

**Коммерциялық емес
акционерлік
қоғам**



**АЛМАТЫ
ЭНЕРГЕТИКА
ЖӘНЕ БАЙЛАНЫС
УНИВЕРСИТЕТІ**

Ақпараттық
жүйелер
кафедрасы

ЖҮЙЕЛІК БАҒДАРЛАМАЛЫҚ ҚАМТАМАСЫЗ ЕТУ

5B060200-Информатика мамандығының студенттеріне
зертханалық жұмыстарды орындауға арналған әдістемелік нұсқаулықтар

Алматы 2015

ҚҰРАСТЫРУШЫЛАР: Г.С. Набиева, С.Е. Смагулова. Жүйелік бағдарламалық қамтамасыз ету. 5В060200- Информатика мамандығының студенттеріне зертханалық жұмыстарды орындауға арналған әдістемелік нұсқаулықтар – Алматы: АЭЖБУ, 2013. – 45 бет.

«Жүйелік бағдарламалық қамтамасыз ету» пәні бойынша студенттер ассемблер негіздерін меңгереді. Меңгеру кезінде компьютерде мәліметтердің берілуі, ассемблер тілінде программаны құру процесі, Turbo Debugger жөндегішімен жұмыс істеу негіздері қарастырылады. Зертханалық жұмыстарды орындау барысында студенттер жүйенің пернетақтамен өзара әрекеттесу процестерімен, символдарды және басқару кодтарын келтірумен, негізгі жүйелік функциялармен танысады.

Пікір беруші: ТКЖ кафедрасының аға оқытушысы Темырканова Э.К.

«Алматы энергетика және байланыс университеті» коммерциялық емес акционерлік қоғамының 2013 жылғы жоспары бойынша басылады.

© «Алматы энергетика және байланыс университеті» КЕАҚ, 2015 ж.

1 Зертханалық жұмыс №1. Қарапайым есептерді орындау

Жұмыстың мақсаты: бағдарламаны ассемблерлеу (аудару), құрастыру және орындау процестерін оқып үйрену, сондай-ақ ЭЕМ-де қарапайым типтік есептерді тестілеу және жөндеуді үйрену. Тәртіп бойынша, бұл есепті қою кезінде оны шешу алгоритмі анықталып қойылған. Енді бұл алгоритмді ассемблер тілінің негізгі конструкцияларын қолданып жазу керек. Жұмыстың дұрыстығын тексеру үшін керекті тестілік деректер болуы керек.

1.1 Келесі тапсырмаларды орындаңыздар

1. Экранға хабарды жолдың басынан бастап шығарыңыз.
2. Экранға хабарды экранның ортасынан бастап шығарыңыз.
3. Экранға хабарды жолдың басынан бастап және псевдографикалық кез келген символдарынан құралған рамканың ішінде шығарыңыз.
4. Алдыңғы хабарды экранның ортасына шығарыңыз.
5. Экранға символды 2h функциясы көмегімен шығарыңыз. Ол үшін кодтар сегментіне мынаны жазыңыз:

```
mov     ah, 2h           ;символды экранға шығару функциясы
mov     dl, 'A'         ;ASCII
int     21h             ;DOS үзуі
```

6. Хабарлама шығарудан бұрын экранды int 10h үзуінің 6 функциясымен тазалаңыз:

```
mov     ah, 6h         ; экранды тазалау функциясы
mov     al, 0          ;0 – бүкіл экранды
mov     ch, 0          ;жоғарғы жолдың сол жақ бұрышының нөмірі
mov     cl, 0          ;жоғарғы бағанның сол жақ бұрышының нөмірі
mov     dh, 24         ;төменгі жолдың оң жақ бұрышының нөмірі
mov     dl, 79         ;төменгі бағанның оң жақ бұрышының нөмірі
mov     bh, 0          ;байт атрибут
int     10h           ;BIOS үзуі
```

Бұл 8 команданы 9-шы жолдан кейін орналастырыңыз

7. Хабарлама шығарудан бұрын курсорды int 10h үзуінің 2 функциясының көмегімен белгілі бір орынға орналастырыңыз:

```
mov     ah, 2h         ; курсорды орналастыру функциясы
mov     bh, 0          ; ағымдағы видеопарақ
mov     dh, 5          ; жол нөмері -5
mov     dl, 10         ; баған нөмері -10
int     10h           ; BIOS үзуі
```

Бұл командаларды хабарлама немесе символ шығарудан бұрын қолданыңыз.

8. Хабарлама шығарудан бұрын `int 10h` үзуінің 6 функциясының көмегімен түрлі-түсті терезе салыңыз және `int 10h` үзуінің 2 функциясының көмегімен курсорды орнатыңыз.

1.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Ассемблер тілінде бағдарламаны құрастыру кезінде келесі кезеңдерді ұсынуға болады:

1) Блок-сұлбаны құру.

2) `NAME.ASM` бастапқы бағдарламасын құру. `NAME` - ДОС-та келісілген кез келген файлдың аты.

3) `NAME.OBJ` объектік бағдарламаны құру.

4) `NAME.EXE` орындалатын бағдарламаны құру.

5) `EXE` бағдарламаны орындау.

6) Бағдарламаның нәтижелерін тексеру.

Егер есептің талаптарына бағдарламаның нәтижелері сәйкес келмесе, онда қателерді тауып, бағдарламаны жөндеу керек.

Бағдарламаның бастапқы тексті кез келген текст редакторында құрылады. Тексттік редакторы ASCII кодаларында берілген бағдарламалардың бастапқы текстерін енгізу және түзету қамтамасыз ететін бағдарлама болып табылады.

Транслятор (компилятор) `NAME.OBJ` (машиналық тілде бағдарламаның тексті, бірақ ішкі бағдарламалардың арасындағы байланыстары жөнге келтірілмеген) объектік модулі бар дискілік файлды жасайды.

Құрастырғыш (жүктегіш) нәтижесінде алынған модульдерімен кітапханалық модульдерін құрастыруын жасайды және адрестік сілтемелерін анықтауын жасап бітеді, яғни `NAME.EXE` ығыстырылып орындалатын модульді жасайды.

Бұл жерде келтірілген Турбо Ассемблер құрамындағы компилятор, құрастырғыш және түзеткіші IBM PC микрокомпьютерлер үшін Borland International фирмасында жасалған. Ол Microsoft Corporation фирмасының MASM ассемблерімен сәйкес, бірақ одан басқа көп пайдалы кеңейтулері мен жақсартулары бар және оның шапшаңдығы одан жоғары.

Турбо Ассемблер компиляторы `TASM.EXE` файлында орналасқан орындалатын бағдарлама.

Компилятор `TASM NAME.ASM` командасымен шақырылады және `NAME.OBJ` ығыстырылатын объектік файлды жасайды. Осы командада `NAME.ASM` аударылатын файлдың аты болып табылады.

Жалпы жағдайда компилятордың шақыруы келесі түрде беріледі:

TASM fileset; fileset;...; fileset

мұнда `fileset` мынадай

options, sources, object, list, xref

түрінде беріледі, соның ішінде options - опциялардың аттары, sources - бастапқы файлдың аты немесе + (плюс) символымен немесе бос аралықпен ажыратылған осындай аттардың тізбектері, object - нәтижесінде алынған .OBJ файлдың аты, list - компиляциялау листингісі жазылатын файл аты, xref - қыйылысқан сілтемелер кестелері жазылатын файл аты болып табылады. Қыйылысқан сілтемелер кестесінде модульдің символдық аттары және олар анықталған орындардың тізімдері мен сол аттарына сілтемелері бар жолдардың нөмірлері берілген.

Мысалы,

TASM /l prim1+prim2, prim, /zi test

команда prim1.asm және prim2.asm бастапқы модульдерден prim.obj нәтижелік модульді жасайды, сонымен жол prim.lst (опция /l) файлға жазылатын компиляция листингісі, содан кейін test.asm бастапқы модульден /zi опция көмегімен test.obj нәтижелік модульді жасайды.

Орындалатын бағдарламаны құру мақсатымен .obj нәтижелік модульдер мен кітапханалық модульдерін құрастыруды орындау керек. Бұл мәселені Borland Int. фирмасының Turbo Link құрастырғышы орындайды.

Құрастырғыш TLINK.EXE файлда сақталатын орындалатын бағдарлама болып табылады.

Жалпы жағдайда TLINK құрастырғыштың шақыруы келесі түрде беріледі:

TLINK options, objects, exec, map,libraries,

мұнда options - опциялар жиынтығы; objects - нәтижелік модульдері бар файлдар аттарының жиыны; exec - орындалатын бағдарлама жазылатын файл аты; map - құрастыру картасы орналасатын файл аты; libraries - нәтижелік модульдердің кітапханасын сақтайтын файлдардың аттардың жиыны. Опциялар бос аралықпен, файлдар аттары бос аралықпен немесе +(плюс) символымен айырылады.

1.3 Жұмысты орындаудың тәртібі мен тапсырмалар

1 Компьютерге кез келген редактор көмегімен бағдарламаның бастапқы ассемблерлік текстін енгізіңіз. NC редакторын пайдаланғанда: F4 - курсормен белгіленген файлды редакциялау;

SHIFT+F4 - дискіде жоқ жаңа файлды құру, файл атын енгізу керек;

ALT+F4 - сыртқы редактор.

Көрсетілген бағдарламаны енгізіңіз:

Stacksg	segment	< 1 >
	dw	32dup(?) < 2 >
Stacksg	ends	< 3 >
Datasgt	segment	< 4 >
string	db	'Фамилия мен аты:..... ' ,13,10 < 5 >' Жасы:
	db', '\$' < 6 >
datasg	ends	< 7 >

```

codesg    segment    < 8 >
           assume    cs:codesg, ds:datasg, ss:stacksg    < 9 >
start:
           mov       ax, datasg < 11 >
           mov       ds, ax < 12 >
           mov       dx, offset string < 13 >
           mov       ah,9h < 14 >
           int       21h < 15 >
           mov       al,0 < 16 >
           mov       ah,4ch < 17 >
           int       21h < 18 >
codesg    Ends
           end       Start < 20 >

```

2 Түзетуден шыққанан соң сіздің бағдарламаңыздың ағымдағы каталогта бар екеніне көз жеткізіңіз - Name.asm. .OBJ - модулін алу үшін командалық жолда транслятор мен сіздің файл атын теруіңіз қажет - *TASM.EXE NAME.ASM*.

Тез орындау тәсілі:

- пернетақтадан TASM.EXE файлын белгілеңіз CTRL+ENTER басыңыз;
- пернетақтадан NAME.ASM файлын белгілеңіз CTRL+ENTER басыңыз.

Командалық жолдағы ақпаратты тексеріңіз. Трансляцияны орындау үшін ENTER-ді басыңыз.

Егер Ассемблер қателер туралы хабарлар берсе, онда бастапқы бағдарламаны редактор көмегімен жөндеп, қайтадан трансляция жасау керек.

3 Орындалатын бағдарламаны алу үшін, командалық жолда TLINK.EXE құрастырғыштың атын және сіздің объектік файлыңыздың атын NAME.OBJ теру керек.

TLINK.EXE NAME.OBJ

ENTER-ді басыңыз. Ағымдағы каталогта NAME.EXE файлы және сегменттер аты мен көлемін жайындағы кесте туралы мәліметі бар файл NAME.MAP пайда болады.

4 Орындалатын файлды курсормен белгілеңіз және ENTER-ді басыңыз. Бағдарламаның бұлай орындалуы анық нәтиже болғанда және оның қатесіз жұмыс істейтіні жайлы толық сенімділік болған жағдайда іске асады.

1.4 Бақылау сұрақтары

1. Ассемблер тіліндегі бағдарламаның құрылымы?
2. Бағдарламаның орындалу кезеңдері?
3. 9h және 2h шығару функциялары?
4. Бастапқы файл қалай жасалады?
5. Бастапқы файлдың кеңейтуі?
6. Транслятор аты?
7. Транслятор не істейді?

2 Зертханалық жұмыс №2. Қарапайым деректер типі түсінігі. TURBO DEBUGGER (td) жөндегіші

Жұмыстың мақсаты: деректердің қарапайым түрлерін сипаттау ережелерін және TD жөндегіш жұмысының негізгі сәттерін оқып үйрену

2.1 Келесі тапсырмаларды орындаңыздар

1 Бағдарламаның бастапқы текстін нұсқаға сәйкес теріңіз:

```
Datasg      segment
Mess        db          ' Деректердің директивалары $'

Pa          db
Pb          dw
Pc          dd
Mas        db          тапсырма нұсқаларындағы деректер
Pole       db
Adr        dw
Adr_full   dd
Datasg     ends

codesg      segment
            assume     cs:codesg, ds:datasg

start:
            mov        ax, datasg
            mov        ds, ax
            mov        dx, offset mess
            mov        ah, 9h
            int        21h
            mov        ax, 4c00h
            int        21h

codesg      ends
            end        Start
```

Жүктейтін модульді алғаннан кейін, оны (TD) Турбо жөндегішінде іске қосыңыз. DUMP терезесінде деректер сегментін қараңыз, сіздің нұсқаңызда көрсетілген барлық айнымалыларды табыңыз және осы айнымалының орналасу орны мен алатын көлемін түсіндіріңіз. Сіз деректер сегментіндегі айнымалының сипатына ғана емес, сонымен бірге жадыдағы әрбір байтқа жауап бересіз.

