	Stek göstəricisi :
6	BP	Baza göstəricisi :
7	SI	Məmbə indeksi
8	DI	Qəbuledicinin indeksi :

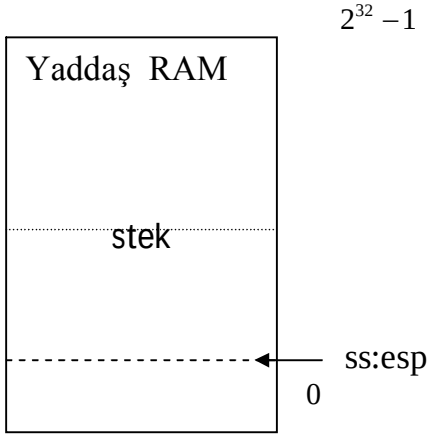
9	**CS**	Əmrlər seqmentinin registri
10	**DS**	Ver-lər seqmentinin registri
11	**SS**	Stek seqmentinin registri.
12	**ES**	Əlavə seqmentinin registri
13	**IP**	Əmrlər göstəricisi
14	**FR**	Bayraq registri.

DX reqistri portun nomrəsi kimi bəzi daxietmə/xaricetmə əməliyyatlarʔnda unvanlarʔ göstərmək ucun istifadə olunur.

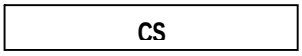
32bit prosesorlar da bəzi dəyişiliklər və qenişlənmələr baş verib.Yəni AX reqistirinə əlavə 16 bitlik reqistiri əlavə olunur və alınan 32 bitlik EAX adlanır (extended). Yeniliklə

32 bit prosesoru strukturu

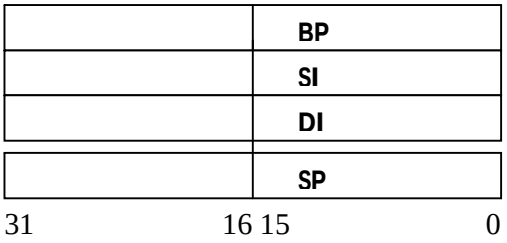


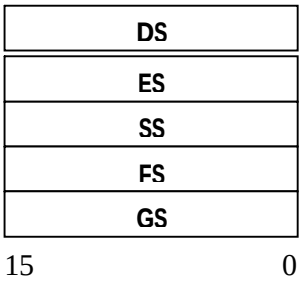


Seqment reqistrləri



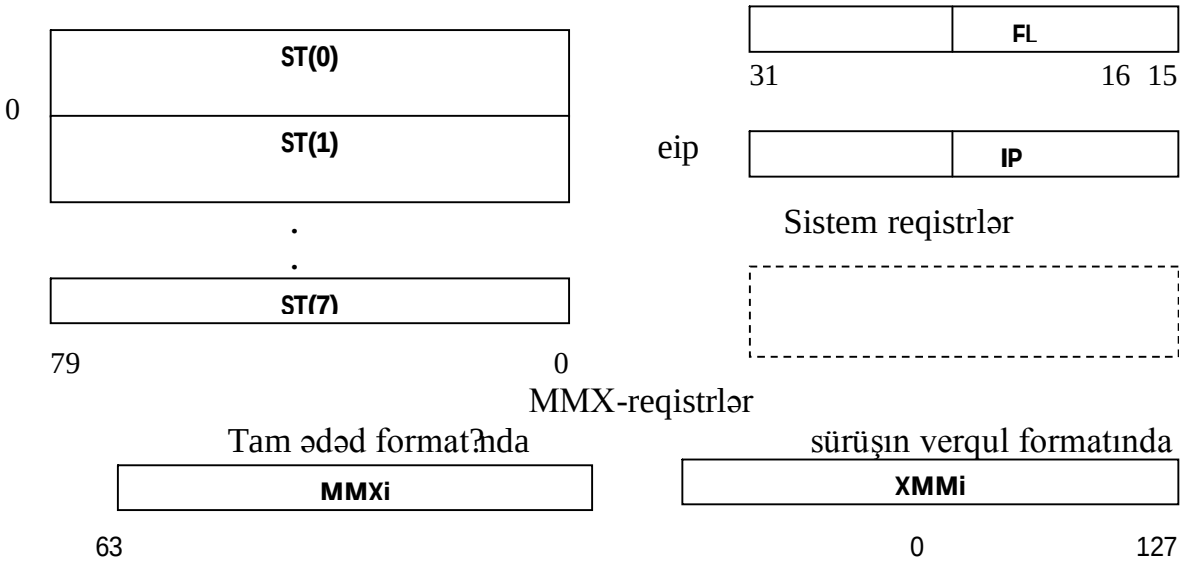
İndeks reqistrləri





Sürüşən verqül fiormatl? reqistrlər

Vəziyyət və idarəetmə reqistrlər



4. Assembler proqram?. Umumi məlumat

Umümi şəkildə assembler proqram? yaddaşda 4 hissədən (seqmentdən) ibarətdir DS, CS, SS, ES. Lazım olmayanda bəzi seqmentlər ixtisar oluna bilər. Yegənə ixtisar olunmayan seqment CS –code seqmentidir !!!!!.

Əğər bu hissələr bir ünvandan başlayırsa onda bu program **.COM** proqram? adlandırılır. Əks halda **EXE.**

16b prosesorlarda bir seqmentin olcusu=4KB.

32b prosesorlarda seqmentin olcusu=4GB.

Hal hazırda bir necə assembler sistemi movcuddur **MASM , TASM, FASM**, **NASM** və başqaları. Eyni **32b, 64b** prosesorlrda əsasən proqram bir seqmentlidir, amma seksiyalara bolunur.

–***Proqram əmrlərdən ibarətdir. Əmrlər unvansız, bir unvanl? və iki unvanl? ola bilərlər.***

Əməliyyat?n kodu	**unvan-1**	**unvan-2**
Əməliyyat?n kodu	**məmbə**	
Əməliyyat?n kodu	**qəbuledici**	**məmbə**

Bu registrlər aşağıdakı funksiyalar? yerinə yetirirlər:

- **CS** əmrlər seqmenti (code segment) registri cari proqram yerləşən unvan?n? gostərir.
- **SS** stek seqmenti (stask segment) registri cari stek seqmentini unvan?n? gostərir. Stek verilənlər və unvanlar?n muvəqqəti saxlanmas? ucun yaddaş sahəsidir.
- **DS** verilənlər seqmenti (data segment) registri cari verilənlər seqmentinin unvan?n? gostərir, hans?lar ki, adətən proqramda al?nan dəyişənləri təşkil edir.
- **ES** əlavə seqment registri sətrlər uzərində əməliyyatilar?n yerinə yetirilməsində istifadə olunan cari əlavə seqmenti gostərir.

Yaddaşın istənilən mutləq unvan? registrlər seqmenti və sürüşmənin kombinasiyas? şəklində verilir. Sürüşmə novbəti 3 elementdən birinin kombinasiyas? kimi al?n?r:

- əmrin operand?nda yerləşən sürüşmə;
- BX registri;
- SI və ya DI indeks registri.

5. Assembler dilində umumi məlumat

Assembler dili yerinə yetirilən əməliyyatlar? təsvir edən operatorlar ardıcıllığıdır. Proqram?n operatorlar? kimi **əmr** və ya **psevdooperator** ola bilər. Assembler dilinin əmrləri prosessorun maşın əmrlərindən birinin simvolik təsəvvurudur. Prosessorun cəmi **90**-dan cox əmri var.

Psevdooperatorlar muəyyən edilmiş funksiyalarʔn yerinə yetirilməsi ucun translyatora gostərişdir, ən əsas proqramʔn başlanğıc və sonunu gostərirlər, verilənlərin tipini və yaddaşın bölüşdürülməsi xarakterini, informasiyanʔn çıxarılışı rejimlərini və s. elan edir. Psevdooperatorlar translyasiya mərhələsində yerinə yetirilir.

Assembler dilinin hər bir əmri novbəti şəkildə 4 sahəyə malik ola bilər:

[nişan:] Əməliyyatʔn_kodu [operandlar] [;izahat].

Beləliklə, vacib olan ancaq əməliyyat_kodudur. Əmrləri istənilən movqedən yığmaq olar, ancaq sahələr arasʔnda mutləq hec olmasa bir dənə ara olmalʔdʔr.

Əmrin nişanı 31-ə kimi simvol tuta bilər və ikinoqtə (:) ilə bitməlidir. Ona A-dan Z-ə və a-dan z-ə kimi hərflər, -. ?,...,_, $ simvollarʔdaxil ola bilər.

Nişanı **rəqəmdən başqa** istənilən simvollarla başlamaq olar, əgər noqtədən istifadə olunursa, onda o, nişanın birinci simvolu olmalʔdʔr.

Registrlərin adlarʔnişan timsalʔnda istifadə oluna bilməz. Əməliyyatʔn _kodu sahəsi mikroprosessrun əmrinin mnemonik adʔnʔ təşkil edir. Əmrlərin adlarʔ 2-6 hərfdən ibarətdir və qısaldılmış uyğun ingilis sozudur.

Məsələn, **MOV**-verilənlərin oturulməsi ***move*** sozunun qısadılmışdır, **ADD**-cəmləmə əmri ***i***-ingiliscədən ***addition***-cəmləmə deməkdir və i.a.

Proqramʔn translyasiyasʔnda uyğun mnemonik əmrlər prosessorun əmrlər sisteminə uyğun olaraq rəqəm əmrləri ilə əvəz olunurlar. Operandlar sahəsi emala aid olan verilənlərin harada tapʔlmasʔnʔ gostərir. Mnemokoddan başqa oz aralarʔnda operandlarʔayʔrmaq lazʔmdʔr.

Məsələn,

MOV CX, DX; CX←--DX

İki operandlʔəmrlərdə birinci dəyişən-**qəbuledici,** ikinci isə-**məmbədir**.

Məmbə operandʔn qəbuledici operandʔn qiyməti ilə cəmləmə, cʔxma, vurma, bolmə, muqayisə ucun mikroprosssor tərəfindən goturulən və ya qəbulediciyə yukləmə qiymətini təyin edir. Başqa sozlə, əməliyyatʔn nəticəsi qəbuledici operandda olur.

Assemblerdə şərhlər ; ilə başlayan ayrıca sətrdə yerləşdirilə bilər, operanddlar sahəsindən sonra da belə.

Məsələn,

```
MOV   SX,0 ;   SX=0
;ədədlərin cevrilməsi proqram?
;işlədi qr.647 qrupun tələbəsi İbrahimova Ayten
```

6. Psevdooperatorlar

Psevdooperatorlar proqramın əvvəlini və sonunu, seqmentlərin və s.təyin edir.

