Аннотация

Язык ассемблера помогает раскрыть все секреты аппаратного и программного обеспечения. С его помощью можно получить представление о том, как аппаратная часть взаимодействует с операционной системой и как прикладные программы обращаются к операционной системе. Большинство программистов работают с языками высокого уровня, где отдельное утверждение преобразовывается во множество процессорных команд. Ассемблер – язык машинного уровня; каждая команда непосредственно интерпретируется в машинный код, что даёт основание считать его языком низкого уровня. Наиболее часто язык ассемблера используется для написания дополнений к операционной системе или для написания программ прямого доступа к аппаратуре. Он необходим также при оптимизации критических блоков в прикладных программах с целью повышения их быстродействия.

Кіріспе

Ассемблер тілі машинаға бағытталған тіл болып саналады, яғни процессор тілі. Ассемблер тілі - программаның кодын оңтайландыратын, драйверлер, трансляторлар, қорғау процедураларын жазатын, кейбір сыртқы құрылғыларды бағдарламалаудың ынғайлы және икемді құралы болып табылатын құрылымды бағдарламалаудың жан-жақты тілі.

Зертханалық жұмыстың негізгі мақсаты студенттерді ассемблер тілінің негізгі түсініктерімен таныстыру, оқыту, процессорының негізінде компьютердің архитектурасын түсіндіру. Құрылғылармен тікелей жұмыс істеу кезінде тіл мүмкіндігін барынша қолдана білу, белгілі алгоритмдерді тарату, сондай-ақ жүйелік және қолданбалы есептерді шеше алу.

Әрбір зертханалық жұмысты бітіргенде оны есеп беру нәтижесімен тұжырымдайды.

Есеп бергенде мына жұмыстар болу керек:

- жұмыстың мақсаты және тапсырмалары;
- жасаған жұмыстың қорытындысы (программаның тексті, программаның жұмыс қорытындысы);
- бақылау сұрақтарына жауап беру;
- әдебиеттер тізімі.

Жасалған жұмыс және есеп беру мазмұны оқытушыда қорғалады.

Программларды орындаушы компьютер туралы сөз қозғағанда, біз ең алдымен оның «жүрегі» – *процессор*, яғни – нұсқаулар, деп жиі аталатын командаларды орындайтын және олардың жұмыс нәтижелерін арнайы регистрлерде сақтайтын арнайы микросұлбаны ескереміз. Мысалы, процессордың келесі нұсқаулар тізбегін алсақ:

mov eax, 2
add eax, 3

eax регистрінің ішінде «5» саны пайда болады. mov eax, 2 – деп жазылған бірінші инструкция «2» санын eax регистрінің ішіне жібереді. add eax, 3 – деп жазылған екінші инструкция (біріншіден кейін орындлатын) eax регистрінің ішіндегі барына «3» санын қосады.

Процессор тек сандарды ғана түсінеді. Сондықтан ассемблер тілінде жазылған программа сандарға аударылуы тиіс, ал бұл сандар компьютердің белгілі бір нұсқауларды орындауына келтіреді. Бірақ келесі жағдайды ескеру қажет. Әрбір ассемблер жолына белгілі бір нұсқау сәйкесті болады.

Ассемблер тілінде жазылған программаның атқаратын міндеті келесі: адам жазған мәтінді оқып, оны сандар тізбегіне аудару (әрине процессор түсінетіндей етіп).

Ассемблер тіліндегі программа мәтіні нұсқаулардан басқа қызмет атқаратын информацияға да иеленеді. Бұл соңғы ақпарат *ассемблер-программа* не істеуі тиіс екенін, яғни оған қандай талап қойылып отырғанын түсіндіруге көмектеседі.

1 Зертханалық жұмыс №1. Процессор командаларының әртүрлі топтарын қолдана отырып программаларды өңдеу

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

Келесі тапсырмаларды орындаңыздар:

1) Берілген сөздегі немесе байтағы бірлердің санын санайтын программа жазыңыз:

```
;деректер        сегменті
A       DB      01101011B немесе A DW 0010111110101001B
```

2) Программаны нөлдерді санайтындай етіп өзгертіңіз.

3) Таңбалы сандардан тұратын A массивін беріп. Осы массивтен теріс сандарды санайтын программа жазыңыз. Сіздің кез-келген вариантыңыз қабылданады. Осыны TEST командасын қолданып қайталаңыз.