2 Кодалық сегментте мына командаларды теріңіз:

```
mov    Ax,    Pa          ; ax = ?
mov    Bx,    Pb          ; bx = ?
mov    Bl,    byte ptr Pb ; bl = ?
mov    Dx,    word ptr Pc ; dx = ?
```

```

mov    Cx,    word ptr Pc+2    ; cx = ?
mov    Dl,    byte ptr Pc      ; dl = ?
mov    Dh,    byte ptr Pc+1    ; dh = ?

```

Осы командалардың орындалу нәтижесін жөндегіштен қараңыз.

2.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

TASM деректерді сипаттауға және өңдеуге арналған құралдарының кең жиынтығын береді. Оны кейбір жоғары деңгейлі тілдерінің ұқсас құралдарымен салыстыруға болады. Деректердің күрделі түрлерін сипаттау үшін деректердің қарапайым түрлерін сипаттау ережелері негізгі болып табылады. Бағдарламада деректердің қарапайым түрлерін сипаттау үшін деректерді инициализациялау және резерв жасаудың арнайы директивалары қолданылады. Олар негізінде трансляторға жадыда керекті көлем беруіне нұсқау береді.

TASM келесі деректер директивасын пайдаланады:

DB (Define Byte)	көлемі 1 байт деректерге жадыда орын алу;
DW(Define Word)	көлемі 2 байт деректерге жадыда орын алу;
DD(Define Double word)	көлемі 4 байт деректерге жадыда орын алу;
DF(Define Far word)	көлемі 6 байт деректерге жадыда орын алу;
DP(Define Pointer)	көлемі 6 байт деректерге жадыда орын алу;
DQ(Define Quarter word)	көлемі 8 байт деректерге жадыда орын алу;
DT(Define Ten Bytes)	көлемі 10 байт деректерге жадыда орын алу;

Деректер директивасының келесі форматы бар:

[Аты] директиваның мнемоникасы[өрнек] [; түсініктеме]

Айнымалыға бастапқы мән бермей анықтау үшін өрнек өрісінде сұрақ белгісін (?) беру керек. Мысалы,

A db ? - инициализацияланбаған *A* айнымалысы (көлемі 1 байт физикалық жадының белгіленген бөлігінде мазмұны өзгермейді).

A db 10011011b - инициализацияланған айнымалысы (транслятор *A* айнымалына жадыда 1 байт орын береді және оған керек мәнін жазады).

Жадыда деректердің орналасуы тәртібі микропроцессордың деректермен жұмыс логикасына байланысты болады. INTEL микропроцессорлары жадыда деректердің орналасуына мына принципті қояды: *кіші байт кіші адреста*.

Turbo Debugger (TD) жөндегіш бағдарламалау көп тілдерінің, соның ішінде ассемблердің, бастапқы тексті деңгейінде бағдарламаларды жөндеудің терезелі ортасы болып табылады. Ол логикалық қатенің орнын және себебін анықтауға мүмкіндік береді.

Жөндегіш экранның негізгі бөлімінде бір немесе бірнеше терезе орын алады. Әр сәтте олардың біреуі ғана белсенді болуы мүмкін. Кез келген

терезені жандандыру үшін терезенің кез келген көрінетін нүктесіне тышқанмен шырт еткізіп іске қосады. Меню жүйесі көмегімен жөндегіштің жұмысын басқаруға болады. Осындай менюдің екі түрі бар:

- *ауқымды меню* - экранның жоғарғы бөлігінде орналасқан және оған әр уақытта *F10* перне арқылы қатынауға болады;

- *жергілікті меню* - жөндегіштің әр терезесіне оның өз менюсін шақыруға болады (*ALT+F10*).

Төрт режимнің ішінен сіз бағдарламаны қадам бойынша орындау режимінде жиі жұмыс жасайсыз. Бұл режимде сіз команда бойынша бағдарламаны орындай аласыз, сонымен бірге әр команданың нәтижесін қарай аласыз. Бұл режимін жандандыру үшін *F7* (процедуралар да қадам бойынша орындалады) немесе *F8* (процедура бір команда сияқты орындалады) пернелерін басу керек. Бұл режимде CPU терезені қолдану пайдалы, оны *VIEW | CPU* меню арқылы шақыруға болады.

Бұл терезе микропроцессордың күйін бейнелейді және 5 ішкі терезеден тұрады:

- дизассемблерленген түрде бастапқы бағдарламасын көрсететін терезе. Бұл *MODULE* терезеде көрсетілген бағдарлама, бірақ машиналық код түрінде. Қадам жөндеуді тура осы терезеде орындауға болады;

- *REGISTERS* - регистрлердің ағымдағы күйлерін көрсететін микропроцессордың регистрлерінің терезесі;

- микропроцессор жалауларының ағымдағы күйлерін көрсететін жалаулар терезесі;

- стекке берілген жадының мазмұнын көрсететін *STACK* стек терезесі;

- *DUMP* жадының дампы терезесі, ол деректер жадысының аймағының мазмұнын көрсетеді.

Аты *MODULE* жөндегіш терезесінде сіз бағдарламаның бастапқы текстін орындалу курсорымен (үшбұрыш түрінде) көресіз.

Бастапқы модульді */zi* опциямен трансляциялау керек:

Tasm /zi бастапқы_модульдің_аты

Модульді құрастыру */v* опциямен жасау керек:

Tlink /v объектік_модульдің_аты

Жөндегішті іске қосу үшін командалық жолдан бағдарламаның орындалатын модулін көрсетіп жасайды: *Td орындалатын_модульдің_аты*.

2.3 Тапсырма нұсқалары

1.	Pa	db	73H	2.	Pa	db	67H
	Pb	dw	0AE21H		Pb	dw	4AEFH
	Pc	dd	38EC76A4H		Pc	dd	12DC4567H
	Mas	db	10 dup(1),2,3		Mas	db	5,6,7,8
	Pole	db	5 dup(?)		Pole	db	6 dup(0)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc

3.	Pa	db	4DH	4.	Pa	db	5DH
	Pb	dw	0ED56H		Pb	dw	0A1A3H
	Pc	dd	32AF8DD7H		Pc	dd	3 dup(4),5,6
	Mas	db	4,3,5, 4 dup(0)		Mas	db	4,3,5, 4 dup(0)
	Pole	db	6 dup(?)		Pole	db	5 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
5.	Pa	db	62h	6.	Pa	db	0FFH
	Pb	dw	7ED1H		Pb	dw	4ADEH
	Pc	dd	0EE45DA31H		Pc	dd	0C23891F5H
	Mas	db	1,2, 6 dup(3),0		Mas	db	4 dup(0),1,2,3
	Pole	db	5 dup(0)		Pole	db	3 dup(' ')
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
7.	Pa	db	0AEH	8.	Pa	db	5BH
	Pb	dw	63BCH		Pb	dw	0BA21H
	Pc	dd	63BCDEF3H		Pc	dd	0FA4A32BCH
	Mas	db	9,8,3 dup(0)		Mas	db	4,5,6,5 dup(0)
	Pole	db	5 dup(" ")		Pole	db	6 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
9.	Pa	db	4AH	10.	Pa	db	7FH
	Pb	dw	0DEFCH		Pb	dw	0ACDEH
	Pc	dd	81ADFF06H		Pc	dd	10B0A488H
	Mas	db	5dup(1),2,3,3 dup(4)		Mas	db	3 dup(0),1,2,3, 4 dup(0)
	Pole	db	6 dup(" ")		Pole	db	5 dup(32)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
11.	Pa	db	0BCH	12.	Pa	db	0FBH
	Pb	dw	903FH		Pb	dw	54ADH
	Pc	dd	6CAA3E41H		Pc	dd	0E04365FAH
	Mas	db	1,2,3, 4 dup(4)		Mas	db	3 dup(0), 4 dup(1),2,3
	Pole	db	5 dup(?)		Pole	db	5 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
13.	Pa	db	11H	14.	Pa	db	10H
	Pb	dw	4D2DH		Pb	dw	1A2DH

	Pc	dd	98ADF156H	Pc	dd	55AEF2C8H
	Mas	db	5 dup(0),1,2,3	Mas	db	1,2,3,4,5,6
	Pole	db	3 dup(?)	Pole	db	5 dup(0)
	Adr	dw	Pc	Adr	dw	Pc
	Adr_full	dd	Pc	Adr_full	dd	Pc
15.	Pa	db	89H			
	Pb	dw	91ADH			
	Pc	dd	0AFD43E55H			
	Mas	db	5,6, 3 dup(9)			
	Pole	db	4 dup(?)			
	Adr	dw	Pc			
	Adr_full	dd	Pc			

2.4 Бақылау сұрақтары

1. Байттар жадыда қалай орналасады?
2. Сөздер және екілік сөздер жадыда қалай орналасады?
3. Сөздер және екілік сөздер қандай байттан адрестеледі?
4. Айнымалының толық адресі жадыда қалай орналасады?
5. Айнымалы мен регистр өлшемі әртүрлі болса, онда ассемблер командасында қандай операторлар қолданылады?

3 Зертханалық жұмыс №3. Бағдарламаға басқаруды беру

Жұмыстың мақсаты: сызықты емес алгоритмдерді бағдарламалаудың тәсілін үйрену.

3.1 Келесі тапсырмаларды орындау қажет

1. В массивін экранға шығару алдында мынандай тексттік жолды шығарыңыз: «Бұл екінші В массиві», яғни экранда мынандай жазу болу керек:

Бұл екінші В массиві 1 2 3 4 5.

Массив элементтерін бос орын (немесе үтір) арқылы шығарыңыз.

2. Массивтегі элементтер санын өзгертіңіз, қажетті өзгертулерді бағдарламаға енгізіңіз:

1) Бағдарлама денесінде бір цикл ұйымдастыру арқылы берілген бағдарламаны өзгертіңіз.

2) Берілген бағдарламаны сөздер массивін тасымалдау бағдарламасына өзгертіңіз.

3) Байт массивін сөз массивіне түрлендіру үшін бағдарламаны жазыңыз;

```
;деректер      сегменті
А                db          1,2,3,4,5,6
В                dw          3 dup(?)
; код           сегменті
```

(сіздің бағдарламаңыз).

4) А және В екі байт массивтерін қосу бағдарламасын жазыңыз, қосындысын С массивінде сақтаңыз.

;деректер	сегменті	
A	db	1,2,3,4
B	db	5,6,3,1
C	db	4 dup(?)

3. Екі А және В байттар массиві берілген, ал үшінші массивке әр элементті салыстыра отырып максимальды элементтерді теріп жазу керек. Бағдарлама жазыңыз, сіздің кез келген нұсқаңыз қаралады.

;деректер	сегменті	
A	DB	7,5,9,3
B	DB	4,8,2,6
C	DB	4 dup(?)

Сіздің бағдарламаңыз орындалғаннан кейін экранда келесі нәтижені алуымыз керек:

Массив C 7 8 9 6

Сәттілік тілейміз!

4. Алдыңғы бағдарламаңызды С массивіне минималды элементті іздеп орналастыратындай етіп өзгертіңіз, яғни нәтиже келесі түрде болуы керек:

Массив C 4 5 2 3

Алға!

5. Енді берілген байттар массивіндегі элементтердің максималды элементін іздейтін бағдарлама жазыңыз. Мысалы,

;деректер	сегменті	
A	DB	9,8,7,6,5,4

Нәтижеде келесі шығады:

Максималды элемент А массивінің – 9 элементі.

Ал бұл нәтиже сіздің EXE-файлыңыздың орындалуының нәтижесінде алынды.

6. Алдыңғы бағдарламаңызды берілген байттар массивінен минималды элемент іздейтіндей етіп жазыңыз.

Әрине, нәтиже басқа болады:

Минималды элемент А массивінің – 4 элементі.

Егер экрандағы нәтиже өзгермесе онда сіз қате жібердіңіз.

7. А массиві берілсін.

A	DB	1,1,-2,-4,6
	DB	2,-5,2,-2,5
	DB	3,-3,-3,4,4
	DB	5,-5,6,-7,7

Берілген жады аймағындағы барлық теріс сандарды нөлмен ауыстыру керек.

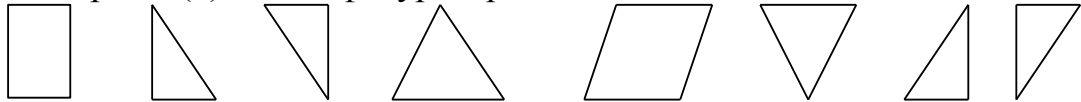
1 1 0 0 6

2 0 2 0 5

3 0 0 4 4

5 0 6 0 7

8. INT 21h 2 функциясының көмегімен текстік режимде жұлдызшалармен (*) келесі фигураларды салыңыз:



3.1 сурет– Фигуралар

9. Екілік кодта экранға шығару процедурасын жазыңыз.