Psevdooperatorlarda 4-ə kimi sahə ola bilər.

[identifikator] psevdooperator [operand]; şərh

Vacib olan ancaq psevdooperatorun sahəsidir. İdentifikatorlar bir cox psevdooperatorlar ucun vacibdir, digərləri ucun qadağan olunmuşdur. Əmrlərdə olduğu kimi psevdooperatorların sahəsi də psevdooperatorlardan ara ilə ayrılmaq şərtilə sətiri istənilən yerində başlana bilər. Makroassemblerdə 60-a yaxın muxtəlif psevdooperatorlar nəzərdə tutulub. Cədvəl 2-də daha cox işlənən, 2 qrupa bolunən: verilənlər psevdooperatorlar? və listinqi idarə edən psevdooperatorlar? gostərilmişdir.

Cədvəl 2

Tip	Psevdooperatorlar
Verilənlən	SEGMENT ASSUME EQU ENDS END DB DW DD
Listinqi idarəetmə	PAGE SUBTTL TITLE

Cədvəl 3-də verilənlər psevdooperatorlarının təsviri verilmişdir.

Cədvəl 3

Psevdooperator	Yerinə yetirilən funksiya
EQU	Format: ad EQU mətn və ya ad EQU ədədi ifadə Mətn və ya ədədi qiymətləri mənimsəyir AD dəyişəninin ifadəsi Başqa şəkli: AD=ədədi ifadə Ola bilsin yenidən mənimsənilmişdir
DB	Format: [ad] DB ifadə [,.....] AD dəyişəni ucun yaddaşı 1 bayt ayrır və ona ifadənin qiymətini mənimsədir
DW	Format: [ad] DW ifadə [,.....] D dəyişəni ucun yaddaş? 2 bayt ayrır və ona ifadənin qiymətini mənimsədir
DD	Format: [ad] DD ifadə [,.....] Ad dəyişəni ucun yaddaşı 4 bayt ayrır və ona ifadənin qiymətini mənimsədir
SEGMENT	Format: Ad SEGMENT ... ...
ENDS	Ad ENDS
ASSUME	Format: ASSUME registr ad, [ad] Translyatora seqment registrindən hansınproqram seqmenti ilə bağlı olduğunu gostərir
END	Proqramın sonunu gostəir

7. Psevdooperatorla aid olan missalar

Gostərilən psevdooperatorların tətbiqi **misallarına** baxaq. EQU psevdooperatoru sadə adların ədədlərə və digər obyektlərə mənimsədilməsi ucun rahatdır.

Bu operator

K EQU 1024;

ALI EQU AX;

operatoruna analojidir. Bu operatorun yerinə yetirilməsindən sonra AX registri ALI kimi adlandɪrɪla bilər.

Bu operatorlar o deməkdir ki, 1024-un yerinə biz K adɪndan istifadə edə bilərik.

Standart AX adɪnɪn yerinə ALI adɪnɪ istifadə etmək olar.

«=» cəmləmə misalɪna baxaq.

A=56; A EQU 56-ya analojidir

A=57+1

Verilənlərin təyini operatorlarɪnɪn tətbiqinə baxaq.

X DB 15 ; X ucun 1 bayt yaddaşı ayɪrɪr və ona 15-i mənimsədir.

S DW 8426 : S ucun 2 bayt ayɪrɪr və 8426 qiymətini mənimsədir.

DB operatorunun koməyilə [-128, 127], DW operatorunun koməyilə isə [-32768, 32767] diapazonunda qiymətləri vermək olar.

Gostərilən operatorlardan massivlərin yuklənməsində və ilkin qiymətlərin mənimsənilməsində də istifadə etmək olar.

S DB 0,1, -15, 24, 46, 64: massivinin 6 elementinə yer ayɪrɪr

T DB 0,1, -15, 0, 0, 0, 0

DB 92, 14, 0, 0, 24, 84, 48, 24

Massivdə 15 elementinə yer ayɪrɪr, onlara ilkin qiymətləri mənimsədir və onları iki sətrdə yerləşdirir.

Bir operatorda istənilən ədədi qiymətləri gostərmək olar, təki onlar 132 movqe uzunluqlu sətrdə yerləşsinlər.

Əgər massivdə təkrarlanan qiymətlər qruplarɪ varsa, onda ixtisar ucun hər dəfə təzədən yığmadan operatorları təkrarlamağa imkan verən xususi DUP əməliyyatɪnɪ istifadə etmək olar.

Məsələn, T massivini novbəti yolla təyin etmək olar:

T DB 0,1, -15, 4 DUP (0), 92, 14

DB 0, 0, 24, 84, 48, 24

İlkin qiyməti mənimsətmədən dəyişəni təyin edərkən ifadə sahəsində ? sual işarəsini gostərmək lazımdır.

Məsələn,

A DB ? ; A dəyişəni ucun 1 bayt ayırır

B DW ? ; B dəyişəni ucun 2 bayt ayırır

Massiv ucun yaddaş ayırmaq olar:

S DW 12 DUP (?) ; massivin 12 elementinin yaddaş ayırır

DB operatorunun koməyilə massivin dəyişəninə mətn tipli qiymət mənimsədilir.

TEXT DB "qrup № 647"

SEGMENT və ENDS psevdooperatorlar? hər bir proqram? seqmentlərə bolurlər.

8. Seqmentlər

Qeyd etdiyimiz kimi, proqramda 4 seqment (hissə) ola bilər: verilənlər, əmrlər, stek və əlavə seqmenti.

Əsasən proqram iki seqmentdən: verilənlər və əmrlər seqmentlərdən ibarət olur ya da bir seqmentdan ibarətdir.

Məsələn, verilənlər seqmenti aşağıdaki kimi verilə bilər:

```
DATA SEGMENT
A   DW ?
B   DW ?
X   DW 1, 8,4, 16, 8, 14
DATA ENDS
```

Seqmentdə iki ikibaytlıq A və B dəyişənləri və ilkin verilənlərin mənimsədilən 6 elementdən ibarət bir X massivi təsvir olunur.

Əmrlər seqmenti aşağıdakı şəkildə ola bilər:

```
PROG SEGMENT
```

MOV AX, BX;

MOV SX, AX;

ADD SX, AX;

.....

Prog ENDS

SEGMENT və ENDS sozləri seqmentin başlanğıc və sonunu qeyd edirlər. Amma onlar hans? tip seqmentin muəyyən olunduğunu xəbər vermirlər. Bunun ucun ayr?ca **ASSUME** psevdooperatorundan istifadə olunur.

- ASSUME operatorunun Format?belədir:
- ASSUME seq_registri:seqmentin_ad?[,.....]

harada ki, seq_registri-DS, CS, SS, ES seqment registrlərindən birinin ad?d?r, seq_ad?isə –SEGMENT psevdooperatorunda gostərilən add?r.

Adətən, **ASSUME** operatoru birbaşa əmrlər seqmentini təyin edən SEGMENT operatorundan sonra yerləşir. Bundan başqa mutləq DS registrinə verilənlər və əmrlər seqmentinin başlanğıc unvan?n?n yuklənməsi təmin edilməlidir.

Beləliklə, əvvəlki iki seqmentdən ibarət proqram bu şəkildə olacaq:

PROG SEGMENT

ASSUME CS : PROG, DS : DATA

MOV AX, DATA

MOV DS, AX

. . .

. . .

MOV AX, BX

MOV SX, AX

. . .

PROG ENDS.

Verilənlər seqmenti dəyişməyəcək.

9. Assembler proqramın strukturu

Assembler proqramın real strukturu assembler sistemindən (MASM, TASM…), seqmentləri sayından (1 ya da birdən cox), konsol/GUİ rejimi və başğa faktorlardan asılıdır.

Biz xususi **EMU8086** sistemindən istifadə edəgiyik. Bu system assemblerdən başğa prosesorun bütün registrləri izləməyə imkan verir, proqram əddımla ilə icra imkanı yaradır və demək olar butun əsas assembler sistemlər ilə uyğun gəlir.

EMU8086 sistemində ən sadə assembler proqramların strukturu aşağıdaki kimi ola bilər. Bu proqramda **1 seqmentli** struktur istifadə olunur, yəni bu proqram **com** formatındadır.

16bit prosesorlar uyğun

```
Org 100h
Start:
Code; burada proqram
Ret
Data; burada deyishenler
End start
```

Numunə

```
. model tiny
. code
org 100h
begin:
 mov ah, 9
 mov dx,offset message
 int 21h
 ret
message db "Salam", 0dh, 0ah, '$'
end begin
```

32bit prosesorlarda bir seqmentli **exe modeli umumi strukturu**

```
.data  ; burada deyishenler
.stack
.code
Start:
; burada proqram
Ret
End start
```

EXE modelin 1 Numunəsi

```
.model small,
.stack
.data
message db "Hello everybody! I am learning assembly language!","$"
.code
main proc
   mov ax, seg message
   mov ds,ax
      mov ah,09
   lea dx,message
   int 21h
      mov ax,4c00h
   int 21h
main endp
end main
```

EXE-proqram?n 2 numunəsi

```
. model small          ; model
. stack 100h   ; stack
. code   ; code seqmenti
Begin:  ; proqram;n bashlang/
 mov ax,@data;       ; data seqmentinin unvani→ DS
 mov ds,ax
 mov dx,offset string          DX  setirin surushmenin yerleshdir.
 mov ah,9       ; ekrana gonder kodu
 int 21h          ; setiri ekrana gonder
 mov ax,4C00h        ; proqraminin bitir. kodu
 int 21h;  "proqram?bitir"
. data    ; verilenler seqmentin bashlan.
string db "Salam", 0Dh,0Ah,'$'    ; sətir
end begin       ; proqram?n sonu
```

10. Əmrlər sistemi

80x86 prosesorlarda (80286 mikroprosessoru **92)** 100 artiq əmr movcuddur. Onlar?**7** funksional qrupa bolmək olar.

1) Registrlər, yuvalar və d/x portlar? aras?nda informasiya mubadiləsini təmin edən verilənlərin oturulməsi əmrləri.
2) Riyazi əməliyyatlar?yerinə yetirən əmrlər.
3) Bitlərlə manipulyasiya əmrləri və registrlərin və yuvalar?n qiymətləri ilə məntiqi əməliyyatlar?yerinə yetirən əmrlər.
4) Proqram?n əmrlərinin ardıcıllığını idarə edən əmrləri. Bunlara kecidlər, proseduran?n çağırışı və ondan qayıdış aiddir, dovrləri təşkil və idarə edən əmrlər.