4) Программаны массив элементтерінің оң сандарын есептейтіндей етіп өзгертіңіз.

5) Жадыдағы сөз ретіндегі екі жинақталмаған BCD сандарын жинақталған AL регистріндегі BCD санына ауыстыру программасын жазыңыз.

6) Символдарды біл жолдан екінші жолға ауыстыру программасын жазыңыз:

```
;деректер сегментінде
str_source      DB      ‘Ауыстырылатын жол’,’$’     ;таратқыш жол
str_dest        DB      20 dup(?)                   ;қабылдағыш жол
```

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

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

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

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

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.

CMP 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 тратқыщ 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 регистрына 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 (коньюкция) ЖӘНЕ командасы.

Команданың форматы: AND қабылдағыш, таратқыш

OR - байт немесе сөз өлшемі бар қабылдағыш және таратқыш операндтардың разрядтарын логикалық қосу (дизъюкция) 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, өзгеретін 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 операндттың 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 1орнатады, сәйкесінше таратқыштың биттері 1тең, XOR – командасы қабылдағыш биттерін терістейді, сәйкесінше таратқыш биттері де терістеледі.

Жылжыту командалары.

Ығыстыру командаларының әсер ету принципі бойынша екі типке бjлуге болады:

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

- циклдық ығысу командалары (қарапаым 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 регитріне 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) нолдер 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) нолдер 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 - цикілдің солға ығысу командасы, ол операнд мазмунын ығысу санауышының мәнімен анықталатын биттер санына солға ығыстырады.Солға ығыстырылған биттер сол операндқа оң жағынан жазылады.

ROL - операнд, ығысулар-санауышы

ROR - цикілдің оңға ығысу командасы, ол операнд мазмұнын ығысу санауышының мәнімен анықталатын биттер санына оңға ығыстырады.

ROR - операнд, ығысулар-санауышы

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

RCL - операнд, ығысулар-санауышы

RCR - тасымал жалауы арқылы цикілдық оңға ығысу командасы ол операнд мазмунын ығысу санауышының 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 сегметінде (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 командаларына 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

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?

2 Зертханалық жұмыс №2. Қарапайым есептерді орындау

Жұмыстың мақсаты: программаны 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 алгоритімі анықталып қойылған. Енді бұл алгоритмді ассемблер тілінің негізгі конструкцияларын қолданып жазу керек. Жұмыстың дұрыстығын тексеру үшін керекті тестілік деректер болуы керек.

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 сенментіне мынаны жазыңыз:

```
mov     ah, 2h      ; символды экранға шығару функциясы
mov     dl, ‘A’     ;ASCII
int     21h         ; 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 нөмері
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 қолдпныңыз.

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

Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар.

Компьютерге кез келген редактор көмегімен программаның 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 программаны енгізіңіз:

```
Stacksg   segment
          dw        32 dup(?)
Stacksg   ends
Datasgt   segment
string    db        ‘Фамилия мен аты‘ ,13,10
          db        ‘Жасы: …..’,’$’
datasg    ends
codesg    segment
          assume    cs:codesg, ds:datasg, ss:stacksg
start:
          mov       ax, datasg
          mov       ds, ax
          mov       dx, offset string
          mov       ah,9h
          int       21h
          mov       al,0
          mov       ah,4ch
          int       21h
codesg    Ends
          end       Start
```

Түзетуден шыққанан соң сіздің программаңыздың 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:

- панельде TASM.EXE файлын белгілеңіз CTRL+ENTER басыңыз;
- панельде 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 программаны редактор көмегімен жөндеп, қайтадан трансляция жасау керек.

Орындалатын программаны 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.

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

Ассемблер тілінде программаны құрастыру кезінде келесі кезеңдерді ұсынуға болады:

1) блок-сұлбаны құру;

2) NAME.ASM бастапқы программасын құру. NAME келісілген кез келген файлдың аты.

3) NAME.OBJ объектік программаны құру.

4) NAME.EXE орындалатын программаны құру.

5) EXE программаны орындау.

6) программаның нәтижелерін тексеру.

Егер есептің талаптарына программаның нәтижелері сәйкес келмесе, онда қателерді тауып, програмаы жөндеу керек.

Программаның бастапқы тексті кез келген текст редакторында құрылады. Тексттік редакторы ASCII кодаларында берілген программалардың бастапқы текстерін енгізу және түзету қамтамасыз ететін программа болып табылады.

Транслятор (компилятор) NAME.OBJ (машиналық тілде программаның тексті, бірақ ішкі программалардың 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 программа.

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 програманы 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 программа 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 программа 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.

Бақылау сұрақтары:

1) Ассемблер тіліндегі программаның құрылымы.
2) Программаның орындалу кезеңдері.
3) 9h және 2h шығару функциялары.
4) Бастапқы файл қалай жасалады?
5) Бастапқы файлдың кеңейтуі.
6) Транслятор аты.
7) Транслятор не істейді?
8) Трансляция нәтижесінде қандай шығару файлдары алынады?
9) Трансляторды іске қосудың командалық жолының мысалы.
10) Құрастырғыш аты.
11) Құрастыру кезінде қандай файлдар енгізу файлдары болады?
12) Құрастыру кезінде қандай файлдар шығару файлдары болады?
13) Орындалатын файлдың қандай кеңейтуі бар?
14) Құрастырғышты іске қосудың командалық жолының мысалы.
15) Листинг файлын қалай қарауға болады?

3 Зертханалық жұмыс №3. Программаға басқаруды беру

Жұмыстың мақсаты: сызықты емес алгоритмдерді бағдарламалаудың тәсілін үйрену.

Келесі тапсырмаларды орындаңыздар:

1) В массивін экранға шығару алдында мынандай текстік 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 программаға енгізіңіз.

3) Программа денесінде бір цикл ұйымдастыру арқылы берілген программаны өзгертіңіз.

4) Берілген программаны сөздер массивін тасымалдау программасына өзгертіңіз.

5) Байт массивін сөз массивіне түрлендіру үшін программаны жазыңыз.

```
;деректер        сегменті
A               db          1,2,3,4,5,6
B               dw          3 dup(?)
; код           сегменті
(сіздің программаңыз)
```

6) А және В екі байт массивтерін қосу программасын жазыңыз, қосындысын С массивінде сақтаңыз.

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

7) Екі А және В байттар массиві берілген, ал үшінші массивке әр элементті салыстыра отырып максимальды элементтерді теріп жазу керек. Программа жазыңыз, сіздің кез – келген вариантыңыз қаралады.

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

Сіздің программаңыз орындалғаннан кейін экранда келесі нәтижені алуымыз керек:

Массив С 7 8 9 6

Сәттілік тілейміз!

8) Алдыңғы программаңызды С массивіне минималды элементті іздеп орналастыратындай етіп өзгертіңіз, яғни нәтиже келесі түрде болуы керек:

Массив С 4 5 2 3

Алға!

9) Енді берілген байттар массивіндегі элементтердің максималды элементін іздейтін программа 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.

10) Алдыңғы программаңызды берілген байттар массивінен минималды элемент іздейтіндей етіп жазыңыз.

Әрине, нәтиже басқа болады:

Минималды элемент А массивінің – 4 элементі.

Егер экрандағы нәтиже өзгрмесе онда сіз қате жібердіңіз.

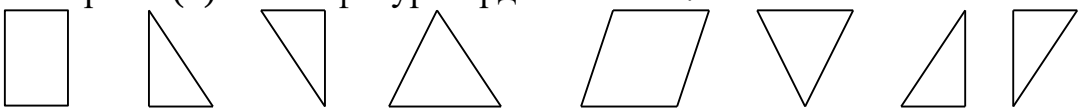
11) А массиві берілсін.

```
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
```

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



13) Екілік кодта экранға шығару процедурасын жазыңыз.

14) Оналтылық кодта экранға шығару процедурасын жазыңыз.

Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар.

А байттар массивін В байттар массиві орнына тасымалдау программасы 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
```

Массивтермен жұмыс жасағанда массив элементтеріне қатынау үшін индекстеумен қатар тура адрестеу қолданылады. Индекстеумен қатар тура адрестеу 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 массивы 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 әр түрлі варианттардың 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 программа орындалуын тоқтатады, қайда және қалай бару керектігінің нүктесі пайда болады. Кей кездері мұндай жағдайлар кандай да бір шартпен байланысты болуы мүмкін, ал кей кездері жай ғана басқа орынға «кетіп қалуң болуы 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 циклды ұйымдастыру командасы болып 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 принциптарына байланысты топтарға бөлінеді:

- басқару тізгінін шартсыз ауыстыру командалары;
- басқару тізгінін шартқа байланысты ауыстыру командалары.