10. Оналтылық кодта экранға шығару процедурасын жазыңыз.

3.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Массивтермен жұмыс жасағанда массив элементтеріне қатынау үшін индекстеумен қатар тура адресіне қолданылады. Индекстеумен қатар тура адресіне кезінде орындалатын адрес ығысу және индекс регистрінің (SI немесе DI) мәндерінің қосынды ретінде есептеледі. Ығысу массивтің басына көрсетеді, ал индекс регистрі - массив элементіне.

Мысалы, егер A - байттар массиві болса, онда келесі командалар

Mov DI,2

Mov AL,A[DI]

массивтің үшінші элементін AL регистріне жүктейді.

MOV – деректерді тасымалдау негізгі командасы. Ол регистрлердің арасында немесе регистрлер және жадының арасында әртүрлі нұсқалардың тасымалдауын орындайды.

MOV қабылдағыш, таратқыш.

Жұмыс алгоритмі:

Екінші операндты бірінші операндқа көшіру.

Команданың орындалуы жалауларға әсер етпейді.

ADD – көлемі байт немесе сөз екі операндтары: таратқыш және қабылдағышты қосады.

ADD қабылдағыш, таратқыш.

Жұмыс алгоритмі:

- қабылдағыш және таратқыш операндтарды қосу;
- қосу нәтижесін қабылдағышқа жазу;
- жалауларды орнату.

LOOP –CX регистрде санағышы бар циклді ұйымдастыру командасы.
LOOP таңба.

Цикл денесін құраушы командалардың алдында циклдің кайталау саны CX регистрдің мәнімен беріледі.

Жұмыс алгоритмі:

- CX регистрінің мәнін бірге кеміту;
- CX регистрінің мәнін талдау;

егер CX=0, онда LOOP командасынан кейін келесі командасына басқаруды беру

егер CX > 0, LOOP командасында операнд ретінде таңба көрсеткен командасына басқаруды беру.

Команданың орындалуы жалауларға әсер етпейді.

Ертеме кешпе кез келген бағдарлама орындалуын тоқтатады, қайда және қалай бару керектігінің нүктесі пайда болады. Кей кездері мұндай жағдайлар қандай да бір шартпен байланысты болуы мүмкін, ал кей кездері жай ғана басқа орынға «кетіп қалуы мүмкін. Ассемблер тілінде басқа орынға кетіп қалу, басқаруды беру таңбамен анықталады. Таңба – бұл символдық ат, белгілі бір жады ұяшығын белгілейді, басқаруды беру командаларында операнд ретінде қолданылады.

Мұндай командалардың бірі ретінде LOOP циклді ұйымдастыру командасы табылады, ол өзінен кейінгі командаға басқаруды береді, ол санағыш CX нөлге тең болғанға дейін немесе цикл ішіндегі санағышқа мән бермейтін командалар орындалған кезде басқаруды ауыстырады. Сондықтан микропроцессордың мұндай командалары атқаратын жұмыс принциптеріне байланысты топтарға бөлінеді:

- 1) Басқару тізгінін шартсыз ауыстыру командалары.
- 2) Басқару тізгінін шартқа байланысты ауыстыру командалары.

Басқару тізгінін шартсыз ауыстыру командалары:

- шартсыз ауысу командасы - *JMP таңба*;
- процедураны шақыру және процедурадан қайту командалары - *CALL процедура_аты* және *RET*;

- бағдарламалық үзулерді шақыру және - *INT үзу_нөмірі*, *IRET* - бағдарламалық үзулерден қайту, *INTO* – асатолу болған кездегі үзу.

Басқару тізгінін шартқа байланысты ауыстыру командалары:

а) салыстыру командасының нәтижесіне байланысты ауысу командасы: *CMP операнд1, операнд2*.

Жұмыс алгоритмі:

- алуды орындау (операнд1- операнд2);
- нәтижеге байланысты белгіні орналастыру, операнд1 және операнд2

өзгертпеу керек (яғни, нәтижені сақтаудың керегі жоқ).

JE *Операнд1= операнд2* *JNE* *операнд1 <> операнд2*

Таңбасыз Таңбалы Шартты ауысу критерилері

JB/JNAE *JL/JNGE* *операнд1 < операнд2*

JBE/JNA *JLE/JNG* *операнд1 <= операнд2*

JA/JNBE	JG/JNLE	операнд1 > операнд2
JAE/JNB	JGE/JNL	операнд1 => операнд2

б) анықталған бір белгі күйіне байланысты өту командалары

JC	(JNC)	Ауысу_таңбасы
JP	(JNP)	Ауысу_таңбасы
JZ	(JNZ)	Ауысу_таңбасы
JS	(JNS)	Ауысу_таңбасы
JO	(JNO)	Ауысу_таңбасы

в) CX регистрінің құрамына байланысты ауысу командасы

JCXZ	Ауысу_таңбасы
------	---------------

3.3 Жұмыстың орындалу тәртібі

А байттар массивін В байттар массиві орнына тасымалдау бағдарламасы берілген.

```

sseg      segment
          Db          128 dup(?)
sseg      ends
dseg      segment
A         Db          1,2,3,4,5
B         Db          5 dup(?)
dseg      ends
cseg      segment
          assume      ss:sseg, cs:cseg, ds:dseg

start:
          mov         ax,dseg
          mov         ds,ax
          mov         si,0
          mov         cx,5

M1:
          mov         al, A[si]
          mov         B[si],al
          inc         si
          loop        M1
          mov         cx,5
          mov         si,0

M2:
          mov         ah,2h
          mov         dl,B[si]
          int         21h
          inc         si
          loop        M2

```

```

        mov     ah,4ch
        int     21h
cseg    ends
        end     start

```

3.4 Бақылау сұрақтары

1. Қандай шартты ауысу командалары таңбалы сандармен жұмыс кезінде қолданылатын?
2. Қандай шартсыз ауысу командалары кейбір команда топтарын айналып өтеді?
3. Қандай командалар цикл ұйымдастыру кезінде қолданылады?
4. Қандай командалар бағдарламаны шақыру үзуді өңдеу және одан қайтуды орындайды?
5. Қандай командалар процедураны (ішкібағдарламаны) шақыру және одан қайтуды орындайды?

4 Зертханалық жұмыс №4. Процессор командаларының әртүрлі топтарын қолдана отырып бағдарламаларды өңдеу

Жұмыстың мақсаты: жекеленген биттермен жұмысты ұйымдастыруды оқып үйрену, тізбектелген командалардың жалпы қасиеттерін оқып үйрену және екілік - ондық арифметика командалары.

4.1 Келесі тапсырмаларды орындаңыздар

1. Берілген сөздегі немесе байтағы бірлердің санын санайтын бағдарлама жазыңыз: деректер сегменті

```
A      DB  01101011B      немесе      A      DW  0010111110101001B
```

2. Бағдарламаны нөлдерді санайтындай етіп өзгертіңіз.
3. Таңбалы сандардан тұратын A массивін беріп. Осы массивтен теріс сандарды санайтын бағдарлама жазыңыз. Сіздің кез келген нұсқаңыз қабылданады. Осыны TEST командасын қолданып қайталаңыз.
4. Бағдарламаны массив элементтерінің оң сандарын есептейтіндей етіп өзгертіңіз.
5. Жадыдағы сөз ретіндегі екі жинақталмаған BCD сандарын жинақталған AL регистріндегі BCD санына ауыстыру бағдарламасын жазыңыз.
6. Символдарды бір жолдан екінші жолға ауыстыру бағдарламасын жазыңыз: деректер сегменті

```
str_source      DB      'Ауыстырылатын таратқыш жол
                    жол','$'
str_dest        DB      20 dup(?)      қабылдағыш жол
```


7. Екі жолды бір символдан (бір элементтен) салыстыратын бағдарлама жазыңыз.

8. Берілген жолдан белгілі бір символды іздеу бағдарламасын жазыңыз, бұл символды алдын ала жадыда анықталған немесе пернетақтадан берілген басқа символмен ауыстырыңыз.

9. Екі жолды салыстыру бағдарламасын жазыңыз, ондағы бірінші ұқсас емес элементті AL регистріне орналастырыңыз.

10. Формула бойынша есептейтін бағдарлама жазыңыз: $y = (a+b)^3 / (c-d)^2$.

4.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Қосу командалары

ADD қосу командасы. Команданың жазылу форматы:

ADD қабылдағыш, таратқыш

ADC тасымалы бар қосу командасы. Команданың жазылу форматы:

ADC қабылдағыш, таратқыш

ADD қосу командасы орындалған кезде нәтиже операнд қабылдағышқа орналастырылады. *ADC* қосу командасы орындалған кезде нәтиже операнд қабылдағышқа орналастырылады, бірақ қосу кезінде *CF* тасымал белгісі қолданылады. Сонымен қатар *ADD* және *ADC* командалары: *CF*, *PF*, *AF*, *ZF*, *SF*, *OF* белгілерін пайдалана алады.

AAA командасы қосу нәтижесін *ASCII* кодта беру үшін қалпына келтіреді. Ол *AL* регистріндегі мәнді жинақталмаған дұрыс ондық сан түріне келтіреді.

Команда келесі контексте қолданылады:

ADD AL,BL

AAA

DAA командасы қосу нәтижесін ондық формада беру үшін қалпына келтіреді. Ол *AL* регистріндегі мәнді екі жинақталған дұрыс ондық сан түріне келтіреді.

Команда келесі контексте қолданылады:

ADD AL,BL

DAA

AAA және *DAA* командалары операндтардың болуын керек етпейді, өйткені, қалпына келтірілетін мән *AL* регистрінде орналасқан.

INC командасы (1-ге өсіру) жады ұяшығын немесе регистр мәнін бірге өсіріп отырады.

Команданың жазылу форматы: *INC* қабылдағыш

Бұл команда жады ұяшығында тізбектей орналасқан индекстік регистр немесе көрсеткішке қатынау кезінде, циклдардағы санашық мәнін өсіріп отыру кезінде өте ыңғайлы болып табылады.

Бұл команда келесі белгілерді өзгертеді: *PF*, *AF*, *ZF*, *SF*, *OF*.

Алу командалары

SUB алу командасы (азайту).

SBB командасы – несиемен қоса алу.

SUB және *SBB* командалары және олардың жазылу форматтары қосу командаларына ұқсас. Ол да алты белгіні өзгерте алады.

AAS командасы алу нәтижесін ASCII кодта беру үшін қалпына келтіреді.

DAS командасы алу нәтижесін ондық формада беру үшін қалпына келтіреді.

AAS және *DAS* командалары, *AAA* және *DAA* командаларына ұқсас. Команда келесі контексте қолданылады:

SUB AL,BL

AAS немесе *SUB AL,BL*

DAS

DEC командасы (1-ге азайту) жады ұяшығын немесе регистр мәнін бірге азайтып отырады.

NEG терістеу командасы. Нөлдік мәннен операнд – қабылдағыш мәнін азайтады. Айталық 100-ден *AL* регистрінің мәнін азайту керек болсын. Мұндағы 100 саны қабылдағыш операнды ретінде жүрмейтін болғандықтан, жазба

SUB 100, AL деп жазуға болмайды. Ол үшін мұндай азайтуды ұйымдастыру керек болса келесі командаларды жазу қажет:

NEG AL

ADD AL,100

Бұл команда келесі белгілерді өзгертеді: *PF*, *AF*, *ZF*, *SF*, *OF* және *CF* = 1, егер операнд нөлдік оң сан болса, басқа жағдайда олар 0-ге тең.

CMR командасы (салыстыру) – нәтижеге байланысты таратқыштан қабылдағышты азайтады, келесі белгілерді нөлдейді немесе орналастырады, ал қабылдағыш пен таратқыш мәндері өзгермейді.

Бұл команда келесі белгілерді өзгертеді: *PF*, *AF*, *ZF*, *SF*, *OF* және *CF*.

Көбейту командалары

MUL таңбасыз сандарды көбейту командасы. Команданың форматы: *MUL* таратқыш.

IMUL таңбасыз бүтін сандарды көбейту командасы. Команданың форматы: *IMUL* таратқыш.

Таратқыш ретінде екілік сөз немесе сөз, байт өлшеміндегі жады ұяшығы немесе жалпы міндетті регистрді алуға болады.

Ал екіші операнд ретінде *AL* (байт үшін), *AX* (сөз үшін) немесе *EAX* (екілік сөз үшін) регистрлерін қолдануға болады. Туынды екі өлшемді орын алады және келесідегідей қайтады: байттарды көбейту 16-биттік туындыны *AH* (үлкен байт) және *AL* (кіші байт) регистрына қайтарады; сөздерді көбейту 32-биттік туындыны *DX* (үлкен сөз) және *AX* (кіші сөз) регистрына қайтарады екілік сөздерді көбейту 32-биттік туындыны *EDX* (үлкен екілік сөз) және *EAX* (кіші екілік сөз) регистрына қайтарады.

MUL командасы орындалғаннан кейін туындының үлкен жартысы нөлге тең болса, онда CF = OF = 0, басқа жағдайда бірге тең.