5) Sətirlərin emal?əmrləri, sətirləri muqayisə edən əmrlər.
6) Mikroprosessoru bəzi ozunəməxsus vəziyyətlərin emal?na qoşan kəsilmə əmrləri.
7) Bayraqlar?təyin edən və atan əmrləi.

Qeyd etdiymiz kimi , prosessorun əmrləri unvans?z, bir və iki unvanl?ola bilər və **1-6 bayta** kimi yer tutur.

Operandlar reqistrlərdə, əmrlərin ozundə, yaddaşda və ya d/x portlar?nda ola bilərlər.

8. Assemblərdə unvanlanma

Assemblərdə 7 unvanlanma rejimi var. Bəzi əmrlər ucun butun 7 rejim tətbiq olunur, , digərləri ucun isə ancaqbəzi rejimlər..

1-Registr unvanlama
2-Bilavasitə
3-Birbaşa
4-Yard?mc?registrli
5-Bazaya gorə unvanlama
6-İndeksə gorə birbaşa unvanlama
7-İndeksə ilə bazaya gorə unvanlama

Unvanlama rejimi haqda informasiya əmrlər rejimi sahəsində yerləşir. İlkin proqramda operandlar?n hans?şəkildə almas?ndan as?l?olaraq, Assembler bu və ya digər qiyməti rejim sahəsinə mənimsədir.

Muxtəlif unvanlama rejimlərində əmrlərin yaz?s?misal?na baxaq.

9. Reqistr, Bilavasitə və Birbaşa ünvanlanma

Registr unvanlamada bir necə numunəyə baxaq.

İkinci reqisterin qiyməti birincisinə mənimsənilir . Məsələn:

```
MOV  AX,  CX          ; AX=CX
```

Reqistrlər eyni olculu olmalıdır !!!!!!!

Bilavəstə 8 və ya 16 bitlik qiymətləri konstant məmbə operand? kimi gostərməyə imkan verir.

MOV CX, 500; 500 ədədini CX registrinə yukləyir.

MOV CL,25; 25 ədədini CL registrinə yukləyir.

İlk iki rejimlərdə tək reqistrlər istifadə olunur.

Yaddaşla əlaqə unvanlarına baxaq.

Qeyd edildiyi kimi mikroprosessorda istifadə olunan yaddaş yuvasının fiziki unvan?hesablanır.

Əmrin ozundə təkcə seqmentin başlanğıcından olan sürüşmə saxlanır.

Sürüşmə 16 bitlik işarəsiz ədəddir və 65535-ə (və ya 64K) kimi blokun istənilən baytına muraciət etməyə imkan verir. Birbaşa unvanlamada idarəedici unvan əmrin əsas hissəsidir (necə ki, iymətlər bilavasitə unvanlamada). Mikroprosessor bu icraedici unvan? DS verilənlər seqmenti registrinin icində olanlara əlavə edir və operandın 20 bitlik fiziki unvanın?alır.

Birbaşa

Məsələn, MOV AX, TABLE

TABLE yaddasının yuvasının icindəkiləri AX registrinə yukləyir.

10. Yardımc?registrli və Bazaya gorə unvanlama unvanlanma

Yardımc? reqistrli unvanlamada operandın icraedici unvan? BX baza reqistrində, BP baza gostəricisi registrində və ya indeks registrində (SI və ya DI) yerləşir. Dolay?registr operandlarını?registr operandlarından ayırmaq ucun kvadrat motərizədə yazmaq lazımdır. Məsələn:

MOV AX,[BX] əmri BX registrinin qiyməti ilə unvanlanan yaddaşın yuvasının icindəkiləri AX registrinə yukləyir.

Unvanın sürüşməsini BX registrinə necə yerləşdirmək olar?

Usullardan biri OFFSET (sürüşmə) əməliyyatı yaddaşın yuvası unvanına tətbiq etməkdən ibarətdir.

Məsələn,

```
MOV   BX ,  OFFSET   TABLE
MOV   AX,   [BX]
```

Bu iki əmr

```
MOV    AX,  TABLE
```

əmrinin eyni şəkildə yerinə yetirilir.

Sürüşmə olan digər yükləmə usulu

```
LEA   BX   TABLE ; load effective address
MOV   AX,  [BX]
```

əmridir.

Bazaya görə unvanlama

- Bazaya görə unvanlamada Assembler BX və ya BP registrlərinin icindəkilərinin sürüşmə qiymətlərini cəmləməyin koməyilə icra olunan unvanı hesablayır. $A_{исп} = A_{смещ} + A_{базы}$, buradac $A_{базы}$ baza reqistrində olan ünvan misal:

MOV AX, SOURCE[BX]

Massivin elementlərinə muraciətdə BX registrindən istifadə etmək əlverişlidir. Bu halda massivin ilkin unvanı BX baza registrinə yerləşdirirlər və massivin ayrı-ayrı elementlərinə muraciət bazaya nəzərən hərəkətlə həyata kecirilir. Məsələn, əgər massivin elementlərinin uzunluğu 4 baytdırsa və massivin başlanğıc unvanı BX registrində yerləşirsə, onda

MOV AX, [BX]+4 əmri ilə massivin **ikinci** elementinin,

MOV AX, [BX]+8 ilə ikincinin və s. ucuncu AX registrinə yükləmək olar.

Assembler bazaya görə unvanlanan operandlar 3 muxtəlif usulla gostərməyə imkan verir.

```
MOV   AX, [BP]+4
MOV   AX, 4[BP]
MOV   AX, [BP+4]
```

Bu usul əsasən massivlərlə işləyəndə istifadə olunur

Misal ucun TABLE cədvəlində soy adı (FAM 20 bayt), ad (NAME 15 bayt) və ünvan (PLACE 50 bayt). Onda

```
MOV                BX, 20
MOV                AL, TABLE[BX]
```
AL reqistirində adın birinci bayt yerləşəcək

11. İndeksə gorə birbaşa ünvanlama və İndeksə ilə bazaya gorə unvanlama

İndeksə görə **Birbaşa indeksləşmə**

$A_{исп} = A_{смещ} + A_{инд}$, burada $A_{инд}$ indeks reqistrində yerləşir.

İndeksləmə ilə bazaya gorə unvanlamada icraedici unvan baza registr indeks registri və mumkun sürüşmənin cəmi hesablanır.

Necə ki, unvanlamanın bu registrində iki ayrı-ayrı sürüşmələr yığılır, ona gorə o, iki olculu massivlərin unvanlanmasında rahatdır, nə vaxt ki, baza registrdə massivin ilkin unvanı, hərəkətin və indeks registrinin qiymətlərində isə-sətr və sutunlara gorə sürüşdürmə olur.

```
MOV Dİ,2
Mov Al, table[di]
```

Butun qeyd olunan usullar iki olculu SOURCE massivin əsasında baxaq

```
SOURCE    DB    1,2,3,4,5,6,
          DB    10,20,30,40,50,60.
```

```
MOV AX, SOURCE[SI].
```
Əğər Sİ 6 yazsaq onda AX reqistirinə 4 ədədi yəni 4 element

7. İndeksə ilə bazaya gorə unvanlama

$A_{исп} = A_{смещ} + A_{базы} + A_{инд}$,

Burada 5 və 6 üsullar birqə istifadə

```
MOV          AX, TABLE[BX+SI+const].
```

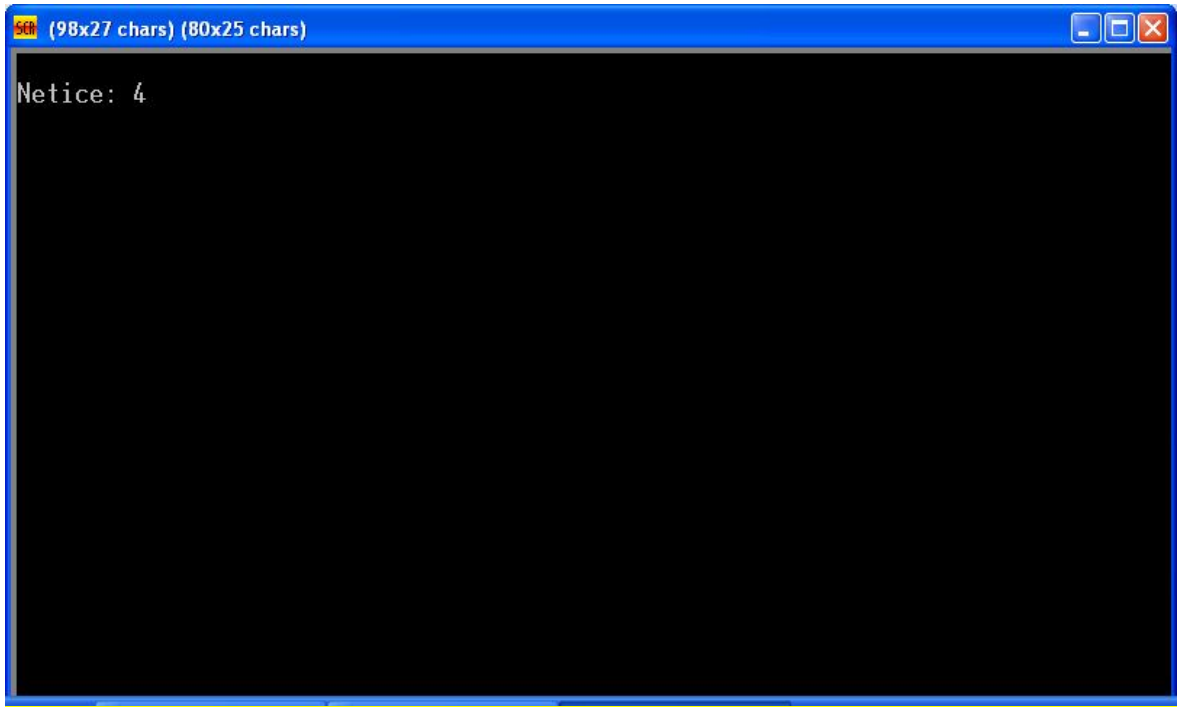
Bu usul iki olculu massivlərlə işləməyə imkan yaradır.