Басқару тізгінін шартсыз ауыстыру командалары:

1) шартсыз ауысу командасы - JMP таңба;

2) процедураны шақыру және процедурадан қайту командалары - CALL процедура_аты және RET;

3) программалық үзулерді шақыру және - INT үзу_нөмірі, IRET - программалық үзулерден қайту, INTO – асатолу болған кездегі үзу.

Басқару тізгінін шартқа байланысты ауыстыру командалары:

1) Салыстыру командасының нәтижесіне байланысты ауысу командасы: CMP операнд1, операнд2

Жұмыс алгоритімі:

- 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

2) 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

3) CX регистрінің құрамына байланысты ауысу командасы

JCXZ Ауысу_таңбасы

Бақылау сұрақтары:

1) Қандай шартты ауысу командалары таңбалы сандармен жұмыс кезінд қолданылатын?

2) Қандай шартсыз ауысу командалары кейбір команда топтарын айналып өтеді?

3) Қандай командалар цикл ұйымдастыру кезінде қолданылады?

4) Қандай командалар прорграмманы шақыру үзуді өңдеу және одан қайтуды орындайды?

5) Қандай командалар процедураны (ішкіпрограмманы) шақыру және одан қайтуды орындайды?

4 Зертханалық жұмыс №4. Қарапайым деректер типі түсінігі. Turbo debugger (TD) жөндегіші

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

Келесі тапсырмаларды орындаңыздар:

1) Программаның бастапқы текстін вариантқа сәйкес теріңіз:

```
Datasg      segment
Mess        db          ‘ Деректердің директивалары $’
Pa          db
Pb          dw
Pc          dd
Mas         db          тапсырма вариантарындағы 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 вариантта 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     Al,     Pa                  ; al = ?
mov     Bx,     Pb                  ; bx = ?
mov     Bl,     byte ptr Pb         ; bl = ?
mov     Dx,     word ptr Pc         ; dx = ?
mov     Cx,     word    ptr         ; cx = ?
                Pc+2
mov     Dl,     byte ptr Pc         ; dl = ?
mov     Dh,     byte ptr Pc+1       ; dh = ?
```

Осы командалардың орындалу нәтижесін жөндегіштен қараңыз.

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. Программада 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 текстi деңгейінде программаларды 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 программаны қадам бойынша орындау режимінде жиі жұмыс жасайсыз. Бұл режимде сіз команда бойынша программаны орындай аласыз, сонымен бірге әр командaның 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 программасын көрсететін терезе. Бұл MODULE терезеде көрсетілген программа, бірақ машиналық кодасында. Қадам жөндеуді тұра 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 дамп терезесі, ол деректер жадысының аймағының мазмұнын көрсетеді.

Аты MODULE жөндегіш терезесінде сіз программаның бастапқы текстін орындалу курсорімен (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 программаның орындалатын модулін көрсетіп жасайды: Td орындалатын_модульдің_аты.

Тапсырма вариантары:

1)	Pa	db	73H	9)	Pa	db	5BH
	Pb	dw	0AE21H		Pb	dw	0BA21H
	Pc	dd	38EC76A4H		Pc	dd	0FA4A32BCH
	Mas	db	10 dup(1),2,3		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
2)	Pa	db	67H	10)	Pa	db	4AH
	Pb	dw	4AEFH		Pb	dw	0DEFCH
	Pc	dd	12DC4567H		Pc	dd	81ADFF06H
	Mas	db	5,6,7,8		Mas	db	5 dup(1),2,3,3 dup(4)
	Pole	db	6 dup(0)		Pole	db	6 dup(« «)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
3)	Pa	db	4DH	11)	Pa	db	7FH
	Pb	dw	0ED56H		Pb	dw	0ACDEH
	Pc	dd	32AF8DD7H		Pc	dd	10B0A488H
	Mas	db	4,3,5, 4 dup(0)		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
4)	Pa	db	5DH	12)	Pa	db	0BCH
	Pb	dw	0A1A3H		Pb	dw	903FH
	Pc	dd	3 dup(4),5,6		Pc	dd	6CAA3E41H
	Mas	db	4,3,5, 4 dup(0)		Mas	db	1,2,3, 4 dup(4)
	Pole	db	5 dup(?)		Pole	db	5 dup(?)