IMUL командасы орындалғаннан кейін туындының үлкен бөлігі кіші бөліктің кеңейтілген таңбасын ғана көрсетсе, онда CF = OF = 0, басқа жағдайда бірге тең.

AAM командасы көбейту нәтижесін ASCII кодта беру үшін қалпына келтіреді. Ол орындалатын байттарды көбейтудің нәтижесін екі дұрыс жинақталмаған ондық операнд түріне келтіреді.

Бөлу командасы

DIV таңбасыз сандарды бөлу командасы. Команданың форматы: DIV таратқыш.

IDIV таңбалы сандарды бөлу командасы. Команданың форматы: IDIV таратқыш.

Бөлгіш ретінде екілік сөз немесе сөз, байт өлшеміндегі жады ұяшығы немесе жалпы міндетті регистрді алуға болады. Бөлінгіштің екілік размері болуы керек, ол AH және AL (8-биттік санға бөлу кезінде) регистрлерінен, DX және AX (16-биттік санға бөлу кезінде) регистрлерінен немесе EDX және EAX (32-биттік санға бөлу кезінде) регистрлерінен алынады. Нәтиже келесідегідей қайтарылады: егер таратқыш байт болса, онда бөлінді AL регистріне, ал қалдық AH регистріне қайтарылады. Егер таратқыш сөз болса, онда бөлінді AX регистріне, ал қалдық DX регистріне қайтарылады және егер таратқыш екілік сөз болса, онда бөлінді EAX регистріне, ал қалдық EDX регистріне қайтарылады.

AAD командасы бөлу нәтижесін ASCII кодта беру үшін қалпына келтіреді. Ол бөлу операциясының алдында орындалады. Бұл команда жинақталмаған бөлінгішті екілік санға ауыстырады да оны AL регистріне жүктемелейді.

Логикалық командалар және ығыстыру командалары.

Логикалық және ығыстыру командалары разрядтармен жұмыс жасауға мүмкіндік береді. Ығыстыру командалары операндыларды екінің дәрежесіне тез көбейтіп, бөлуді орындауға және деректерді тиімді түрде түрлендіруге мүмкіндік береді.

AND - байт немесе сөз өлшемі бар қабылдағыш және таратқыш операндтардың разрядтарын логикалық көбейту (конъюнкция) ЖӘНЕ командасы.

Команданың форматы: AND қабылдағыш, таратқыш

OR - байт немесе сөз өлшемі бар қабылдағыш және таратқыш операндтардың разрядтарын логикалық қосу (дизъюнкция) НЕМЕСЕ командасы.

Команданың форматы: OR қабылдағыш, таратқыш

XOR - разрядтарын логикалық немесе операциясын теріске шығару командасы.

Команданың форматы: XOR қабылдағыш, таратқыш

Нәтиже келесі операцияның шындық кестеде көрсетілгендей орындалады:

Қабылдағыш	Таратқыш	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

TEST – («test»-тексеру). Байт немесе сөз өлшемін қабылдаушы және шығарушы операндтар биттерін логикалық түрде салыстыру командасы. Бірақ операндтар күйлері сақталады, өзгертін тек ZF, SF және PF жалаулары. Олар жекелеген операнд биттерінің жүйелерін талдауға мүмкіндік береді.

Команданың форматы: *TEST* қабылдағыш, таратқыш.

NOT- Байт немесе сөз өлшемін таратушы операндтың әр битін логикалық түрде терістеу командасы.

Команданың форматы: *NOT*-таратқыш. Таратқыш операндының барлық биттерін терістеу (инвертировать): 1-ден 0-ге, 0-ден 1-ге ауыстыруды орындайды. Команданың орындалуы жалауларға әсер етпейді.

Мысалдар.

1. *mov AL, 1111 0000b* 2. *mov AL, 1111 0000b* 3. *mov AL, 1111 0000b*
 and AL, 1010 1010b *or AL, 1010 1010b* *xor AL, 1010 1010b*
 нәтиже 1010 0000b ; нәтиже 1111 1010b ; нәтиже 0101 1010b

4. *mov AL, 11110000b* 5. ; санды жұптыққа тексеру
 not AL ; AL=00001111b *test AL, 0000 0001b*
 jz m ... ; тақ сан m: ; жұп сан

Мысалда көрсетілгендей *AND* командасы қабылдағыш битін 0 орнатады, сәйкесінше таратқыштың биті нөлге тең, *OR* командасы – қабылдағыш битін 1 орнатады, сәйкесінше таратқыштың биттері 1 тең, *XOR* – командасы қабылдағыш биттерін терістейді, сәйкесінше таратқыш биттері де терістеледі.

Жылжыту командалары.

Ығыстыру командаларының әсер ету принципі бойынша екі типке білуге болады:

- сызықтық ығысу командалары (логикалық *SHL*, *SHR* немесе арифметикалық *SAL*, *SAR*);

- циклдық ығысу командалары (қарапайым *ROL*, *ROR* және жалау тасымалы бойынша *RCL*, *RCR*).

Барлық ығысу командалары операция кодасына байланысты операнд өрісіне беттерді солға немесе оңға жылжытады. Ығысу командаларының барлығының пішіні бірдей: бірінші операнд болып операнд-шығарушы, екінші операнд болып ығысу саны табылады. Ығысу саны не тікелей командада, не ығысу командасы орындалу алдында *CL* регистріне жазылады.

Мысалы:

```
MOV CL,4          .386
SHL AX,1          SHL AX,CL      SHL AX,4(үлкен процессорлар үшін).
```

SHL - логикалық солға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады. Оң жаққа (кіші байт позициясына) нөлдер жазылады.

SHL операнд, ығысулар-санауышы

SHR - логикалық оңға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады. Сол жаққа (үлкен байт позициясына) нөлдер жазылады.

SHR- операнд, ығысулар-санауышы

SAL - арифметикалық логикалық оңға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады. Сол жаққа (үлкен байт позициясына) нөлдер жазылады.

SAL - операнд, ығысулар-санауышы

SAL командасы таңбаны еске сақтамайды. *SAL* командасы толығымен *SHL* командасына ұқсас.

SAR - арифметикалық логикалық оңға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына оңға ығыстырады. Сол жаққа (үлкен байт позициясына) нөлдер жазылады, егер оң сандар жылжыса нөлдер жазылады, сол сан жылжыса бір жазылады.

SAR операнд, ығысулар_санауышы

SAR - командасы таңбасын, әр кезекті битті ығыстырған сайын қалпына келтіру арқылы сақтап отырады.

ROL - циклдің солға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады. Солға ығыстырылған биттер сол операндқа оң жағынан жазылады.

ROL- операнд, ығысулар-санауышы

ROR - циклдің оңға ығысу командасы, ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына оңға ығыстырады.

ROR- операнд, ығысулар-санауышы

RCL - тасымал жалауы арқылы циклдық солға ығысу командасы ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады, ығысқан биттер өз кезегі бойынша *CF* тасымал жалауының мәні болады.

RCL- операнд, ығысулар-санауышы

RCR - тасымал жалауы арқылы циклдық оңға ығысу командасы ол операнд мазмұнын ығысу санаушының мәнімен анықталатын биттер санына солға ығыстырады, ығысқан биттер өз кезегі бойынша *CF* тасымал жалауының мәні болады.

RCR- операнд, ығысулар-санауышы

Жолдарды өңдеу командалары.

Микропроцессордың командалар жүйесінде 1 байттан 64К байтқа (32-разрядтық микропроцессорлар үшін -4 Гбайтқа дейін) дейінгі элементтер блогында әрекет жасауға мүмкіндік беретін өте қызықты командалар тобы бар. Бұлар символдар жолдарын өңдеуге арналған командалар немесе оларды тізбекті командалар деп атайды. Бұл блоктар логикалық тірде әртүрлі мәндер қабылдайтын элементтер тізбегін құруы мүмкін, олар жадыда екілік кодалар түрінде сақталады. Микропроцессор әр кезде қабылдағыш-жол қосымша сегментте (ES сегменттік регистрі көмегімен адрестелетін), ал таратқыш-жол деректер сегментінде (DS сегменттік регистрі көмегімен адрестелетін) болады деп есептейді. Микропроцессор қабылдағыш-жолды DI регистрі арқылы, ал таратқыш-жолды SI регистрі арқылы адрестейді. Қабылдағыш-жолға арналған сегментті алдын ала анықтаға болады, ал таратқыш-жолға мұны пайдалануға болмайды.

Пайдаланатын тізбекшелік командаларға байланысты DI және SI регистрлерінің мазмұнының өсуін немесе кемуін автоматты түрде іске асырып отырады. Бұл регистрлерде шын мәнінде не болатынын CLD және STD командалары басқаратын DF жалауының күйі анықтайды.

Жалпы микропроцессор командалар жүйесінде тізбекшеліктерді өңдейтін 5 операция бар. Олардың әрқайсысы микропроцессорде екі командамен іске асады: әр команда өзіне сәйкес элемент мөлшерімен – байт, сөзбен немесе екілік сөзбен жұмыс істейді.

Жолдарды жіберу:

MOVS қабылдағыш_адресі, таратқыш_адресі =>	MOVSB MOVSW MOVSD
--	-------------------------

Жолдарды салыстыру:

CMPS қабылдағыш_адресі, таратқыш_адресі =>	CMPSB CMPSW CMPSD
--	-------------------------

Жолда іздеу:

SCAS қабылдағыш_адресі =>	SCASB SCASW SCASD
---------------------------	-------------------------

Жолдағы элементті жүктеу:

LODS таратқыш_адресі =>	LODSB LODSW LODSD
-------------------------	-------------------------

Элементті жолда сақтау:

STOS қабылдағыш_адресі =>	STOSB STOSW STOSD
---------------------------	-------------------------

4.3 Бақылау сұрақтары

1. Көбейту командасын орындаған кезде көбейткіш мен көбейтінді қайда орналасады?
2. Бөлу командасын орындаған кезде бөлінгіш, бөлгіш және нәтиже қайда орналасады?
3. Тізбекті командаларда жолдың ұзындығы қандай регистрде сақталады?

5 Зертханалық жұмыс №5. Қолданбалы бағдарламада жүйелік функцияларды қолдану

Жұмыстың мақсаты: пернетақтадан енгізу, экранға шығару, DOS және BIOS құралдарының көмегімен файлдармен жұмыс жасау және оларды пайдалануды оқып үйрену.

5.1 Келесі тапсырмаларды орындаңыздар

Бағдарлама жазу керек.

1. INT 21h 0Ah (INT 21h 3Fh) функциясы көмегімен пернетақтадан жолды енгізу.
2. Күпиясөзді енгізу және оны тексеру (INT 16h 00 функциясы, INT 21h 07, 08, 3Fh функциялары көмегімен).
3. ENTER (F1, Home) пернесін басуды тексеру.
4. Пернені басу арқылы бағдарламаны тоқтату және жалғастыруды ұйымдастыру (кез келген, нақты).
5. Жолды енгізу және 0Dh байтын 0-ге алмастыру.
6. INT 16h 01, INT 21h 01 функциялары көмегімен пернетақтадан жолды енгізу.
7. Int 21h 2 (Int 10h 9) функция көмегімен символды шығару.
8. Int 21h 2 (Int 10h 9) функция көмегімен символдар тобын шығару.
9. Массивті экранға шығару.
10. Int 10h 10 (Int 10h 14, Int 10h 9) функция көмегімен горизонталь сызықты шығару.
11. Int 21h 9 (Int 21h 40h, Int 21h 2, Int 10h 10) функция көмегімен тексттік жолын экранға шығару.
12. Файлға тексттік жолды жазу.
13. Файлдың ұзындығын анықтау.
14. Файлды оқу және оның ішіндегіні экранға шығару.
15. Файлдағы 10-байтты ауыстыру.
16. Файлдағы 5-байтты оқу және оны экранға шығару.
17. Бір файлды екіншіге көшіру.
18. Файлды 20-шы байттан бастап екінші файлға көшіру.
19. Файлды оқу және соңғы байтқа бақылау қосынды санын жазу.

20. Екі файлды ұқсастыққа салыстыру.
21. Екі файлды біріктіру.
22. Берілген байттың нөмірін анықтау.
23. Файлға басқа ат беру және файлды құрған уақыт пен күнін (датасын) орнату.
24. Файлға басқа ат беру және оның атрибутын ауыстыру.

5.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Пернетақтадан енгізу жүйелік функцияларына шолу.

Пернетақтадан енгізу функциялары INT 16h (0, 1 функциялары) және INT 21h (1, 7, 8, 0Ah, 3Fh функциялары) үзулерінің көмегімен жүзеге асады.

INT 16h 00h функциясы пернені басуды күтеді.

Перне басылғаннан кейін (егер ASCII кодасы бар символды перне басылған болса) AL регистріне перненің ASCII-кодасы немесе (егер функциялық перне басылған болса) 0 жазылады, ал AH регистріне перненің скан - кодасы жазылады.

AL регистрінің мәніне қарай символды немесе функциялық перненің басылғандығын анықтауға болады.