Burada baza ünvanı massivin başlanğic ünvanı təmin edir süruşmə və indeks reqistri muvafiq olar sətirin nömrsini vəsütun nömrəsini qöstərir.

```
MOV     AX, [BX+2+DI],
MOV     AX, [DI+BX+2]
MOV     AX, [BX+2][DI]
MOV     AX, [BX+2+DI]
MOV     AX, [BX][DI+2]
MOV     AX, NUMBER [BP][SI].
```

MISAL farida massivi ilə iş

```
include 'emu8086.inc'
ORG    100h
mov bx,1    ; indeks 1
MOV di,2    ; indeks  2
 MOV Al, farida[di+bx]
CALL   pthis
DB  13, 10, 'Netice: ', 0
CALL   print_num      ; print number in AX.
RET                   ; return to operating system.
farida  DB 1,2,13,4,5,6,10,45,67
DEFINE_SCAN_NUM
DEFINE_PRINT_STRING
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS  ; required for print_num.
DEFINE_PTHIS
DEFINE_CLEAR_SCREEN
END                   ; directive to stop the compiler.
```



12. MOV qəbuledici, mənbə

Ən çox işlənən əmrlərdən biri MOV əmridir (baytlar və ya sozlərin registrlər və ya registrlə yaddaşın yuvası arasında ötürülməsi). O, həmcinin bilavasitə unvanlanan qiyməti registrə və ya yaddaşın yuvasına ötürə bilər. Onun format?

Əmrlərin numunəsi

MOV AX, TABLE;

MOV TABLE, AX;

MOV DS, AX; 16 bitlik registrlər arasꞃnda gondərmə

MOV BL, AL; 8 bitlik registrlər arasꞃnda gondərmə

MOV CL, 30; konstantlarꞃn registrlərə gondərilməsi

MOV DD, 25; konstantlarꞃn yaddaşa gondərilməsi

1) Verilənləri yaddaşın bir yuvasꞃndan digərinə birbaşa gondərmək olmaz, əvvəlcə yuvadan registrə, sonra isə registrdən son yuvasꞃna yerləşdirmək lazꞃmdꞃr:

Məsələ, $X \rightarrow Y$

MOV AX,X

MOV Y,AX

2) Bilavasitə unvanlanan operand? **seqment** registrinə yuklənmək olmaz. Əvvəlcə umumi təyinatl? registrə yukləmək, sonra isə seqment registrinə gondərmək laz?md?r:

MOV AX, DATA

MOV DS, AX

3)Bir seqment registrinin qiymətini digərincə gondərmək olmaz. Umumi təyinatl? registrlərdən istifadə etmək laz?md?r.

13. Ədədlərin formatlar?

80x86 mikroprosessoru işarəli və işarəsiz ikilik ədədləri, həmcinin onluq ədədlərin uzərində hesab əməliyyatlar? yerinə yetirə bilir.

1. İkilik ədədlər:

İkilik ədədlər 8 və ya 16 mərtəbəli və işarəli yaxud işarəsiz ola bilərlər. İşarəsiz ədəddə 8 və ya 16 bitin şamısı qiymətlidir.

Buna gorə də işarəsiz ikilik ədədlər 0-dan 255 (8 bitlik) və ya 0-dan 65535-ə (16 bitlik) kimi qiymətlər ola bilərlər. İşarəli ədəddə boyuk bir (4-cu və ya 5-ci) onun işarəsini gostərir, yerdə qalan bitlər isə ədədin qiymətli hissəsidir. Beləliklə işarəli ədədlər -128-dən +127-ə (8 bitlik) və ya -32768-dən 32767-ə (16 bitlik) kimi qiymətlər ola bilərlər.

14. cəm əmri

İki cəm əmri vardır.

ADD (addition-cəm) və **ADC**-kocurmə ilə cəmləmə, hansılar ki, 8 və 16 bitlik operandlar cəmləyə bilərlər.

ADD əmrinin formatı:

ADD qəbuledici ,məmbə

Simvolik şəkildə bunu belə təsəvvur etmək olar: qəbuledici=qəbuledici +məmbə.

Qəbuledici və məmbənin unvanlarının yazılışında coxlu muxtəlif kombinasiyalara yol verilir, ancaq iki **operand yaddaşda yerləşdirmək və bilavasitə qəbuledici kimi istifadə etmək olmaz.**

Əmrinin ad?	ƏMR	İcra
Topla	ADD DST,SRC	(DST)←(SRC) + (DST)
Topla	ADC DST,SRC	(DST)← (SRC) + (DST) +(CF)

Məsələn,

ADD AL, BL; 8 bitlik ədədin cəmlənməsi, registr-registr

ADD AX, BX; 16 bitlik ədədin cəmlənməsi

ADD AL, BL; 16 bitlik ədədin cəmlənməsi, registr-registr-yaddaş

ADD AL, BL; yaddaş-registr

ADD AL, BL;

ADD AL, BL;

Cəmləmə əməliyyatının xususiyyəti ondadır ki, burada elə bir vəziyyət yarana bilir ki, harada ki, iki ədədin cəmi qəbulediciyə yerləşə bilmir və kocurmə problemi yaranır.

Məsələn, məlumdur ki, 8 bitlik registr 0-dan 255-ə kimi diapazonda işarəsiz qiymətlər tuta bilir. Əgər biz 200 və 100 ədədlərinin cəmlənməsini yerinə yetirsək, onda nəticə 300 ozunə gorə 9 bit tələb edəcəkdir. Amma əgər bu əməliyyatın

yerinə yetirilməsində biz 8 bitlik registrlərdən istifadə etmişiksə, onda kicik 8 bit qəbulediciyə yerləşəcək, 9-cu bit isə CF PSF keçid bayrağında yadda saxlanılacaq.

Adindir ki, qəbuledicidə olan nəticə duzgun deyil və koreksiya ehtiyac? var. Belə nəticənin təqdim olunmas?ucun 16 bitlik registr zəruridir, başqa sozlə, boyuk bitlərin təqdimi ucun bir dənə də 8 bitlik registri cəlb etmək gərəkdir.

CF-dən bu registrə kecirmək ucun ADC qəbuledici, mənbə əmrindən istifadə olunur, hans? ki, **qəbuledici=qəbuledici+mənbə+kecirmə** sxemi uzrə işləyir. Bu əmr əsasən **16b** prosessorda **32b** ədədlərin cəmlənməsi ucun təyin olunmuşdur. Doğruluğu yoxlamaq ucun, CF və SF bayraqlarına sorğu etmək lazɜmdɜr.

Operandlar (ədədlər) 8 və 16 bit ola bilər. Əğər nəticə DST yerləışmir onda prosesor **aşma** vəziyətini fiksa edir və CF bayragina 1 təyin edir.Belə vəziyyətdə 2 ki variant mumkundur

1. programın işini dayndırmaq
2. DST olcusunu artɜrmaq

İkinci variantı nəzərdən kecirdək.

Qəbul edək ki 2 8-bitlik ədəd toplanɜr və onlar AL və BL reqistrlərdə yerləşir

```
ADD AL, BL    ; nəticə 8 bitlik reqistrlərə aid
              ; olan sərhədi aşa bilər
JNC  M1       ; aşma vəziyyətini yoxlayirɜq, əğər yox onda  ->  M1:
ADC  AH,0     ; əks halda DST genişləndiririk
              ; AH reqisterrini əlavə edərək
              ;indi nəticə  16-bitli AX reqisterində yerləşəcək
M1:
```

Burada JNC əmrini istifadə etməmək olar və birbaşa ADC əmrini tətbiq etməq olar, amma nəzərə almaq lazɜmdɜr ki nəticəni AX reqisterində yerləışir. Bu yolla 32 bit olan ədədləri toplamaq olar. Burada 1 ədəd AX və BX yerləşməlidir, ikinci ədəd DX və CX =də.

Gələcəkdə fərz edəcəyik ki, butun hesab əməliyyatlarɜnɜn nəticələri daşma vəziyyəti yranmɜr və korreksiya zəruriyyət yoxdur.

Xususi **İNC** əmri movcuddur ki, registrin və ya yuvanɜn icindəkiləri **bir vahid** artɜrɜr. Bu sayğacda qiymətlərin dovru artmas?ucun rahatdɜr: Məsələn,

+1	INC OPR	(OPR)←(OPR) + 1

INC məmbə

INC CX

INC AL

14. C?xma əmrləri

C?xma oz əlavə kodlar?nda təqdim olunan operandlar?n cəmlənməsi yolu ilə yerinə yetirilir:

SUB (substruct-c?xma) və SBB (surüşmə ilə c?xma), hans?lar ki, uyğun ADD və ADC cəmləmə əmrlərinə analojidirlər.

Ancaq c?xmada kecirmə bayrağı borc əlaməti kimi hərəkət edir. SUB əmri operand-məmbəni operand-qəbuledicidən c?x?r və nəticəni operand-qəbulediciyə qaytar?r, daha doğrusu:

qəbuledici=qəbuledici-məmbə

SBB əmri də eyni işi edir, ancaq əlavə olaraq kecirmə bayrağı c?x?r. Dovrlərin təşkili ucun istifadə olunur. C?xma əməliyatlar qrupuna əlavə DEC əmri yəni bir c?x, NEG işarəni dəyişdir, və CMP muqaisə et. Əmrləri icra alqoritmi cədvələdə qostərilib

Əmrinin ad?	ƏMR	İcra
C?x	SUB DST,SRC	(DST)←(DST) - (SRC)
C?x borcdan istifadə etməklə	SBB DST,SRC	(DST)←(DST)-(SRC)-(CF)
-1	DEC OPR	(OPR)←(OPR) - 1
İşarəni dəyişdir	NEG OPR	(OPR)←(OPR) - (OPR)

Muqaisə	CMP OPR1,OPR2	(OPR1) - (OPR2)

14. Vurma əməliyyat?

İki vurma əmri vardır:

MUL (multiple-vurma) əmri-işarəsiz ədədlərin vurulması və **IMUL**-işarəli tam ədədlərin vurulması.

Hər iki əmr baytlar kimi sozləri də vura bilirlər.

Bu əmrlərin formatı belədir:

MUL məmbə

IMUL məmbə

MUL və IMUL əmrlərinin ikinci operandının timsalında AL (baytların vurulmasında) və AX (sozlərin vurulmasında) registrinin icindəkiləri istifadə edirlər.

Misal

MUL BX; BX-i AX-ə işarəsiz vurmalı

MUL T; T-ni AL-ə işarəsiz vurur

Qorundiyi kimi operandlar mutləq eyni olcudə olmalıdır. Muxtəlif olcudə olanda onları bir olcuyə qətirməq lazımdır

CBW bayt->word (8→16)

CWD word → double word (16->32)

Hər iki funksiyanın argumentləri yoxdur

CBW hesab edir ki bayt AL reqisterindədir nətici AX olacaq

CWD hesab edir ki soz AX nəticə DX:AX olacaq.

Hər iki əmr işarəsiz ədədlərlə düzqün işləməyə bilər.

MOV DX, 10

MUL DX

əmrlərin yerinə yetirilməsi nəticəsində AX-in icindəkilər 10-a vurulur.