	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
5)	Pa	db	62h	13)	Pa	db	0FBH
	Pb	dw	7ED1H		Pb	dw	54ADH
	Pc	dd	0EE45DA31H		Pc	dd	0E04365FAH
	Mas	db	1,2, 6 dup(3),0		Mas	db	3 dup(0), 4 dup(1),2,3
	Pole	db	5 dup(0)		Pole	db	5 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
6)	Pa	db	0FFH	14)	Pa	db	11H
	Pb	dw	4ADEH		Pb	dw	4D2DH
	Pc	dd	0C23891F5H		Pc	dd	98ADF156H
	Mas	db	4 dup(0),1,2,3		Mas	db	5 dup(0),1,2,3
	Pole	db	3 dup(‘ ‘)		Pole	db	3 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
7)	Pa	db	0AEH	15)	Pa	db	10H
	Pb	dw	63BCH		Pb	dw	1A2DH
	Pc	dd	63BCDEF3H		Pc	dd	55AEF2C8H
	Mas	db	9,8,3 dup(0)		Mas	db	1,2,3,4,5,6
	Pole	db	5 dup(« «)		Pole	db	5 dup(0)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc
8)	Pa	db	67H	16)	Pa	db	89H
	Pb	dw	4AEFH		Pb	dw	91ADH
	Pc	dd	12DC4567H		Pc	dd	0AFD43E55H
	Mas	db	5,6,7,8		Mas	db	5,6, 3 dup(9)
	Pole	db	6 dup(0)		Pole	db	4 dup(?)
	Adr	dw	Pc		Adr	dw	Pc
	Adr_full	dd	Pc		Adr_full	dd	Pc

Бақылау сұрақтары:

1) Байттар жадыда калай орналасады?

2) Сөздер және екілік сөздер жадыда калай орналасады?

3) Сөздер және екілік сөздер қандай байтан адрестеледі?

4) Айнымалының толық адресі жадыда калай орналасады?

5) Айнымалы мен регистр өлшемі әртүрлі болса, онда ассемблер командасыда қандай операторлар қолданылады?

5 Зертханалық жұмыс №5. Қолданбалы программада жүйелік функцияларды қолдану

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

Келесі тапсырмаларды орындаңыздар:

Программа жазу керек:

1) INT 21h 0Ah (INT 21h 3Fh) функциясы көмегімен пернетақтадан жолды енгізу.

2) Құпиясөзді 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 программаны 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 текстік жолын экранға шығару.

12) Файлға текстік 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.

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.
        CMP     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 кажет. Аралық жадының бірінші байтта өлшемі болу керек. Функция орындалғаннан кейін аралык жадынының 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 дискрипторы
    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
```

Программаларда 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   ' тексдік жолының соңы',13,10
          DB   'CX санауышпен анықталады'
```

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

```
RAMKA     DB   10,13,10 DUP (20h                    ' ,10,13
          DB   10 DUP (20h),                        ' ,10,13
          DB   10 DUP (20h),          '             $'
```

немесе

```
RAMKA     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'
```

3-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
    ; Программаның 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 АХрегистрына 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 АХ 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 қатесі
04	өте көп файл ашылған	13	
05	қатынауға рұқсат жоқ	15	деректердің қатесі
06	файл нөмірінің қатесі	16	дискжетек қатесі
07	жадыны басқару блогы бұзылған	17	каталогты жою әрекеті
08	жады жетіспейді	18	ол құрылғы емес
09	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 оку үшін ғана ашу.

1 - файлға жазу үшін ғана ашу.

2 - енгізу мен шығару үшін файлды ашу.

42h функциясы файлдың кез келген орнына тура қатынауды 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 = сағат * 2048 + минуттар * 32 + секунді / 2

Ал күн болса DX регистріне жазылады және мына формуламен есептелінеді.

DX = (жыл - 1980) * 512 + ай * 32 + 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:

1) файлды құру және ашу;

2) нұсқағышты орнату;

3) файлдың ұзындығын анықтау;

4) керекті байтты, блокты оқу; байтты, блокты жазу; файл толтыру;

5) файлды жабу;

6) файлды жою.

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

Керекті байтты оқыған кезде оның нөмерін немесе мәнін көрсетуге болады.

4-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   ?
```

5-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 - адресті кішірейту
```

. . . ; нөмірді экранға шығару

M: . . .