KEY:	MOV	AH, 0	; пернені басуды күту
	INT	16h	; BIOS- ты шақыру.
	CMR	AL, 0	; функциялық пернені
	JE	M1	; басуды тексеру
	JMP	KEY	; функциялық перне емес
M1:		; функциялық перне

INT 16h 01h функциясы пернетақтаның аралық жадысының күйі жайлы ақпаратты оқуды жүзеге асырады, егер аралық жады бос болса, онда нөлдік жалаушасында 1-ді қайтарады (ZF=1), егер аралық жады бос болмаса, нөлдік жалаушасында 0-ді қайтарады (ZF=0). Ал AX регистріне келесі символды, қалғанын аралық жадыға жазады.

INT 21h 01h функциясы пернетақтадан символды енгізуді орындайды. Оны экранға кескіндейді және Ctrl-Break пернелерін тексеруді орындайды. Символды енгізгеннен кейін AL-де символдың ASCII кодасы болады.

```
MOV    AH, 1h
INT    21h
```

INT 21h 07h функциясы пернетақтадан символды экранға кескіндемей енгізуді орындайды. Функция орындалғаннан кейін AL регистрінде символдың ASCII кодасы болады.

```
MOV    AH, 7h
INT    21h
```

INT 21h 08h функциясы 07h функциясына ұқсас, одан басқа Ctrl-Break пернелерін тексеруді орындайды.

```
MOV    AH, 8h
```


INT 21h

01, 07, 08 функцияларының көмегімен функциялық пернелерді оқу үшін перненің кодасын оқудың екі түрлі амалын орындау керек. Бірінші оқу операциясы AL регистріне 0, екінші оқу операциясы AL регистріне скан-кодасын (кеңейтілген кодасын) енгізеді.

```
KEY: MOV AH, 8h ; перненің басылуын күту
      INT 21h
      CMP AL, 0 ; кеңейтілген кода ма?
      INZ ERROR ; жоқ болса қате жайлы хабарлау
      MOV AH, 8 ; скан коданы оқу
      INT 21h
      CMP AL, 33h ; F1 пернесі басылды ма?
      JE F1 ; ия
      JMP KEY ; жоқ
      ERROR: .....
```

INT 21h 0Ah функциясы пернетақтадан жолды аралық жадыға енгізуді орындайды. Аралық жадының адресі DX регистріне жүктеледі. Деректер сегментінде аралық жадыны сипаттап, оның өлшемін беруі қажет. Аралық жадының бірінші байтта өлшемі болу керек. Функция орындалғаннан кейін аралық жадының екінші байтта бағыттағышты кері қайтару кодасымен аяқталатын, енгізілетін жолдың нақты өлшемі болады.

```
; деректер сегментінде
      BUF DB 22, 21 DUP (?)
; кода сегментінде
      MOV AH, 0Ah
      LEA DX, BUF
      INT 21h
```

Енгізілген жол жадыда BUF+2 адресінен басталады.

INT 21h 3Fh функциясы деректерді құрылғыдан енгізуді орындайды. Құрылғының дескрипторы ретінде 0-ді (стандартты енгізу -пернетақта) беру қажет. CX регистрінде енгізілетін байттардың саны көрсетіледі, DX регистріне аралық жадының адресі жүктеледі. Функция орындалғаннан кейін AX- та енгізілген байттардың нақты саны болады.

```
; деректер сегментінде
      BUF DB 20 DUP (?)
; кода сегментінде
      MOV AH, 9FH
      MOV BX, 0 ; пернетақта дескрипторы
      MOV CX, 20
      LEA DX, BUF
      INT 21h
      MOV CX, AX ; нақты енгізілген байттардың
; санын CX-та сақтау.
```

Құпиясөзді енгізу үшін жоғарыда айтылған функциялардың кез келгенін пайдалануға болады. Жасырын құпиясөзді енгізу үшін енгізілетін символды экранға шығармай пернетақтадан енгізу функциялары пайдаланылады.

Экранға шығару жүйелік функцияларына шолу.

Дисплейге шығарылатын барлық ақпарат ASCII кодасында көрсетілу керек. Символдарды дисплейге шығару үшін - үш функцияға (9,10,14) BIOS қызмет етеді.

Int 10h 9 функциясы - символды атрибутымен курсордың ағымдағы позициясына шығарады. Функцияны шақыру кезінде: BH - дисплей беті (текстік режим: 0-ағымдағы бет), BL - символдың атрибуты, CX - шығарылған символдардың қайталану саны, AL - шығарылатын символ.

```
MOV     AH,  9h
MOV     BH,  0
MOV     BL,  07h
MOV     CX,  1
MOV     AL,  'A'
INT     10h
```

Int 10h 10 функциясы - атрибутын өзгертпей символды ағымдағы курсор позициясына шығарады. Функция шақыру кезінде: BH - дисплей беті (0 - ағымдағы бет), CX - шығарылатын символдардың қайталану саны, AL - шығарылатын символ.

```
MOV     AH,  10
MOV     BH,  0
MOV     CX,  1
MOV     AL,  'A'
INT     10h
```

09h және 10 (0Ah) функцияларын қолдану кезінде, мынаны есте сақтау керек. Бұл функциялар символды (не символдар тобын) шығару кезінде курсорды жылжытпайды.

Int 10h 14 (0Eh) функциясы - символды экранға шығарады және курсорды келесі позицияға жылжытады. Функция шақыру кезінде:

```
AL - шығаратын символ
BH - беттердің нөмірі
MOV     AH,  14
MOV     AL,  'A'
MOV     BH,  0
INT     10h
```

DOS құралдарымен экранға шығару функциялары:

Int 21h 2 функциясы: символды курсор жылжытумен бірге дисплейге шығарады. Шақыруда:

```
DL - символ
MOV     AH,  2
MOV     DL,  'A'
INT     21h
```

Int 21h 9 функциясы: символдар жолын дисплейге шығарады.

Шақыруда:

DS:DX - \$ - белгісімен бітетін символдар жолының адресі.

Курсор жолдың соңына жылжиды.

;деректер сегментінде

```
STRING DB 'Текстік жол $'
```

;кода сегментінде

```
MOV AH, 9
```

```
LEA DX, STRING
```

```
INT 21h
```

Int 21h 40h функциясы: файлға жазу немесе деректерді құрылғыға шығару. Шақыруда:

BX - файлдың немесе құрылғының логикалық нөмірі. Дисплейдың логикалық нөмірі - 1;

CX - шығарылатын символдардың саны;

DS:DX - буфердың деректер алынатын адресі.

Қайта келу кезінде:

Егерде CF=0, онда AX факты бойынша жазылған байттардың саны. Егерде CF=1, онда AX - қайта келу коды (қате коды).

Кез келген функцияны шақыру кезінде сипатталған DS:DX белгісі келесі жолмен пайдаланылады. Деректер сегментінде немесе кода сегментінде адресі DX регистріне жүктелетін символдар жолы сипатталу керек. Ол мына командалар көмегімен орындалады.

```
LEA DX, <символдар жолының аты> ; немесе
```

```
MOV DX, offset <символдар жолының аты>
```

символдар жолы орналасқан сегментінің адресі DS регистрінде болады.

;деректер сегментінде

```
STRING DB 'текстік жол'
```

```
STRINGLEN EQU $- STRING
```

;кода сегментінде

```
MOV AH, 40h
```

```
MOV BX, 1 ; дисплейдің логикалық нөмірі
```

```
MOV CX, STRINGLEN
```

```
LEA DX, STRING
```

```
INT 21h
```

1 мысал. Горизонталь сызықты шығару.

```
MOV CX, 10
```

```
M1: MOV AH, 14
```

```
MOV AL, 196 ; ' — ' символдың ASCII кодасы
```

```
MOV BH, 0
```

```
INT 10h
```

```
LOOP M1
```

2 мысал. Атрибутты ауыстыру бойынша горизонталь сызықты шығару, сары символдар көк түсті фонда.

```

MOV     AH, 9
MOV     BH, 0           ; ағымдағы бет
MOV     CX, 15
MOV     AL, '='        ; шығарылатын символ
MOV     BL, 00011110B
INT     10h

```

Деректерді сипаттау.

Айнымалыларды деректер сегментінде деректерді *DB*, *DW*, *DD* директивалар көмегімен сипаттауға болады.

Мысалы:

```

A   DB  1, 2, 3, 4
B   DW  1122h, 3344h
C   DD  11223344h, 55667788h

```

Егер *байттармен* жұмыс істеу керек болса, онда оларға *byte ptr* оператор көмегімен қатынасуға болады.

```

MOV     AL, A           ; AL = 1
MOV     AL, byte ptr B ; AL = 22h
MOV     AL, byte ptr C ; AL = 44h

```

Екінші және үшінші мысалдарда *AL* регистріне сөздің және қоссөздің кіші байттары жазылады.

Үлкен байттарға қатынасу үшін адреске қажетті ығыстыруды қосамыз:

```

MOV     AL, byte ptr B +1; AL = 11h
MOV     AL, byte ptr C +3; AL = 11h

```

Егер *сөздермен* жұмыс істеу қажет болса, оларға қатынасу *word ptr* оператор көмегімен іске асады.

```

MOV     AX, word ptr A           ; AX = 0201h
MOV     AX, B                   ; AX = 1122h
MOV     AX, word ptr C          ; AX = 3344h

```

Үлкен сөздерге қатынасу үшін адреске 2 еселенген ығысу қосылады:

```

MOV     AX, word ptr C+2       ; AX = 1122h

```

Бағдарламаларда *EQU* немесе «=» директивалар көмегімен сипатталған тұрақтылар (константа) пайдаланылады.

;деректер сегментінде

```
K   EQU 255
```

```
A   DB  ?
```

;кода сегментінде

```
MOV  A, K
```

Бұл жағдайда екінші операнд тікелей операнд болып келеді, ал бірінші операнд адресін көрсетеді.

Массивтерді келесі түрде беруге болады:

```
MAS_A   DB  1, 2, 3, 4, 5, 6, 7, 8, 9
```

```
MAS_B   DB  31H, 32H, ..., 39H
```

MAS_C DB '1,2,3,4,5,6,7,8,9' ; Бұл массив жедел жадыда 17 байт орнын алады, неге десеңіз, ол өз бойында бөлгіш (үтір) ұстайды.

MAS_D DB 1B,10B,11B,100B,101B,110B,111B,1000B

;массив екілік кодасында берілген.

Екілік және ондық түрінде берілген массивтерді дисплейге шығару алдында ASCII - кодасына түрлендіру керек.

MOV AH, 2

MOV DL, mas[SI]

ADD DL, 30H; ASCII - кодасы алынды.

INT 21H

Символдар жолдары ASCII - кодасында беріледі. Сондықтан міндетті түрде тырнақшаға алынады.

StringA DB ' INT 21H 9 функцияны пайдалану '

DB ' кезінде текстік жол \$ доллар',13,10

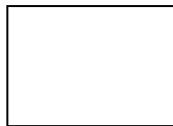
DB ' символмен бітуі қажет'

StringB DB ' INT 21H 40h функцияны қабылдау кезінде'

DB ' текстік жолының соңы',13,10

DB 'CX санауышпен анықталады'

Жақтауды дисплейге шығару кезінде оны деректер сегментінде текстік жол түрінде сипаттауға болады.

РАМКА DB 10,13,10 DUP (20h), '  ',10,13

DB 10 DUP (20h), ' ',10,13

DB 10 DUP (20h), '\$'

немесе

РАМКА DB 10,13,10 DUP (20h),218,8 DUP(196),191

DB 10,13,10 DUP (20h),179,8 DUP(20h),179

DB 10,13,10 DUP (20h),192,8 DUP(196),217,'\$'

Жақтауды сызуға арналған псевдографика ASCII кодалары:

┌ ─ ─ ─	218	194	191	┌ ─ ─ ─	201	203	187
---------	-----	-----	-----	---------	-----	-----	-----

└ ─ ─ ─	195	197	180	└ ─ ─ ─	204	206	185
---------	-----	-----	-----	---------	-----	-----	-----

└ ─ ─ ─	192	193	217	└ ─ ─ ─	200	202	188
---------	-----	-----	-----	---------	-----	-----	-----

	179	—	196		186	=	205
--	-----	---	-----	--	-----	---	-----

Символды дисплейге шығару кезінде DL регистріне (int 21h 2 функция) символдың ASCII-кодасы жүктеледі:

MOV DL, 41h ; 'A' символдың ASCII кодасы

немесе

MOV DL, 'A'

Мысал.

SSEG SEGMENT

DB 128 DUP (?)

SSEG ENDS

DSEG SEGMENT

STRING DB 'Текстік жол'

```

STRINGLEN    EQU    $- STRING
DSEG        ENDS
CSEG        SEGMENT
ASSUME      DS:DSEG, CS:CSEG, SS:SSEG
START:
    MOV AX, DSEG
    MOV DS, AX
    ;Символдық жолды экранға шығару
    MOV AH, 40h
    MOV BX, 1    ; дисплейдің логикалық нөмірі
    MOV CX, STRINGLEN
    LEA     DX, STRING
    INT     21h
    ;Бағдарламаның аяқталуы
    MOV AH, 4Ch
    INT     21h
CSEG        ENDS
END         START

```

Файлмен жұмыс жүйелік функцияларына шолу.