15. Bolmə əmrləri.

2 ayr?-ayr? bolmə əmrləri var. DIV əmri (divide-bolmək) işarəsiz ədədlərin, IDIV isə işarəli ədədlərin bolunməsini yerinə yetirir. Bu əmrlərin format?:

DIV məmbə

İDIV məmbə

Burada tək bolən qostərilir, bolunən AX reqisterində olmal?d?r ya da DX:AX cut reqistrlərində .

haradak? mənbə – umumi təyinatl? registrlə və ya yaddaşda yerləşən bayt və ya soz olcudə bolucudur.

Bolunən ikiqat olcudə olmal?d?r, o, AH və AL registrlərindən (8 bitlik ədədə boləndə) və ya DX və AX registrlərindən (16 bitlik ədədə boləndə) c?xar?l?r.

Nəticələr aşağıdaki şəkildə qaytar?l?r.

Əgər operand-mənbə sozdursə, onda xususi hissə AX registrinə qal?q isə AH registrinə qaytar?l?r.

Bolmə əməllərinin bir necə tipik misallar?n? gətirək:

DIV BX; Bolməli AX da olmal?d?r

DIV T; Bolməli AH: AL da olmal?d?r.

DIV və IDIV əmrləri bilavasitə qiymətlərə bolməyə yol verilir; onu oncə registrə və ya yaddaşın yuvas?na yukləmək laz?md?r. Məsələn,

MOV BX, 20

DIV BX; bolməli DX:AX da olmal?d?r

16. Daxiletme/Xaricetme

Assembler dilində daxiletmə/ xaricetmə əməliyatlar?ucun xususi

In port

Cout port

Amma əsasən bu əməliyyatlar **int 21h** (int 33) kəsilmələrlə həyyata kecirdirlər . Kəsilmələr İnterrupt (İN) əmrləri BİOS da yerləşir .

1. simvolun monitora xaricedilməsi

```
mov ah, 2  ; KOD
MOV DL,65
İNT 21H
```

2. simvolun daxiledilmesi

```
MOV AH,1; KOD
İNT 21H ; Simvol AL reqistirine daxil olunur
```

Ədədləri daxil etmək ucun xususi funksiylar ya da makroslar yaz?mal?d?. Numunə kimi EMU8086 bir necə funksiya movcuddur və onlar xususi kitabxana emu8086.inc yerləşir

- **PRINT_STRING** – procedure to print a null terminated string at current cursor position, receives address of string in **DS:SI** register. To use it declare: **DEFINE_PRINT_STRING** before **END** directive.
- **PTHIS** – *procedure to print a null terminated string at current cursor position (just as PRINT_STRING), but receives address of string from Stack. The ZERO TERMINATED string should be defined just after the CALL instruction. For example*:

```
CALL PTHIS
db 'Hello World!', 0
```

To use it declare: **DEFINE_PTHIS** before **END** directive.

- **GET_STRING** – procedure to get a null terminated string from a user, the received string is written to buffer at **DS:DI**, buffer size should be in **DX**. Procedure stops the input when 'Enter' is pressed. To use it declare: **DEFINE_GET_STRING** before **END** directive.
- **CLEAR_SCREEN** – procedure to clear the screen, (done by scrolling entire screen window), and set cursor position to top of it. To use it declare: **DEFINE_CLEAR_SCREEN** before **END** directive.
- **SCAN_NUM** – procedure that gets the multi-digit SIGNED number from the keyboard, and stores the result in **CX** register. To use it declare: **DEFINE_SCAN_NUM** before **END** directive.
- **PRINT_NUM** - **AX** registerində yerləşən ədəd. Proqramininsonunda end operoturundan əvvəl **DEFINE_PRINT_NUM** , **DEFINE_PRINT_NUM_UNS** yerləşməlidir.
- **PRINT_NUM_UNS** – procedure that prints out an unsigned number in **AX** register. To use it declare: **DEFINE_PRINT_NUM_UNS** before **END** directive.

To use any of the above procedures you should first declare the function in the bottom of your file (but before the **END** directive), and then use **CALL** instruction followed by a procedure name. For example:

include ' emu8086.inc '

Misal

```
include 'emu8086.inc'

ORG    100h

LEA    SI, msg1       ; ask for the number
CALL   print_string   ;
CALL   scan_num       ; get number in CX.

MOV    AX, CX         ; copy the number to AX.

; print the following string:
CALL   pthis
DB  13, 10, 'You have entered: ', 0
```

```
CALL   print_num      ; print number in AX.

RET                   ; return to operating system.
Msg1   DB  'Enter the number: ', 0

DEFINE_SCAN_NUM
DEFINE_PRINT_STRING
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS  ; required for print_num.
DEFINE_PTHIS

END                   ; directive to stop the compiler.
```


```
include 'emu8086.inc'


;y=(a*x-b)/(c*x+d)


ORG   100h

GOTOXY 10,2 ; cursor to row=2 and column=10

LEA   SI, msgX      ; ask for the number  X

CALL  print_string  ;

CALL  scan_num      ; get number in CX.

MOV   X,CX

GOTOXY 10,3

LEA   SI, msgA      ; ask for the number  A
```

```
CALL   print_string   ;
CALL   scan_num       ; get number in CX.
 MOV   A,CX
 GOTOXY 10,4
LEA    SI, msgB       ; ask for the number B
CALL   print_string   ;
CALL   scan_num       ; get number in CX.
MOV   B,CX
GOTOXY 10,5
LEA    SI, msgC       ; ask for the number  C
CALL   print_string   ;
CALL   scan_num       ; get number in CX.
MOV    C,CX
GOTOXY 10,6
LEA    SI, msgD       ; ask for the number D
CALL   print_string   ;
CALL   scan_num       ; get number in CX.
MOV   D,CX
MOV AX,A
IMUL   X
SUB AX,B
MOV CX,AX
MOV AX,X
IMUL C
```

```
ADD AX,D

MOV BX,AX

MOV AX,CX

IDIV BX

GOTOXY 10,8

CALL   pthis

DB  13, 10, 'Netice: ', 0

CALL   print_num     ; print number in AX.

RET                  ; return to operating system.

Msgx   DB  'X daxil et: ', 0

msgA   DB  'A daxil et: ', 0

msgB   DB  'B daxil et: ', 0

msgC   DB  'C daxil et :' , 0

msgD   DB  'D daxil et: ', 0

X      DW  ?

A      DW  ?

B      DW  ?

C      DW  ?

D      DW  ?

Y      DW  ?

DEFINE_SCAN_NUM

DEFINE_PRINT_STRING

DEFINE_PRINT_NUM

DEFINE_PRINT_NUM_UNS  ; required for print_num.
```

DEFINE_PTHIS

DEFINE_CLEAR_SCREEN

END ; directive to stop the compiler.

17. İdarəetmə əmrləri

Bu əmrləri 3 qrupa bolmək olar-şərtsiz kecid əmrləri, şərtli kecid əmrləri, dovrlərin idarə əmrləri.

Şərtsiz kecid əmri

JMP ad; burada ad nishandir

əmrini qeyd edək, harada ki, ad-operatorun nişanıdır.

30-dan yuxar? idarəetmənin şərtli kecid əmrləri movcuddur, hans?lar ki, muəyyən şərtlərdən as?l? olaraq proqram?n yerinə yetirilmə yolunu uyğun olaraq dəyişir.

Qəbuledici ilə məmbə aras?ndak? munasibətlərlə bağlı daha çox işlənən idarəetmənin oturmə əmrlərinə baxaq:

< > < = = > = <>

Bunun ucun CMP qəbuledici, məmbə əmrindən istifadə olunur

Keçidin şərtləri	CMP-dan lter novbəti əmr	
Qəbuledici Məmbə	işarəsiz ədədlər ucun	işarəli ədədlər
>	JA	JG
<	JB	JL
>=	JAE	JGE
<=	JBE	JLE
=	JE	JE
≠	JNE	JNE

Məsələn:

```
CMP    AX, BX
JA       T1
```

Əgər AX >BX onda T1 nişanına keçid

A (above) **B (below) mənas?**

18. Dovrlər

Dovrlərin təşkili üçün **CX** sayğac registrindən istifadə olunur. Dovrlərin idarə əmri CX registrinin icindəkiləri 0-a kimi 1 vahid azalır.

Məsələn, muəyyən qrup əmrlərin yuzdəfə yerinə yetirilməsi ucun novbəti proqramdan istifadə etmək olar:

```
MOV  CX, 100; cx=100
START:
LOOP START; əgər CX 0-a bərabər deyilsə, START nişanına
```

; əks halda dovrdən c?xmal?

MİSAL DÖVRİN TƏŞKİLİ kecid əmrin vas?təsi ilə

```
include ‘emu8086.inc’
; sum calculation

ORG    100h
  mov bx,0
  MOV di,0
vag:  ADD Al, farida[di+bx]
   inc di
  cmp di,9
   jle
CALL   pthis
DB  13, 10, ‘Netice: ‘, 0
CALL   print_num      ; print number in AX.
RET                ; return to operating system.
Farida  DB 1,2,13,4,5,6,10,45,67
DEFINE_SCAN_NUM
DEFINE_PRINT_STRING
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS  ; required for print_num.
DEFINE_PTHIS
DEFINE_CLEAR_SCREEN
END                ; directive to stop the compiler.
```

MISAL LOOP vasitəsi il?dovr

```
include ‘ emu8086.inc ’
; sum calculation
```

```
ORG    100h
  mov bx,0
   mov di,0
  MOV cx,9
vag:  ADD Al, farida[di+bx]
     inc di
     loop vag
CALL   pthis
DB  13, 10, ‘ Netice: ‘ , 0
CALL   print_num      ; print number in AX.
RET                   ; return to operating system.
Farida  DB 1,2,13,4,5,6,10,45,67
DEFINE_SCAN_NUM
DEFINE_PRINT_STRING
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS  ; required for print_num.
DEFINE_PTHIS
DEFINE_CLEAR_SCREEN
END                   ; directive to stop the compiler.
```

19. C proqramlarında Assembler hissələri

C proqramlarda Assembler dilində yazılmış hissələrdən istifadə etmək olar. Onun ucun bu hissələır asm operatordan başlamalıdır

```
asm mov ax, 34
```

Əğər bir neçə operator istifadə olmalıdır onda figur motərzədən istifadə olunmalıdır

```
#include  <stdio.h>
main ()
{
int a ,b ;
a=2;
asm   {
mov ax,a
add ax,5
mov b,ax
}
printf (“%d” ,b);
}
```

```
asm mov ax,bx
asm
{
push bp
mov bp,sp
mov ax,[bp-4]
}
```

Şərhlər C formatında istifadə olmalıdır

```
asm mov ax,ds;              /* duzqun kom */
```

```
asm {pop ax; pop ds; iret;}   /* düzqün şərh */
asm push ds                   ;səhv şərh
```

Hər assembler əmri C əmri kimi qəbul olunur

```
myfunc()
{
        int i;
        int x;
        if (i>0)
                asm mov x,4
        else
        i = 7;
}
```

Assembler hissərdə bəzi məhduduiyatlar var , onlardan biri assembler nişanlardan istifadə etməq olmaz. Bu halda C nişanlardan istifadə edilir

```
#include <stdio.h>

main ()

{     int x=2,y=1,z;

     asm {

          mov ax,x

          cmp ax,y

          jle nezir

          mov ax,8

                mov x,ax

     }

     nezir:  z=x+y;

     printf("%d\n",z);

}
```

Assemblerdən C də istifadə bu dilin imkanların qenişləndirir

20. C++ da fayllar?n,direktoriyalar?n yrad?lmas?və silinməsi

Əməliyyat sistemlərinə məhsus olan fayylarla və direktoriyyalarla bəzi əməliyyatları C++ yazılmış proqramlar vasitəsi ilə kecirtmək olar.