; символ табылмады деген хабарды экранға шығару

; деректерді жабу.

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 мәні неге тең?

8) Символды экранға шығару функциясы.

9) Символдар жолын экранға шығару функциясы.

10) INT 21h 9 және 40h функцияларын орындау кезінде символдар жолының соңын қалай анықтайды?

11) Символды шығару кезінде қандай функциялар курсорды жылжытпайды?

12) Қандай функция дисплейге шығу кезінде атрибут символын ауыстырады?

13) Файл дескрипторы деген не?

14) ASCIIZ - жолы деген не?

15) Стандартты енгізу және стандартты шығару дескрипторлары.

16) Кез-келген файлдық операцияның сәтті аяқталғаны қалай анықталады?

17) Файл атрибуттары.

18) Файлды ашқандағы қатынау кодалары.

19) Файл ұзындығын қалай анықтауға болады?

20) Файлды жабу функциясы не істейді?

21) 41h функциясы арқылы қандай файлды жоюға болмайды?

6 Зертханалық жұмыс №6. Windows арналған графикалық қосымшаны өңдеу

Жұмыстың мақсаты: Win32 ортасында бағдарламалаудың негізін оқып үйрену.

Келесі тапсырмаларды орындаңыздар:

1) Негізгі терезе процедурусын оқып үйрену.

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

Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар.

Қарапайым программаны 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 MS DOS-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.

```
.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 маустың сол басқышын басқан кезде келеді
WM_LBUTTONDOWNequ    201h
; хабарлама терезе аймағынан маустың оң басқышын басқан кезде келеді
WM_RBUTTONDOWN       equ    204h
; терезе қаситеті
CS_VREDRAW   equ    1h
```

```
CS_HREDRAW    equ    2h
CS_GLOBALCLASS    equ    4000h
WS_OVERLAPPEDWINDOW        equ    000CF0000H
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
    MSHWND    DD    ?          ; хабарлама идентификаторы
    MSWPARAM      DD    ?      ; хабарлама жайындағы қосымша ақпарат
    MSLPARAMDD    ?            ; хабарлама жайындағы қосымша ақпарат
    MSTIME    DD    ?          ; хабарламаның жіберілген уақыты
    MSPT      DD    ?      ; хабарламаны жіберген кездегі курсор орны
    MSGSTRUCT     ENDS
    ;--------------------------
    WNDCLASS STRUC
    CLSSTYLE      DD    ?      ; терезе стильі
    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
    CLMENUNAME    DD   ?      ;менюдің идентификатор-аты
    CLNAME        DD   ?      ;терезе класының аты
спецификацияланады
    WNDCLASS ENDS
    ;деректер сенменті
    _DATA SEGMENT DWORD PUBLIC USE32 'DATA'
    NEWHWND DD   0
    MSG   MSGSTRUCT    <?>
    WC    WNDCLASS     <?>
    HINST       DD   0          ;мұнда қосымша дескрипторы сақталады
    TITLENAME   DB   '32-биттік қосымшаның қарапайым мысалы',0
    CLASSNAME   DB   'CLASS32',0
    CAP         DB   'хабарлама',0
    MES1        DB   'Сіз маустың 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], OFFSET      WNDPROC
    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.CLMENUNAME], 0
MOV   DWORD PTR [WC.CLNAME], OFFSET CLASSNAME
PUSH  OFFSET    WC
CALL  RegisterClassA
; тіркелген кластағы терезе құру
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    TITLENAME     ; 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            ; құрылған 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 (процессті жабу)
PUSH  [MSG.MSWPARAM]
CALL  ExitProcess
_ERR:
JMP     END_LOOP
;-------------------------------------------------------------------------
; терезе процедурасы
; стектегі параметрлердің орналасуы
; [EBP+14H]  LPARAM
; [EBP+10H]  WPARAM
; [EBP+0CH]  MES
; [EBP+8]      HWND
WNDPROC   PROC
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 PTR [EBP+0CH], WM_LBUTTONDOWN    ;сол 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]
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
WNDPROC   ENDP
_TEXTENDS
END    START
```

Бақылау сұрақтары:

1) Программада қандай API-функциялар қолданылады?

2) Программада 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.

Келесі тапсырмаларды орындаңыздар:

1) Консоль терезесінің өлшемін өзгертіңіз.