Бұл функцияның бәрі кенейтілген DOS нұсқасына жатады, онда файлда қатынасу дескриптор көмегімен жүзеге асады. ASCIIZ- жол форматымен берілген файлды ашу немесе құру нәтижесінде, дескриптор немесе файлдың логикалық нөмірі операциялық жүйе арқылы файлға меншіктелінеді.

ASCIIZ-жол - символдық жол, ол екілік нольмен бітеді. Файл дескрипторы AXрегистрына жазылады және оны жады ұяшығына сақтап қою керек.

Бірінші бес нөмір (0-4) стандартты құрылғыларға меншіктелінеді:

- 0 - стандарттық енгізу (CON).
- 1 - стандарттық шығару (CON).
- 2 - стандарттық қате (CON).
- 3 - стандарттық көмекші порт (AUX).
- 4 - стандарттық принтер (PRN).

0 дескрипторын біз пернетақтадан енгізуге қолдануымызға болады, ал 1 немесе 2 дескрипторларын экранға шығару үшін қолдануға болады.

Файлдармен жұмыс кезінде сәтті орындалған операция CF жалауына 0 орнатады. Егер операция орындалмаса, онда CF жалауына 1 орнатылады, ал AX регистріне қате кодасы қойылады.

Қателер кодалары:

Қате коды	түсініктемесі	Қате коды	түсініктемесі
01	функция нөмірінің қатесі	10	жабдықтар қатесі
02	файл табылмаған	11	формат қатесі
03	қатынау жолы табылмаған	12	қатынау кодасының

04	өте көп файл ашылған	13	қатесі
05	қатынауға рұқсат жоқ	15	деректердің қатесі
06	файл нөмірінің қатесі	16	дискжетек қатесі
07	жадыны басқару блогы	17	каталогты жою әрекеті
08	бұзылған	18	ол құрылғы емес
09	жады жетіспейді адрес қатесі		енді файлдар жоқ

3CH пен *5BH* функциясы файлды берілген спецификациямен құруға мүмкіндік береді (жол мен файлдың аты ASCIIZ-жол форматында). Бұл функциялардың айырмашылығы:

3CH функциясы бар файлды жояды да және осы атпен жаңа файл құрады, ал *5BH* функциясы - егер осындай аты бар файл болса, онда CF=1 болып бітеді.

Мысалы:

;Деректер сегментінде файлдың атын қою және дескрипторға

;ұяшықты резервтеу.

fname DB 'F1.TXT',0

handle DW ?

...

MOV AH, 3CH ; құруға сұрату

MOV CX, 00 ; әдеттегі файл

LEA DX, fname; файл атының адресі

INT 21h

JC ERR ; қате бойынша өту

MOV handle, AX ; дескрипторды сақтау.

Бұл мысалда CX регистріне файл атрибуты енгізіледі. Келесі атрибуттарды қолдануға болады:

01h - тек оқуға арналған файл

02h - жасырылған файл

04h - жүйелік файл

08h - том белгісі

10h - файл өзімен каталогты көрсетеді

20h - архивтеу атрибуты

3DH - функциясы - бар файлды ашуға мүмкіндігін береді.

Мысалы:

MOV AH, 3DH ; файлды ашу функциясы

MOV AH, 2 ; енгізу - шығару үшін

LEA DX, fname ;

INT 21h

MOV handle, AX

Файлды ашқан кезде AL - регистріне қатынау коды енгізіледі:

0 - файлды оқу үшін ғана ашу.

1 - файлға жазу үшін ғана ашу.

2 - енгізу мен шығару үшін файлды ашу.

42h функциясы файлдың кез келген орнына тура катынауды ұйымдастыру үшін қолданылады. Нұсқағышты файлдың басына (AL=0), ағымдағы орнына (AL=1) және файлдың соңына (AL=2) орнатуға болады. Онда нұсқағыштың ығыстыру мәні CX регистріне (үлкен жартысы) және DX регистріне (кіші жартысы) енгізіледі.

Мысалы:

```
MOV    BX, handle
MOV    AH, 42h ; нұсқағышты орнату
MOV    AL, 2   ; файлдың соңына
MOV    CX, 0
MOV    DX, 0
INT    21h
```

3Fh функциясы файлдан немесе құрылғыдан оқуға пайдаланылады. Файлдың соңынан оқуға әрекет жасағанда, AX=0.

Мысалы:

```
MOV    AH, 3Fh ; оқу функциясы
MOV    BX, handle; дескрипторды орнату
MOV    CX, 80  ; қанша оқу қажет
LEA    DX, buf ; және қайда
INT    21h
MOV    CX, AX  ; факты бойынша қанша оқылды.
```

40h функциясы файлға немесе құрылғыға жазу үшін пайдаланылады.

Мысалы:

```
MOV    AH, 40h ; жазу функциясы
MOV    BX, handle; дескриптор орнату
MOV    CX, 80  ; қанша жазу қажет
LEA    DX, buf ; және қайда
INT    21h
```

43h функциясы файлдың атрибутын алу үшін (AL=0), немесе орнату үшін (AL=1) пайдаланылады.

Мысалы:

```
MOV    AH, 43h ; атрибут жұмысының функциясы
MOV    AL, 1   ; атрибуттар орнату
MOV    CX, 1   ; "оқу үшін ғана"
MOV    DX, offset fname; файл атының адресі
INT    21h
```

56h функциясы файлдың атын өзгерту үшін пайдаланылады.

Мысалы:

;сегменттік регистрді ES деректер сегментіне баптау.

```
PUSH    DS
```

```
POP     ES
```

;файлдың атын өзгерту

```
MOV    AH, 56h ; ат өзгерту функциясы
```



```
MOV     DX, offset oldname    ; ескі атының адресі
MOV     DI, offset newname    ; жаңа атының адресі
INT     21h
```

;деректер сегментіне

```
oldname db 'F1.TXT',0
nevname db 'NEWF.TXT',0
```

57h функциясы файлдың құрылған уақытын алу үшін (AL=0) және күнін орнату үшін (AL=1) пайдаланылады.

Уақыт CX регистріне жазылады, сосын мына формуламен есептелінеді.

$$CX = \text{сағат} * 2048 + \text{минуттар} * 32 + \text{секунді} / 2$$

Ал күн болса DX регистріне жазылады және мына формуламен есептелінеді.

$$DX = (\text{жыл} - 1980) * 512 + \text{ай} * 32 + \text{күн}$$

Мысалы:

; файлдың құрылған күні мен уақытын өзгерту

```
MOV     AH, 57h    ; уақыт / күн функциясы
MOV     AL, 1      ; уақытты / күнді орнату
MOV     BX, HANDLE ; дескрипторды орнату
MOV     CX, 0      ; CX - тазалау
OR      CX, sec    ; секундты қосу
OR      CX, min    ; минутты қосу
OR      CX, hour   ; сағатты қосу
XOR     DX, DX     ; DX - тазалау
OR      DX, day    ; күнді қосу
OR      DX, mon    ; айды қосу
OR      DX, year   ; жылды қосу
INT     21h
```

;деректер сегментіне

```
sec dw 6/2        ; 6 секунд
min dw 15*32      ; 15 минут
hour dw 16*2048   ; 16 сағат
day dw 25         ; 25 күннің саны
mon dw 3*32       ; наурыз
year dw 15*512    ; 15 жыл (1995-1980)
```

3EH - функциясы файлды жабу үшін пайдаланылады. Бұл операция каталогты және FAT таблицасын түзету үшін қажет.

Мысалы:

```
MOV     AH, 3EH    ; жабу функциясы
MOV     BX, handle
INT     21h
```

41h функциясы файлды жою үшін пайдаланылады. (Тек ғана «оқу үшін ғана» деген файл атрибутынан басқа) .

```
MOV     AH, 41h   ; жою функциясы
LEA     DX, fname ; файл атының адресі
```

INT 21h ; DOS - ті шақыру

Файлмен жұмыс бірнеше кезеңнен тұруы мүмкін:

- файлды құру және ашу;
- нұсқағышты орнату;
- файлдың ұзындығын анықтау;
- керекті байтты, блокты оқу; байтты, блокты жазу; файл толтыру;
- файлды жабу;
- файлды жою.

Бағдарламаны өңдеу кезінде кезеңдердің бәрі керек емес және олар басқа ретпен жүруі мүмкін.

Керекті байтты оқыған кезде оның нөмірін немесе мәнін көрсетуге болады.

1-мысал. 10-шы байтты оқу, оқу үшін файлды ашу.

```
...
;нұсқағышты 10-байтқа орнату
MOV AH, 42h ; нұсқағышты орнату
MOV BX, handle
MOV AL, 0 ; файлдың басына
MOV CX, 0
MOV DX, 10 ; 10-байтқа
INT 21h
```

```
;керекті байтты оқу
MOV AH, 3Fh
MOV BX, handle
LEA DX, BUF
MOV CX, 1
INT 21h
```

```
...
;деректер сегментіне
```

```
BUF db ?
```

2-мысал. " A " деген символды табу және оның нөмірін анықтау.

```
;файлды ашу
```

```
...
;файлдың ұзындығын анықтау
MOV AH, 42h ; нұсқағышты орнату
MOV BX, handle
MOV AL, 2 ; файлдың соңына
MOV CX, 0 ; ығысудың үлкен жартысы
MOV DX, 0 ; ығысудың кіші жартысы
INT 21h
MOV FLEN,AX ; файл ұзындығын FLEN ұяшықта ; сақтау
; нұсқағышты файлдың басына орнату ...
;файлды BUFIN-ға оқу
```

```
...
```

;"A" символды іздеу
 CLD ; іздеу солдан оңға жүргізіледі
 MOV CX, FLEN; файлдың ұзындығы
 MOV AL, A ; ізделіп отырған символ
 LEA DI, BUFIN ; жол адресі
 REPNE SCASB ; іздеу
 JNZ M ; символ табылмаса, M-ге өту
 DEC DI ; символ табылды - адресі кішірейту
 ... ; нөмірді экранға шығару
 M: ...
 ;символ табылмады деген хабарды экранға шығару
 ;деректерді жабу.
 ...

5.3 Бақылау сұрақтары

- 1) INT 21h 1,7,8 функцияларының айырмашылығы қандай?
- 2) INT 21h 0Ah функциясын пайдаланған кезде аралық жадының бірінші және екінші байтына не жазылады?
- 3) Функциялық пернені басқанда енгізу қалай орындалады?
- 4) 0 (INT 16h) және 7 (INT 21h) функцияларының орындалуы қалай ажыратылады?
- 5) INT 21h 0Ah және 3Fh функцияларының айырмашылығы қандай?
- 6) INT 21h 0Ah функциясын пайдаланғаннан кейін, деректер аралық жадыда қай байттан басталады?
- 7) INT 21h 0Ah функциясын пайдаланғаннан кейін жолдың соңғы байтының

6 Зертханалық жұмыс №6. Windows арналған графикалық қосымшаны өңдеу

Жұмыстың мақсаты: Win32 ортасында бағдарламалаудың негізін оқып үйрену.

6.1 Келесі тапсырмаларды орындаңыздар

1. Негізгі терезе процедурасын оқып үйрену.
2. Хабарламаларды өңдеу циклын оқып үйрену.
3. Бағдарламада қолданылатын API-функцияларды оқып үйрену.
4. GetMessageA функциясының параметрлерін өзгерту.
5. CreateWindowExA функциясының параметрлерін өзгерту.
6. WNDPROC терезенің негізгі процедурасын өзгерту.

6.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Енді нақты бір мысалды өңдеуге кірісейік. Сурет 1-де қарапайым бағдарлама келтірілген. Оны мұқият талқылаңыз – ол ары қарайғы бағдарлама құрудың негізі болып табылады

Қарапайым бағдарламаны қарастырайық MASM32 кетесінде түрлі кітапханалар бар. Біздің мысалда екеуі қолданылады: user32.lib және Kernel32.lib. Олар includelib директивасы көмегімен қосылады.

Бірінші біз тұрақтылар және сыртқы кітапханалық процедураларды анықтаймыз. Барлық осы анықтауларды MASM32 кестесіндегі Include-файлдарда табамыз. Біз стандартты Include-файлдарды екі себеппен қолданбаймыз: біріншіден, осылай бағдарламалау технологиясын түсіну ыңғайлы екіншіден MASM-нан TASM-ға өту жеңіл болады.

Терезе процедурасының жұмысын түсіну өте қажет, өйткені дәл осы Windows-та бағдарламалау мәнін анықтайды. Бұл процедураның мақсаты-барлық келетін хабарларға дұрыс жауап беру. Барлық өңделмеген хабарлар Defwindow ProcA фунуциясы көмегімен жүйеге қайтарылуы қажет. Біз төрт хабарды қарастырамыз: WM_CREATE, WM_DESTROY, WM_LBUTTONDOWN, WM_RBUTTONDOWN.