Onun ucun xususi **stdio.h**, **dir.h** kitabxanalari movcuddur.

Kitabxanaya aşağıdaki funksiyyalar daxildir

dir.h

Chdir findfirst findnext fnmerge fnsplit **getcurdir** getcwd **Getdisk** **mkdir** mktemp **rmdir** searchpath **setdisk**

Stdio.h

Fayl?n silinməsi

```
/* remove example */
#include <stdio.h>
int main(void)
{
   char file[80];
   /* prompt for file name to delete */
   printf(“File to delete: “);
   gets(file);
  /* delete the file */
   if (remove(file) == 0)
     printf(“Removed %s.\n”,file);
   else
     perror(“remove”);
   return 0;
}
```

Onlarda bəzilərin nəzərdən kecirdək

1.**mkdir** – direkoriyan?yaratmaq v?ə rmdir –direktoriyan?sil

Funksiyalar?n prototipləri aşağıdaki kimidir

- **int mkdir(const char *path);**

- **int rmdir(const char *path);**

Funksiya direktoriyyan?n unvan?(path) qondərilir və nəticədə tam ədəd qaytar??r Əgər onun qiyməti=0 onada proses uğurla yerinə yetirilib, əks halda proses sona catmayab

Misal

```
/* mkdir    ltern */
#include <stdio.h>
#include <conio.h>
#include <process.h>
#include <dir.h>
int main(void)
{
   int stat;
char DIRNAME[]=”c:\\vag44”;
   stat = mkdir(DIRNAME);
   if (!stat)
        printf(“Directory created\n”);
   else
   {
     printf(“Unable to create directory\n”);
     exit(1);
   }
}
```

3. Direktoriyan?silməq

Silinən direktoriya boş olmalıdır, yəni butun faylar əvvəl silinməlidir

```
/* rmdir example */
#include <stdio.h>
#include <conio.h>
#include <process.h>
#include <dir.h>
int main(void)
{
   int stat;
char DIRNAME [100]=”c:\\vag44”;  //   unvan

   stat = rmdir(DIRNAME);
   if (!stat)
        printf(“\nDirectory deleted\n”);
   else
   {
  perror(“\nUnable to delete directory\n”);
     exit(1);
   }
   return 0;
}
```

21. Məntiqi disklər, ad dəyɂməsi və aktiv direktoriyalar

Mumkun say olan məntiqi disklərin siyahɂnɂn alɂnmas?setdisk (diskin nɂmrəsi)

Getdisk () – aktiv diski muyyən edir

26. **getdisk example */**

```
#include <stdio.h>
#include <dir.h>

int main(void)
{
  int disk, maxdrives = setdisk(2);
  disk = getdisk() + 'A';
  printf("\nThe number of logical drives is:%d\n", maxdrives);
  printf("The current drive is: %c\n", disk);
  return 0;
 }
```

Fayların adların deyişdiriləsi

Rename int rename(oldname, newname), əğər rename=0 ugurla sona cətɔb.

Misal

```
#include <stdio.h>
int main(void)
{
char oldname[80], newname[80];
printf("File to rename: ");
gets(oldname);
printf("New name: ");
gets(newname);
if (rename(oldname, newname) == 0)
printf("Renamed %s to %s.\n", oldname, newname);
else
perror("rename");
```

return 0;

}

Curdir Aktiv direktoriy muyyən etmək

getcurdir <DIR.H>

Gets current directory for specified drive.

Declaration

int getcurdir(int drive, char *directory);

getcurdir gets the name of the current working directory for the drive indicated by drive.

Drive Specifies a drive number (0 = default, 1 = A, etc.)

directory Points to an area of memory (of length MAXDIR) where the null-terminated directory

curdir example

/* getcurdir example */

```
#include <dir.h>
#include <stdio.h>
#include <string.h>
char *current_directory(char *path)
{
```

```
  strcpy(path, "X:\\");      /* fill string with form of response: X:\ */
  path[0] = 'A' + getdisk();    /* replace X with current drive letter */
  getcurdir(0, path+3);  /* fill rest of string with current directory */
  return(path);
}

int main(void)
{
  char curdir[MAXPATH];

  current_directory(curdir);
  printf("The current directory is %s\n", curdir);

  return 0;
}
```

–

- **getdisk returns the current drive number (0 = A, 1 = B, 2 = C, etc.)**
- **setdisk returns the total number of drives available**

22. C# dilinin əsaslar?.

C# dili Microsoft tərəfindən yaradılmış C dilinin ən yeni bir versiyas?d?r. Bu dil cox popular olan Java dilində olan Java Virtual mashine (JVM) ideyas? lternative olan NET texnologiyas?

əsas?nda yarad?l?b. Proqramci ucun əsas yeniliklər aşağıdakilardır

(Console rejimində)

1. Yeni STRİNG (sətir) **tipi.**

C++ dilində sətirlərlə işləməq ucun char massivlədən istifadə olunur

char farida="ggfhfhh";

C# bu string farida="ggfhfhh"; kimi yaz?la bilər

Və sətirlərə qiyməti mənimsətmə operator vasitəsi ilə verməq olar

a="gfggfhhjfj";

2. Massivlərin elan?dəyişib.

Bir olculu massivərin elan?

C++

Int a [10]={2,4,5};

C# da

int[] a = new int[10] { 2,4,5 };

Əğər qiymətlər əvvəlcə təyin olunmay?b onda butun elementlər =0

tip[] ad;

tip[] ad = new тип [olcu];

tip[] ad = { qiymətlər};

tip[] ad = new tip [] { qiymətlər };

tip[] ad = new tip [olcu] { qiymətlər };

```
int[] a;                                    // qiymətlər təyin olunmaya bilər
int[] b = new int[4];                       // butun elementlər=0
int[] c = { 61, 2, 5, -9 };                 // new yazılmaya bilər
int[] d = new int[] { 61, 2, 5, -9 };       // olcusu hesablanır
int[] e = new int[4] { 61, 2, 5, -9 };      // artıqlamasıilə təyinat
const int n = 6;
int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };
```

İki ölçülü massivləri elan?

```
  int[,] c1 = { { 1, 2, 3 }, { 4, 5, 6 } }; // new yazılmaib
  int[,] c2 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } }; // olcu hesablanır
  int[,] d2 = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } }; // artıq
```

3. Daxietmə/xaricetmə

Scanf/printf **cin/cout** əvəzinə **Readline()** və **WriteLine() ya da Write()** metodlardan istifadə olunur. Nəzərə alaraq ki bu usullar **Console** sinifinə daxildir, muraciyyət edəndə

Console.ReadLine(s) yada Console. WriteLine(s) yada Console. Write(s) kimi yazılmalıdır.

Burada s- sətir tipli dəyişən

Əğər başğa tip verilınlər daxil ya da xaric olunur onda xususi cevirmə funksiylar tətbiq olunmalıdır.

Convert.ToString() **– ədədi tip verilənləri String tipinə cevirir.**

Convetrt.ToDouble() **– String tipini double tipinə cevirir.**

***Convert.ToSingle()* – String tipini float tipinə cevirir.**

***Convert.ToInt32()* – String tipini int tipinə çevirir.**

Misal

```
int a;
a=Convert.Toİnt32(Console.ReadLine()); // tam ədədin daxil edilməsi
float b;
b=Convert.ToSingle(Console.ReadLine()); // həqiqi ədədin daxil edilməsi
double c;
c=Convert.ToDouble(Console.ReadLine()); // iki qat ədədin daxil edilməsi

Console.WriteLine(Convert.ToString(a));  // tam ədədin xaric  edilməsi
Alternativ variant
string  s, S1;
int  a;
s=Convert.ToString(a);
Console.WriteLine(s);
Birbaşa da olar
Console.WriteLine(a);

Sətirləri birləşdirməq ucun  + əməlliyat?istifadə olunur
S1="cavab"+s;
Console.WriteLine(S1);
```

Başğa tiplər ucun eynidir.

```
Console.ReadKey(); // ekranda nəticələrin saxlanmas?
```

4. **Yeni dovr operatoru**

```
foreach (int j in mas)
{
s = s + j + " ";
}
```

23. C# Proqram?n strukturu

```
using System;
namespace A
{
  class Class1
  {
        static void Main()
    {
      // burada proqram
          }
  }
}
```

Standart funksiyalar

- Math.Abs(x) mutləq
- Math.Exp(x) exponentx
- Math.Floor(x) aşağadan ən yax?n tam qiymət.
- Math.Log(x) natural loqorifm
- Math.Log10(x) 10 əsas?nda loqorifm
- Math.Pow(B,E) quvvət b^E

- **Math.Round(x) yuvarlanma**
- **Math.Truncate(x) Tam hissənin ay?rlanmas?**
- **Math.Ceiling(x) yxar?dan ən yax?n tam qiymət.**
- **Math.PI Pİ ədədi 3,1415...**
- **Math.E E ədədi 2,7128...**
- **Math.Sin(x) Math.Cos(x) Sinus, cosinus radianla**
- **Math.Atan(x) Arctangens radianla**
- **Math.Sqrt(x) kvadrat kok**

Misal:

$$r = \frac{(a+b)\sin x^2 + ab}{\sqrt{\cos^2\left(x+\frac{\pi}{2}\right) - b}}$$

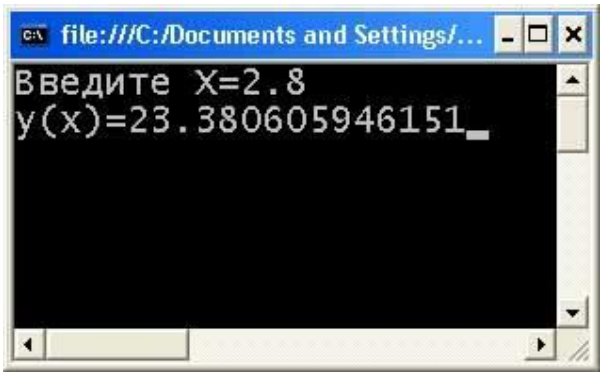
```
R=((a+b)*Math.Sin(Math.Pow(x,2))+a*b)/
   Math.Sqrt(Math.Pow(Math.Cos(x+Math.PI/2),2)-b)
```

25.