2) Консоль терезесінің тақырыбын өзгертіңіз.

3) Курсор позициясын өзгертіңіз.

4) Текстің түстік атрибуттарын өзгертіңіз.

5) NUMPAR және GETPAR процедураларының жұмыс алгоритімін түсіндіріңіз.

Зертханалық жұмысты орындау үшін керекті әдістемелік нұсқаулар.

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

Айта кететін жағдай, бір процестің бір ғана консолы болады, сондықтан программа 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 MS DOS-кодировкаға ауыстыру үшін қолданылады.

Көптеген консольді функциялар аяқтауды дұрыс орындалса онда нөлдік мән қайтарады. Басқа жағдайда, яғнм қате болса EAX регистріне 0 орналастырылады.

Мысал 1. Өз консоліңізді құру

```
; cons1.asm
.386P
.MODEL FLAT, stdcall
; тұрақтылар
STD_OUTPUT_HANDLE       equ   -11
STD_INPUT_HANDLE  equ   -10
; атрибуты цветов
FOREGROUND_BLUE  equ   1h              ; әріптің көк  түсі
FOREGROUND_GREENequ   2h               ; әріптің жасыл  түсі
FOREGROUND_RED          equ   4h       ; әріптің қызыл түсі
FOREGROUND_INTENSITY    equ   8h       ; 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 2текст 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: NEAR
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    STD_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 текстің 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    OFFSET  BUF
        PUSH    HANDL1
        CALL    ReadConsoleA
; алынған жолды шығару
; бірінші көрінетін текстің 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, 01FFFFFFFh
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 нөмері
  INC   EDI
  MOV CNT, EDI
; EDI нөмерңмен 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
```

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?

Әдебиеттер тізімі

1 Юров В., Хорошенко С. Ассемблер. - СПб.: «Питер», 2009. – 470 с.

2 Зубков С.В. Assembler для DOS, Windows и Unix. – М.: ДМК, 2014. – 378 с.

3 Пирогов В.Ю. ASSEMBLER. Учебный курс. – М.: «Нолидж», 2012.- 589 с.

4 Финогенов К.Г. Самоучитель по системным функциям MS-DOS. - М.: МП «МАЛИП», 2008. – 432 с.

5 Пирогов В.Ю. ASSEMBLER для Windows. – М.: Издатель Молгачева С.В., 2013. – 521 с.

6 Том Сван. Освоение Turbo Assembler. - М.: «Диалектика», 2006. – 312 с.

7 Рудаков П.И., Финогенов К.Г. Язык ассемблера: уроки программирования. – М.: «ДИАЛОГ-МИФИ», 2005. – 542 с.

8 Скэнлон Л. ПЭВМ IBM PC и XT. Программирование на языке Ассемблера. - М.: «Радио и Связь», 2007. – 368 с.

9 Абель П. Язык Ассемблера для IBM PC и программирования. - М.: «Высшая школа», 2014. – 687 с.

10 Брэдли Л. Программирование на языке Ассемблера для персональных ЭВМ IBM.-М.: «Радио и Связь», 2005. – 354 с.

11 Нортон П., Соухэ Д. Язык Ассемблера для IBM PC: Пер.с англ., - М.: «Компьютер», 2013. – 741 с.

12 Нортон П. ПК фирмы IBM и OS MS-DOS. -М.: «Радио и Связь», 2006. – 365 с.

13 Пузанков Д.В. Микропроцессорные системы. – СПб.: «Политехника», 2007, – 935 с.

14 Система программирования на макроассемблере MS-DOS (в 4-х частях). – Киев: НПО «Горсистемотехника», 2006. – 874 с.

Мазмұны

2015 ж. баспа жоспары, реті 211

Қоржымбаев Тұрсын Толебаевич

WINDOWS ОРТАСЫНДА АССЕМБЛЕР ТІЛІНДЕ ПРОГРАММАЛАУ

5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету мамандығының студенттері үшін зертханалық жұмыстарды орындау бойынша әдістемелік нұсқаулар

Редакторы Ж.А.Байбураева
Стандарттау бойынша маман Н. Қ. Молдабекова

Басуға___ ___ _____ қол қойылды
Таралымы 30 дана
Көлемі 3,2 есептік-баспа табақ

Пішімі 60х84 1/16
Баспаханалық қағаз №1
Тапсырыс Бағасы 1600 теңге

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