WM_CREATE, WM_DESTROY хабарлары объектті бағдарламалауда конструктор және дескриптор қызметін атқарады: олар терезе функциясына терезені құрғанда және жойғанда келеді. Терезенің оң жақ бұрышындағы крешті басса, онда терезе функциясына WM_DESTROY хабары келеді. Одан кейін PostQuitMessage функциясы орындалады және қолданбаға WM_Quit хабары жіберіледі, ол күту циклынан шығуды және ExitProcess функциясын орындайды, ал бұл өз кезегінде жадыдан орындайды, ал бұл өз кезегінде жадыдан қолданбаның жойылуына алып келеді.

ERR таңбасына көңіл аударыңыз – ол таңбаға ауысу қате болған жағдайда орындалады, және мұнда сәйкесінше хабарлама беруге болады. (мысал қосымша А келтірілген)

6.3 Бақылау сұрақтары

1. Бағдарламада қандай API-функциялар қолданылады?
2. Бағдарламада қандай құрылымдар қолданылады?
3. Қосымшаның графикалық құрылымы.
4. Хабарламаларды өңдеу циклдары қандай API-функцияларды орындатады?

7 Зертханалық жұмыс №7. Консольдік қолданба

Жұмыстың мақсаты: консольді қосымшалар үшін API функцияны қолдану негіздерін үйрену.

7.1 Келесі тапсырмаларды орындаңыздар

1. Консоль терезесінің өлшемін өзгертіңіз.
2. Консоль терезесінің тақырыбын өзгертіңіз.
3. Курсор позициясын өзгертіңіз.
4. Тексттің түстік атрибуттарын өзгертіңіз.
5. NUMPAR және GETPAR процедураларының жұмыс алгоритмін түсіндіріңіз.

7.2 Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар

Консоль құру үшін AllocConsole функциясын қолданамыз. Бағдарлама жұмысын аяқтағаннан кейін консоль автоматты түрде немесе FreeConsole функциясының көмегімен босатылады.

Айта кететін жағдай, бір процестің бір ғана консолі болады, сондықтан бағдарлама орындаудың басында FreeConsole функциясын міндетті түрде орындап кету керек.

Консоль буферінен оқу үшін ReadConsole функциясы қолданылады.

Консольда курсор позициясын орналастыру үшін SetConsoleCursorPosition функциясын қолдану керек.

Шығарылатын символдар түсін SetConsoleTextAttribute функциясы көмегімен жасаймыз.

Консоль терезесінің тақырыбын анықтау үшін SetConsoleTitle функциясы қолданылады.

CharToOem функциясы DOS-кодировкадан Windows-кодировкаға ауыстыру үшін қолданылады.

Көптеген консольді функциялар аяқтауды дұрыс орындалса онда нөлдік мән қайтарады. Басқа жағдайда, яғни қате болса EAX регистріне 0 орналастырылады. (өз консоліңізді құруға мысал қосымша Б келтірілген)

7.3 Бақылау сұрақтары

1. GetStdHandle API функциясы қандай аргументтерді қолданады?
2. SetConsoleTextAttribute функциясымен қолданылатын әріптер және символдардың түстерін атаңыз?
3. CharToOem функциясының параметрлерін атаңыз?
4. Операциялық жүйемен қандай жағдай типтері резелфтелген?

А қосымшасы

Графикалық қосымшаны өңдеу

```
.386P
.MODEL FLAT, stdcall
; тұрақтылар
; хабарлама терезені жабқан кезде
келеді
WM_DESTROY equ 2
; хабарлама терезені құрған кезде
келеді
WM_CREATE equ 1
; хабарлама терезе аймағынан
тінтуірдің сол басқышын басқан
кезде келеді
WM_LBUTTONDOWN
    equ 201h
; хабарлама терезе аймағынан
тінтуірдің оң басқышын басқан
кезде келеді
WM_RBUTTONDOWN
    equ 204h
; терезе қасиеті
CS_VREDRAW equ 1h
CS_HREDRAW equ 2h
CS_GLOBALCLASS equ 4000h
WS_OVERLAPPEDWINDOW equ
    000CF000H
style equ CS_HREDRAW +
CS_VREDRAW +
CS_GLOBALCLASS
стандартты иконканың
идентификаторы
IDI_APPLICATION equ 32512
; курсор идентификаторы
IDC_CROSS equ 32515
; терезені көрсету режимі – қалыпты
SW_SHOWNORMAL equ 1
; сыртқы процедура прототиптері
EXTERN MessageBoxA: NEAR
EXTERN CreateWindowExA:
NEAR
EXTERN DefWindowProcA:
NEAR
EXTERN DispatchMessageA:
NEAR
EXTERN ExitProcess: NEAR
EXTERN GetMessageA: NEAR
EXTERN GetModuleHandleA:
NEAR
EXTERN LoadCursorA: NEAR
EXTERN LoadIconA: NEAR
EXTERN PostQuitMessage: NEAR
EXTERN RegisterClassA: NEAR
EXTERN ShowWindow: NEAR
EXTERN TranslateMessage: NEAR
EXTERN UpdateWindow: NEAR
; кітапхананы қосу үшін
жөндеуішке арналған директивалар
includelib
c:\win32\tasm32\lib\import32.lib
;-----
-----
; құрылымдар
; хабарлама құрылымы
MSGSTRUCT STRUC
MSHWNDD DD ?
; хабарлама идентификаторы
MSWPARAM DD ?
; хабарлама жайындағы
қосымша ақпарат
MSLPARAM DD ?
; хабарлама жайындағы қосымша
ақпарат
MSTIME DD ?
; хабарламаның жіберілген уақыты
MSPT DD ?
; хабарламаны жіберген кездегі
курсор орны
MSGSTRUCT ENDS
;-----
WNDCLASS STRUC
    CLSSTYLE DD ?
```

```

; терезе стилі
    CLWNDPROC      DD  ?
; терезенің процедурасының
көрсеткіші
    CLSCEXTRA      DD  ?
; берілген құрылымның қосымша
байттары жайындағы ақпарат
    CLWNDEXTRA     DD  ?
; терезеге арналған қосымша
байттары жайындағы ақпарат
    CLSHINSTANCE   DD  ?
; қосымша дискрипторы
    CLSHICON       DD  ?
; терезе иконкасының
идентификаторы
    CLSHCURSOR     DD  ?
; терезе курсорының
идентификаторы
    CLBKGROUND     DD  ?
; терезе қаламшасының (кист)
идентификаторы
    CLMENUMAME     DD  ?
; менюдің идентификатор-аты
    CLNAME         DD  ?
; терезе класының аты
спецификацияланады
WNDCLASS  ENDS
; деректер сегменті
_DATA    SEGMENT    DWORD
PUBLIC USE32 'DATA'
    NEWHWNDD      DD  0
    MSG MSGSTRUCT <?>
    WC WNDCLASS   <?>
    HINST         DD  0
; мұнда қосымша дескрипторы
сақталады
    TITLENAM     DB  '32-
биттік қосымшаның қарапайым
мысалы',0
    CLASSNAME    DB
'CLASS32',0
    CAP         DB
'хабарлама',0
    MES1        DB  'Cіз

```

```

тінтуірдің сол басқышын
бастыңыз',0
    MES2
DB'Программадан шығу. Сау бол!',0
_DATA  ENDS
; кодтар сегменті
_TEXT  SEGMENT    DWORD
PUBLIC USE32 'CODE'
START:
; қосымшаның дескрипторын алу
    PUSH    0
    CALL    GetModuleHandleA
    MOV     [HINST],EAX
REG_CLASS:
; терезе құрылымын толтыру
    MOV     [WC.CLSSTYLE], style
; хабарламаны өңдеу процедурасы
    MOV     [WC.CLWNDPROC],
OFFSETWNDPROC
    MOV     [WC.CLSCEXTRA], 0
    MOV     [WC.CLWNDEXTRA], 0
    MOV     EAX, [HINST]
    MOV     [WC.CLSHINSTANCE],EAX
;-----терезе иконкасы
    PUSH    IDI_APPLICATION
    PUSH    0
    CALL    LoadIconA
    MOV     [WC.CLSHICON],
EAX
;-----терезе курсоры
    PUSH    IDC_CROSS
    PUSH    0
    CALL    LoadCursorA
    MOV     [WC.CLSHCURSOR],
EAX
;-----
    MOV     [WC.CLBKGROUND],17;
терезе түсі
    MOV     DWORD PTR
[WC.CLMENUMAME], 0

```

```

MOV DWORD PTR [WC.CLNAME], OFFSET CLASSNAME
PUSH OFFSET WC
CALL RegisterClassA
; тіркелген кластағы терезе
күру
PUSH 0
PUSH [HINST]
PUSH 0
PUSH 0
PUSH 400 ; DY –
терезе биіктігі
PUSH 400 ; DX –
терезе ені
PUSH 100 ; Y –
жоғарғы сол жақ бұрыш
координаттары
PUSH 100 ; X -
жоғарғы сол жақ бұрыш
координаттары
PUSH
WS_OVERLAPPEDWINDOW
PUSH OFFSET
TITLENAMЕ ; терезе аты
PUSH OFFSET
CLASSNAME ; класс аты
PUSH 0
CALL CreateWindowExA
; қатеге тексеру
CMP EAX, 0
JZ _ERR
MOV [NEWHWND], EAX
PUSH
SW_SHOWNORMAL
PUSH [NEWHWND]
CALL ShowWindow
күрылған терезені көрсету
PUSH [NEWHWND]
CALL UpdateWindow ;
WM_PAINT хабарламасы, терезенің
көрінетін бөлігін қайта салу
командасы
; хабарламаны өңдеу циклы
MSG_LOOP:
PUSH 0
PUSH 0
PUSH 0
PUSH OFFSET MSG
CALL GetMessageA
CMP EAX, 0
JE END_LOOP
PUSH OFFSET MSG
CALL TranslateMessage
PUSH OFFSET MSG
CALL DispatchMessageA
JMP MSG_LOOP
END_LOOP:
; программадан шығу (процесті
жабу)
PUSH [MSG.MSGPARAM]
CALL ExitProcess
_ERR:
JMP END_LOOP
;-----
; терезе процедурасы
; стектегі параметрлердің
орналасуы
; [EBP+14H] LPARAM
; [EBP+10H] WPARAM
; [EBP+0CH] MES
; [EBP+8] HWND
WNDPROCPROC
PUSH EBP
MOV EBP, ESP
PUSH EBX
PUSH ESI
PUSH EDI
CMP DWORD
PTR [EBP+0CH],
WM_DESTROY
JE WMDESTROY
CMP DWORD
PTR [EBP+0CH],
WM_CREATE
JE WMCREATE
CMP DWORD

```


<pre> PTR [EBP+0CH], WM_LBUTTONDOWN; басқыш JE LBUTTON CMP DWORD PTR [EBP+0CH], WM_RBUTTONDOWN ;оң басқыш JE RBUTTON JMP DEFWNDPROC ; оң басқышты басу терезені жабады RBUTTON: JMP WMDESTROY ; сол басқышты басу LBUTTON: ; хабарламаны шығарамыз PUSH 0 MB_OK PUSH OFFSET CAP PUSH OFFSET MES1 PUSH DWORD PTR [EBP+08H] CALL MessageBoxA MOV EAX, 0 JMP FINISH WMCREATE: MOV EAX, 0 JMP FINISH DEFWNDPROC: PUSH DWORD PTR [EBP+14H] </pre>	<pre> PUSH DWORD PTR [EBP+10H] PUSH DWORD PTR [EBP+0CH] PUSH DWORD PTR [EBP+08H] CALL DefWindowProcA JMP FINISH WMDESTROY: PUSH 0 ; MB_OK PUSH OFFSET CAP PUSH OFFSET MES2 PUSH DWORD PTR [EBP+08H] ; терезе дескрипторы CALL MessageBoxA PUSH 0 CALL PostQuitMessage WM_QUIT хабарламасы MOV EAX, 0 FINISH: POP EDI POP ESI POP EBX POP EBP RET 16 WNDPROCENDP _TEXT ENDS END START </pre>
--	---