24. Xətti proqram

$$y = \frac{\frac{x}{2}\cdot e^x + \ln(1+e^x)}{\sin^2 x - \sin\sqrt{x}}$$

```
Double x,y; string s;            // elan
  Console.Write("Введите x=");   // dəvət
  s = Console.ReadLine();        // s daxil edilməsi
  x = Convert.ToDouble(s);       // cevirmə x y = (x / 2 *
Math.Exp(x) + Math.Log(1 + Math.Exp(x))) /
(Math.Pow(Math.Sin(x), 2) + Math.Sin(Math.Sqrt(x)));//
hesablama
  Console.Write("y(x)={0}",y);   // cap
Console.ReadKey();               // qozləmə
```



26. Formatlar

Xaric olunan verilənlərini formatlaçdırmaq üçün aşağılaki konstruksiylardan istifadə etməq olar:

```
…
int x=23, y=-4;
…
Console.WriteLine(“x={0}, y={1}”, x, y);
…
```

Burada **{0}**, **{1}** x və y ardıciliğını qöstərir. (sıralanma 0 başlayır). Verilənlər standart formatla şap olunur. Formatl kodlar?C++ da kimidir

d – tam ədədlər üçün

f – həqiqi

x 16 sistemində

e - exponensial format?nda

Aşağıdaki misallar formatların tətbiqini qöstərir

```
…
int a=38;
// 0038 Cap olunur
Console.WriteLine(“a={0:d4}”, a);

double pi=3.1415926;
// 3.14  Cap olunur
Console.WriteLine(“pi={0:f2}”, pi);

 int b=255;
//FF Cap olunur
Console.WriteLine(“b={0:X}”, b);

 int c=255;
// ff   Cap olunur
Console.WriteLine(“c={0:x}”, c);

double e=213.1;
//2.131000e+002   Cap olunur
Console.WriteLine(“e={0:e}”, e);
…
```

26. Budaqlanan proqram

Burada sadə budaqlanan proqram?n?təqdim olunur

```
using System;

namespace ConsoleApplication1
```

```
{   class Class1
   {   static void Main()
      {
         Console.WriteLine( "X koordinat?n?daxil et" );
         double x = Convert.ToDouble(Console.ReadLine() );

         Console.WriteLine( "y koordinat?n?daxil et " );
         double y = double.Parse(Console.ReadLine() );

                              if  ( x * x + y * y <= 1 ||
                                    x <= 0 && y <= 0 && y >= - x – 2 )
                                 Console.WriteLine( " Noqtə sahəyə düşür " );
                              else
                                 Console.WriteLine( " düşmür " );
}}}
```

27. Dovr proqram?

```
using System;
namespace ConsoleApplication1
{   class Class1
   {   static void Main()
      {
         double Xn = -2, Xk = 12, dX = 2, t = 2, y;
```

```
        Console.WriteLine( "|    x    |    y    |" );
        double x = Xn;
        while ( x <= Xk )
        {
            y = t * x;
            Console.WriteLine( "| {0,9} | {1,9} |", x, y );
            x += dX;
        }
    }
  }
}
```

28. Bir olculu massiv

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
  class Program
  {
      static void Main(string[] args)
      {
          int[] mas = {1,2,5,6,8,4,9,10};
          int i;

          string s = "for: ";
          for (i = 0; i < 8; i++) { s = s + mas[i] + " "; }

          Console.WriteLine(s);
          s = " while: ";
          i = 0;

          while (i < mas.Length)
          {
              s = s + mas[i] + " ";
```

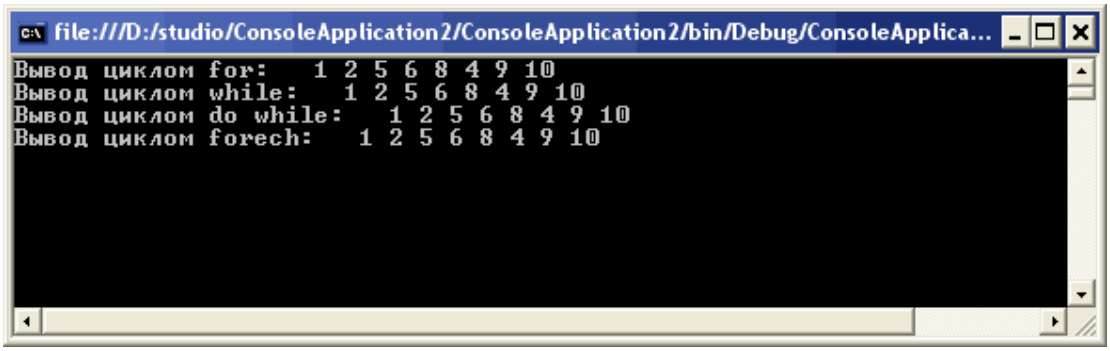
```
            i++;
        }

        Console.WriteLine(s);
        s = "do while: ";
        i = 0;

        do
        {
            s = s + mas[i] + " ";
            i++;
        }
          while (i < mas.Length);

          Console.WriteLine(s);
          s = " forech: ";

          foreach (int j in mas)
          {
              s = s + j + " ";
          }
          Console.WriteLine(s);
          Console.ReadKey();
      }
    }
}
```



29. İki olcu massiv

```
// dəyişənlərin  elan?

int i,j, M,N; string s;
```

```
// say?n? daxil edirik
Console.Write(" Sətirlərin say? N=");
s = Console.ReadLine();
N = Convert.ToInt32(s);
Console.Write("Sutunlar?n say? M=");
s = Console.ReadLine();
M = Convert.ToInt32(s);
 // Matrisan? yarad?r?q muvafiq olcudə
int[,] my_matrix= new int[N,M];
// dovrləri təşkil ediriq
for (i = 0; i < N; i++) //  Sətirlər dovru
 for (j = 0; j < M;j++ ) //sutunlar dovru
   {
     // i,j-elementini daxil ediriq
     Console.Write(" Matrisan?n elementi ({0},{1}) ",
i+1,j+1);
     s = Console.ReadLine();
     my_matrix[i,j] = Convert.ToInt32(s);
   }
Console.WriteLine();
for (i = 0; i < N; i++)
 {
   for (j = 0; j < M; j++)
     Console.Write("{0,3} ", my_matrix[i, j]);
   Console.WriteLine();
 }
```

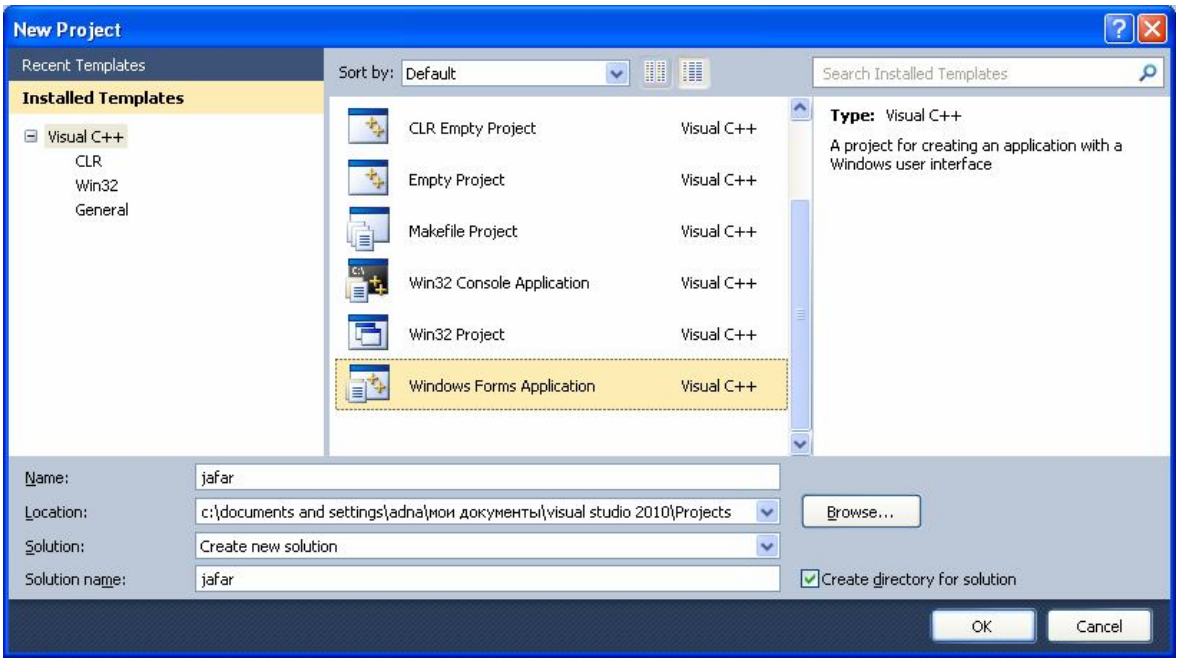
Yaxarıda baxılan proqramlaşdırma texnologiyası consol texnologiyası adlamndırılır. Orda klassik (MS DOS) dan qələn daxi/xaric etmə mexanizmi..

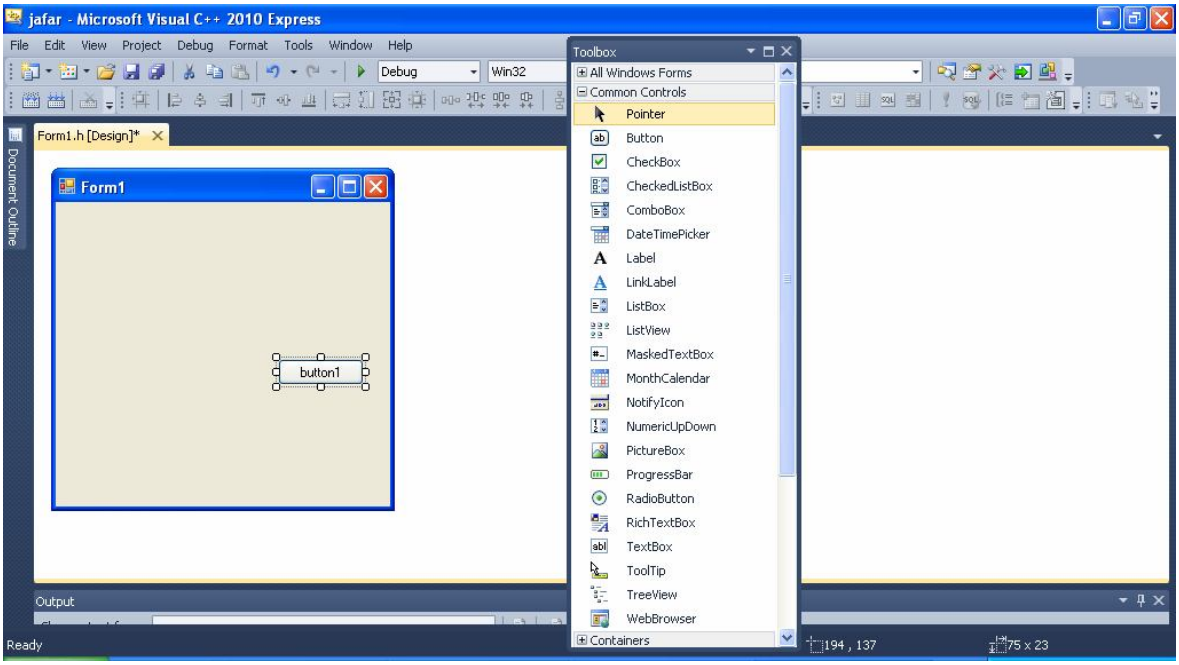
Ms Windows sistemin yaradılmsı proqramlaşdırmada böyuk dəyişıklərə qətirib. Və ilk novbədə bu daxil/xaric etmə prosesində baş verib. Yeni **Windows Form** ya GUİ (graphic user interface) usulu yeni standart kimi qəbul olunub və butun dillərdə muvafiq dəyşiliklər baş verib..

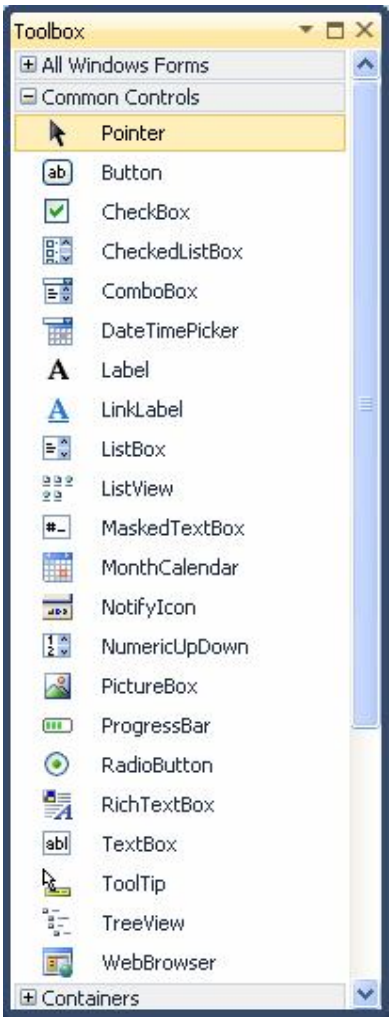
Yeni anlıaışlar yaranıb visual komponentlar (button, textbox,listbox,checkbox,radiobutton,...).

Visual proqramlaşdırması ilə bərabər yeni xadisə proqramlaşdırması (events programming)

Əsas dəyişiliklər məs daxil/xaric etmədə baş verib.




jafar - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Format Tools Window Help
Debug
Win32
Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Form1.h [Design]*
Form1
button1
Document Outline
Output
Ready
194 , 137
75 x 23



Son versiyda **Ms Visual C++ 2010** aşağıdaki sintaksisdan istifadə olunur

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)

                    {

 Burada proqram? yaz

                    }

     };
```

```
}
```

C++ da Visual proqramlaşdırma misallar? təqdim olunur

MİSALLAR

```
{
                        int j=5;
                        float ff=78.9;
                        float i = Convert::ToSingle(textBox1->Text); // textbox daxil



                        label1->Text = Convert::ToString(i); // label

                         textBox2->Text=Convert::ToString(ff);// textbox xaric
                    }
```

C# misal

```
                                    int a,b;
            a = Convert.ToInt32(textBox1.Text);
            b = a * 2;
            textBox1.Text = Convert.ToString(b);// xaric etme
        }



private void button1_Click(object sender, EventArgs e)
        {
            int b;
            string ss;
```

```
            ss = textBox1.Text;

            b = Convert.ToInt32(ss);

          label1.Text = Convert.ToString(b);// xaric etme

          textBox2.Text = Convert.ToString(b); // xaric  etme

          listBox1.Items.Add("vagif"); // xaric

          comboBox1.Items.Add("Apelsin");

          checkedListBox1.Items.Add("Mandarin");



        }



    }

}

for (int i = 0; i < 100; i++)
     comboBox1.Items.Add(i);



public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        int aa;
        private void button1_Click(object sender, EventArgs e)
        {
            int a, b;
            a = Convert.ToInt32(textBox1.Text);
            b = a * a;
            textBox2.Text = Convert.ToString(b);
            b = Convert.ToInt32(Math.Pow(a, 3));




        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            aa = Convert.ToInt32(comboBox1.SelectedItem.ToString());
            MessageBox.Show(comboBox1.SelectedItem.ToString());
            label1.Text = Convert.ToString(aa);
        }


        private void Form1_Load(object sender, EventArgs e)
        {
            for (int i = 0; i < 100; i++)
                comboBox1.Items.Add(i);
        }

        private void textBox3_MouseMove(object sender, MouseEventArgs e)
```

```
        {
            textBox2.Text = "vagif";
        }

        private void textBox3_MouseClick(object sender, MouseEventArgs e)
        {
textBox2.Text = "vagif1";
        }
    }
}
```

31. C# da fayllarla,direktoriyallrla və diskl?rlə iş

C#dilində bu məsələlərlə bağlı xüsusi siniflər (classlar) var. Onlar?n bəziləri nəzərdən kecirdək.

1. **File sinifi**. Bu sinifdə faylarla işləməkq ucun muxtəlif usullar –yratmaq, silməq, ad?n?dəyişdirməq var. Əlavə olaraq xususi **FileİnfO** sinifi movcuddur orda da əlavə bəzi əməliyytlar var.

File sinifinə aşağıdaki üsullar daxildir

» Create – fayl?yratmaq.
» Exists – faylın mövcudlığın yoxlamaq
» Delete – fayl?silməq.
» Move – fayl?kocurtməq ya da ad?n?dəyişdirməq.
» Copy – fayl?kopyalamaq.

Misallar

```
using System;
using System.IO;
namespace contest
{
   class Test
   {
      public static void Main()
      {
```

```
            // fayl?n yarad?lmas?.
            File.Create("C:\\0.txt");
            // fayl?n movcudlığın yoxlamaq.
            if(File.Exists("C:\\1.txt"))
            {
               // fayl? pozmaq.
               File.Delete("C:\\1.txt");
            }
            // adını deyişdirmeq  a.txt b.txt
            File.Move("C:\\a.txt", "C:\\b.txt");
            // fayl? kocurtməq
            File.Move("C:\\c.txt", "D:\\c.txt");
          //  fayl? kopyalamaq.
           File.Copy("D:\\z.txt", "D:\\x.txt");
       }
    }
}
```

32. Environment sinifi

Bu sinifdə butun diskləri haqq?nda məlumat əldə etməq olar GetLogicalDrives()

Usulu ilə.

Misal

```
string[] drives=Environment.GetLogicalDrives();
foreach(string s in drives)
{
   // Butun disklər haqq?nda məlumat.
   Console.WriteLine(s);
}
```

Butun direktoriylar haqq?nda məlumat

```
using System;
using System.IO;
 class AllFolders
 {
public static void WriteAllFolders(String path)
 {
```

```
// path. Direkroriyda olan butun alt direktoriyalar?cap edir
String[] di=Directory.GetDirectories(path);
// butun direktoriyalar və altdirektoroyalar ucun  adlar?cap edir.
foreach (String s in di)
{
Console.WriteLine(s);
WriteAllFolders(s);
}
}

}
//Класс для тестирования.
class App
{
static void Main()
{
// butun direktoriyalar ucun adlar cap olunur.
AllFolders.WriteAllFolders("D:\\_progs");
}
}
```

33. Directory sinifi

direktoriyllarla işləməq üçün istifadə olunur

Əsas üsullar aşağıdakilardı\
» CreateDirectory – direktoriyan?yaratmaq.
» Exists – mövcuddluğı yoxlamaq.
» Delete – silməq .
» Move – ad dəyişməq ya da köçürtməq.

А вот пример их употребления:

```
using System;
// lazЋm olan adlar fəzanı qoşuruq
using System.IO;
namespace constest
{
   class Class1
   {
      ...
      static void Main(string[] args)
      {
         // Yratmaq
         Directory.CreateDirectory("C:\\temp");
         // Mövcudluğ.
         if(Directory.Exists("C:\\temp1"))
         {
            Console.WriteLine("Папка \"temp1\" существует");
```

```
            }
            else
            {
          Console.WriteLine("Папка \"temp1\" movcud deyil  ")
Ğ           if(Directory.Exists("C:\\temp"))
            {
               Console.WriteLine("Папка \"temp\" movcuddȅr;
            }
            else
            {
               Console.WriteLine("Папка \"temp\" movcud deyil   ");
            }
            //kocurtməq
            Directory.Move( "C:\\temp", "C:\\temp2");
            // silməq
            Directory.Delete( "C:\\temp2");
        }
    }
}
```

34. DirectoryInfo sinifi

directoryİnfo sinifdə olan üsullarla bərabər əlavə direktoriyalar haqqȁnda məlumat əldə etməq olar

```
 // yratmaq
        DirectoryInfo di = new DirectoryInfo("c:\\tmp\\tmp2");

        // Создаем папку.
        di.Create();

        // məlumat.
        String s = "";
        s += "Full nane: " + di.FullName + "\n";
        s += "Root: " + di.Root + "\n";
        s += "Name: " + di.Name;
        MessageBox.Show(s);

        // oturməq
        di.MoveTo("c:\\tmp\\tmp3");

        // alt direktoriylar? yaratmaq
        di.CreateSubdirectory("subdir1");
        di.CreateSubdirectory("subdir2");
```

```
// butun alat direktoriyallar haqq?nda məlumat
DirectoryInfo [] ds = di.GetDirectories();
foreach(DirectoryInfo d in ds)
{
   MessageBox.Show(d.Name);
}

//  direktoriyalar? və altdirek. Silməq
di.Delete(true);
```