Б қосымшасы

Консольдік қолданба

```
; cons1.asm
.386P
.MODEL FLAT, stdcall
; тұрақтылар
STD_OUTPUT_HANDLE equ -
11
STD_INPUT_HANDLE equ -10
; түстер атрибуты
FOREGROUND_BLUE equ
; әріптің көк түсі
FOREGROUND_GREEN equ 2h
; әріптің жасыл түсі
FOREGROUND_RED equ 4h ;
әріптің қызыл түсі
FOREGROUND_INTENSITY equ
; жоғарғы интенсивтік
BACKGROUND_BLUE equ 10h
; фонның көк түсі
BACKGROUND_GREEN equ 20h
; фонның жасыл түсі
BACKGROUND_RED equ 40h
; фонның қызыл түсі
BACKGROUND_INTENSITY equ
80h
; жоғарғы интенсивтік
COL1 = 2h + 8h ; шығарылатын
текст түсі
COL2 = 1h + 2h + 8h ;
шығарылатын 2 текст түсі
; сыртқы процедура прототиптері
EXTERN GetStdHandle: NEAR
EXTERN WriteConsoleA: NEAR
EXTERN
SetConsoleCursorPosition:
NEAR
EXTERN SetConsoleTitleA: NEAR
EXTERN FreeConsole: NEAR
EXTERN AllocConsole: NEAR
EXTERN CharToOemA: NEAR

EXTERN
SetConsoleCursorPosition:
EAR
EXTERN SetConsoleTextAttribute:
NEAR
EXTERN ReadConsoleA: NEAR
EXTERN SetConsoleScreenBufferSize
: NEAR
EXTERN ExitProcess: NEAR
; кітапхананы қосу үшін жөндеуіш
директивалары
includelib c:\win32\work\import32.lib
;-----
-----
COOR STRUC
X WORD ?
Y WORD ?
COOR ENDS
; деректер сегменті
_DATA SEGMENT DWORD
PUBLIC USE32 'DATA'
HANDL DWORD ?
HANDL1 DWORD ?
STR1 DB 'Жолды енгізіңіз: ',
13, 10, 0
STR2 DB 'Консоль
жұмысының қарапайым мысалы', 0
BUF DB 200 dup (?)
LENS DD ? ;
енгізілген символдар саны
CRD COOR <?>
_DATA ENDS

_TEXT SEGMENT DWORD
PUBLIC USE32 'CODE'
START:
; жоды қайта кодтайық
PUSH OFFSET STR1
PUSH OFFSET STR1
CALL CharToOemA
```

```

; консоль құрастырушы
; бірінші бар консольды жою
керек
CALL FreeConsole
CALL AllocConsole
; HANDL1 енгізуді алу
PUSH
STD_INPUT_HANDLE
CALL GetStdHandle
MOV HANDL1, EAX
; HANDL шығаруды алу
PUSH
TD_OUTPUT_HANDLE
CALL
GetStdHandle
MOV HANDL, EAX
; Консоль терезесінің жаңа өлшемін
беру
MOV CRD.X, 100
MOV CRD.Y, 25
PUSH CRD
PUSH EAX
CALL
SetConsoleScreenBufferSize
; Консоль тақырыбын беру
PUSH
OFFSET STR2
CALL SetConsoleTitleA
; Курсор орнын беру
MOV CRD.X, 0
MOV CRD.Y, 10
PUSH CRD
PUSH HANDL
CALL SetConsoleCursorPosition
; Көрінетін тексттің түстік
атрибутын беру
PUSH COL1
PUSH HANDL
CALL SetConsoleTextAttribute
; жолды шығару
PUSH OFFSET STR1
CALL LENSTR ; EBX
жол ұзындығы
PUSH 0
PUSH OFFSET LENS
PUSH EBX
PUSH OFFSET STR1
PUSH HANDL
CALL WriteConsoleA
; енгізілетін жолды күту
PUSH 0
PUSH OFFSET LENS
PUSH 200
PUSH BUF
PUSH HANDL1
CALL ReadConsoleA
; алынған жолды шығару
; бірінші көрінетін тексттің түстік
атрибутын беру
PUSH COL2
PUSH HANDL
CALL SetConsoleTextAttribute
;-----
-
PUSH 0
PUSH OFFSET LENS
PUSH [LENS]
PUSH OFFSET BUF
PUSH HANDL
CALL WriteConsoleA
; кішкентай тоқтатым
MOV ECX, 01FFFFFFh
L1:
LOOP L1
; консольды жабу
CALL FreeConsole
CALL ExitProcess
; жол - [EBP+08h]
; EBX ұзындық
LENSTR PROC
; PUSH EBX
; MOV EBP, ESP
ENTER 0, 0
PUSH EAX
;-----
CLD
MOV EDI, DWORD PTR
[EBP+08h]

```

```

MOV EBX, EDI
MOV ECX, 100
; жолдың ұзындығын шектеу
XOR AL, AL
REPNE SCASB
; 0 символын табу
SUB EDI, EBX
; 0 қосқандағы жол ұзындығы
MOV EBX, EDI
DEC EBX;
POP EAX
; POP EBP
LEAVE
RET 4
LENSTR ENDP
_TEXT ENDS
END START

```

Командалық жолмен жұмыс жасау үшін GetCommandLineA API функциясы қолданылады, ол көрсеткішті командалық жолға қайтарады. Бұл функция консольді қосымшалар үшін де және GUI (Graphic Universal Interface) қосымшалары үшін де бірдей жұмыс істейді.

Мысал 2. Командалық жолдың параметрлерін шығаруға программа құру

```

; cons_2.asm
.386P
.MODEL FLAT, stdcall
; тұрақтылар
STD_OUTPUT_HANDLE
equ -11
; сыртқы процедура прототиптері
EXTERN GetStdHandle:
NEAR
EXTERN WriteConsoleA:
NEAR
EXTERN ExitProcess: NEAR
EXTERN
GetCommandLineA: NEAR

```

```

; кітапхананы қосу үшін
жөндеуіш директивалары
includelib
c:\win32\work\import32.lib
; деректер сегменті
_DATA SEGMENT DWORD
PUBLIC USE32 'DATA'
HANDL DWORD ?
NUM DWORD ?
BUF DB 100 dup (0)
LENS DWORD ? ;
енгізілген символдар саны
CNT DWORD ?
_DATA ENDS
; код сегменті
_TEXT SEGMENT DWORD
PUBLIC USE32 'CODE'
START:
; HANDLE шығаруды алу
PUSH
STD_OUTPUT_HANDLE
CALL
GetStdHandle
MOV HANDL, EAX
; параметрлер санын алу
CALL NUMPAR
MOV NUM, EAX
MOV CNT, 0
;-----
--
; командалық жолмен
параметрлерді шығару
LL1:
MOV EDI, CNT
CMP NUM, EDI
JE LL2
; параметр нөмірі
INC EDI
MOV CNT, EDI
; EDI нөмерімен параметр алу
LEA EBX, BUF
CALL GETPAR
; параметр ұзындығын алу

```

```

        PUSH     OFFSET
BUF
        CALL     LENSTR
; соңында - келесі жолға өту
MOV BYTE PTR [BUF+EBX], 13
MOV BYTE PTR [BUF+EBX+1], 10
MOV BYTE PTR [BUF+EBX+2], 0
        ADD EBX, 2
; жолды шығару
        PUSH     0
        PUSH     OFFSET LENS
        PUSH     EBX
        PUSH     OFFSET BUF
        PUSH     HANDL
        CALL     WriteConsoleA
        JMP LL1
LL2:
        PUSH     0
        CALL     ExitProcess
; жол - [EBP+08H]
; ұзындық EBX
LENSTR PROC
        PUSH     EBX
        MOV EBP, ESP
        PUSH     EAX
;-----
        CLD
        MOV EDI,     DWORD   PTR
[EBP+08H]
        MOV EBX, EDI
        MOV ECX, 100
; жолдың ұзындығын шектеу
        XOR AL, AL
        REPNE     SCASB
; 0 символын табу
        SUB EDI, EBX
; 0 қосқандағы жол ұзындығы
        MOV EBX, EDI
        DEC EBX
;-----
--
        POP EAX
        POP EBP
        RET 4

```

```

        LENSTR     ENDP
; параметрлер санын анықтау
(->EAX)
        NUMPAR     PROC
        LOCALS
        CALL
GetCommandLineA
        MOV ESI, EAX
; жол көрсеткіші
        XOR ECX, ECX
; санауыш
        MOV EDX, 1
; қызметі
        @@L1:
        CMP BYTE PTR [ESI], 0
        JE     @@L4
        CMP BYTE PTR [ESI], 32
        JE     @@L3
        ADD ECX, EDX
; параметр нөмері
        MOV EDX, 0
        JMP @@L2
        @@L3:
        OR  EDX, 1
        @@L2:
        INC ESI
        JMP @@L1
        @@L4:
        MOV EAX, ECX
        RET
        NUMPAR     ENDP
; параметрді алу
; EBX – параметр орналасқан
буферді көрсетеді
; буферге соңында 0 бар жол
орналастырылады
; EDI - параметр нөмері
        GETPAR     PROC
        LOCALS
        CALL
GetCommandLineA
        MOV ESI, EAX
; жол көрсеткіші
        XOR ECX, ECX

```

```

; санауыш
    MOV EDI, 1
; қызметі
@@L1:
    CMP BYTE PTR [ESI],
0
    JE    @@L4
    CMP BYTE PTR [ESI],
32
    JE    @@L3
    ADD ECX, EDX
; параметр нөмері
    MOV    EDX, 0
    JMP   @@L2
@@L3:
    OR    EDX, 1
@@L2:
    CMP ECX, EDI

```

```

JNE    @@L5
MOV AL, BYTE PTR [ESI]
MOV BYTE PTR [EBX], AL
INC    EBX
@@L5:
INC    ESI
JMP   @@L1
@@L4:
MOV BYTE PTR [EBX], 0
RET
GETPAR ENDP
_TEXT ENDS
END START
Программаның компиляциясы
үшін келесі командалық жолдар
қолданылады:
TASM32 /ml prog.asm
TLINK32 /ap pgog.obj

```

Әдебиеттер тізімі

Негізгі әдебиеттер

- 1 Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. –СПб.: Питер 2011.- 736 с.
- 2 Зубков С.В. ASSEMBLER для DOS, Windows, Unix.- М.:ДМК, 2009.- 640 с.
- 3 Пустоваров В.И. Ассемблер в программировании информационных и управляющих систем.-М.: «ЭНТРОП», 2010.- 307 с.
- 4 Сван Том. Освоение Turbo Assembler пер. с англ, Диалектика, 2009.- 544 с.
- 5 Юров В., Хорошенко С. ASSEMBLER: учебный курс. Питер Ком, 2010
- 6 Финогенов К.Г. Основы языка ассемблера. - М: Радио и связь, .2000.- 288 с.
- 7 Пильщиков В.Н. Программирование на языке ассемблера IBM PC-М.: «Диалог-МИФИ», 2011- 288 с.
- 8 Шайхин Б. М., Мусатаева Г. Т., Конуспаева А. Т., Байжанова Д. О. Ассемблер. 5B070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету мамандығының студенттері үшін дәрістер конспектісі. – Алматы: АЭЖБУ, 2011. – 81 б.
- 9 Мусатаева Г. Т., Байжанова Д. О. Ассемблер. Конспект лекций для студентов всех форм обучения специальности 050704 – Вычислительная техника и программное обеспечение. - Алматы: АИЭС, 2009 – 82 с.
- 10 Д. О. Байжанова. Ассемблер. 050704 – Есептеуіш техника және программалық жабдықтау мамандығының күндізгі бөлім студенттері үшін зертханалық жұмыстарды орындауға арналған әдістемелік нұсқау. – Алматы: АЭЖБИ, 2009. – 53 б.

Қосымша әдебиеттер

- 1 Абель П. Язык ассемблер и программирования.- Высш. шк. 2012. - 447 с.
- 2 Пирогов В.Ю. ASSEMBLER . Учебный курс.- М.: «Нолидж», 2011.- 848 с.
- 3 Пирогов В.Ю. Ассемблер для Windows.-М.: Издатель Молгачева С.В. 2012.- 552 с.
- 4 Рудаков П.И., Финогенов К. Г. Програмируем на языке ассемблера IBM PC-Изд. 3-е-Обнинск: Изд-во «Принтер», 2009
- 5 Скенлон Л. Программирование на языке ассемблера: Пер с англ. – М.: Радио и связь. 2009.

Мазмұны

1	Зертханалық жұмыс №1. Қарапайым есептерді орындау.....	3
2	Зертханалық жұмыс №2. Қарапайым деректер типі түсінігі. TURBO DEBUGGER (td) жөндегіші.....	7
3	Зертханалық жұмыс №3. Бағдарламаға басқаруды беру.....	11
4	Зертханалық жұмыс №4. Процессор командаларының әртүрлі топтарын қолдана отырып бағдарламаларды өңдеу.....	16
5	Зертханалық жұмыс №5. Қолданбалы бағдарламада жүйелік функцияларды қолдану.....	23
6	Зертханалық жұмыс №6. Windows арналған графикалық қосымшаны өңдеу.....	35
	Зертханалық жұмыс №7. Консольдік қолданба.....	37
	А қосымшасы.....	38
	Б қосымшасы.....	42
	Әдебиеттер тізімі.....	47

Гульназ Социалиевна Набиева
Сауле Ермаковна Смагулова

ЖҮЙЕЛІК БАҒДАРЛАМАЛЫҚ ҚАМТАМАСЫЗ ЕТУ

5B060200-Информатика мамандығының студенттеріне
зертханалық жұмыстарды орындауға арналған әдістемелік нұсқаулықтар

Редактор Қ.С.Телғожаева
Стандарттау бойынша маман Н.Қ. Молдабекова

Басуға _____ қол қойылды
Таралымы 30 дана.
Көлемі 3,0 есептік-баспа табақ

Пішімі 60x84 1/16
Баспаханалық қағаз №1
Тапсырыс ____ Бағасы 1500 тенге

«Алматы энергетика және байланыс университеті»
коммерциялық емес акционерлік қоғамының
көшірмелі-көбейткіш бюросы
050013, Алматы, Байтұрсынұлы көшесі, 126