“Basics of microcontroller”

Lecture 3.

8085 microprocessor and its components

Lecturer: Onon Otgonbaatar, MD.

Ассемблер хэлний командууд

Ассемблер хэлний командуудын ерөнхий формат:

[name]	operation	[operand(s)]
нэр	үйлдэл	гишүүд

Нэг мөрөнд хамгийн ихдээ 132 тэмдэгт байна. Жишээ нь:

```
COUNT   DB      1       ; нэр, үйлдэл, нэг гишүүн
MOVE    AX      0       ;үйлдэл, 2 гишүүн
```

Нэр:

A→Z; a→z; 0→9; ?; .; @; _; $ гэх мэт үсэг, тоо, тэмдэгтүүд оролцож болно. Хамгийн ихдээ 31 ширхэг тэмдэгт байх ба регистрүүдтэй ижил нэртэй байж болохгүй. PAGE25, $E10 гэх мэт байж болно.

Үйлдэл:

Ассемблерийн ямар үйлдэл хийхийг заана. Өгөгдлийн сегмент дотор үйлдэл нь ажлын талбар, эсвэл тогтмол утга аль нь болохыг тодорхойлно. Код сегмент дотор (mov-ачаалах, add-нэмэх) хийх үйлдлийг нь заана.

Гишүүн:

Гишүүн нь өгөгдлийн анхны утгыг заах буюу эсвэл үйлдлийн хаана хийгдэхийг заана. Жишээ нь:

COUNTR DB 0; COUNTR гэсэн нэртэй байтыг 0 гэсэн анхны утгатайгаар тодорхойлно.

```
RET                     ;буцаах үйлдэл, гишүүнгүй
INC   CX                ;нэгээр нэмэгдүүлэх үйлдэл, 1 гишүүнтэй
ADD   AX,   12          ;ах дээр 12-г нэмэх ба 2 гишүүнтэй
```

Хийсвэр үйлдлүүд (Pseudo operations) буюу директивүүд

Ийм үйлдлүүд нь машины код үүсгэдэггүй ба программын хөрвүүлэлтийн үед зөвхөн ажиллана. Өргөн хэрэглэгддэг хийсвэр үйлдлүүдийг дурдвал:

1. PAGE

Программын эхэнд 1 хуудсанд байх мөрний тоо, нэг мөрт байх тэмдэгтийн тоог зааж өгнө. 1 хуудсанд 10-255 мөр, 1 мөрт 60-132 тэмдэгт байж болно. Үүнийг тодорхойлж өгөөгүй бол PAGE 66,80 гэж автоматаар авна.

PAGE 60,132; нэг хуудсанд 60мөр, нэг мөрт 132 тэмдэгт байлгах үйлдэл.

2. TITLE

Программд гарчиг өгөх ба үүнийг хуудас бүрийн эхэнд хэвлэнэ. Программын тайлбартайгаа нийлээд 60 тэмдэгтийн урттай байна.

Жишээ нь: TITLE ASMSORT-assembler program to sort customer names гэх мэт

3. SUBTTL

Дэд программд нэр өгнө.

4. SEGMENT

Бүх ассемблер программууд нь хамгийн багадаа нэг сегмент агуулна, Код сегментийг биелэгдэх кодууд ашиглана. Зарим програмууд стек сегментийг стек санах ойтой ажиллахдаа, өгөгдөл тодорхойлохдоо дата сегментийг ашигладаг. Формат нь:

Name SEGMENT [options]

…

…

…

Name ENDS –сегментийн төгсгөл

Option нь 3 янз байна.

- **alignment type**
  Сегментийн эхлэх завсарыг заана. Сегментийн эхлэх хаяг 16-д хуваагдах тоо байх ба үүнийг Paragraph boundary гэнэ. Эндээс эхлэнэ гэж үзвэл PARA гэж тавина. Segment гэж тодорхойлоогүй бол para гэж ойлгоно.
- **combine type**
  Өөр сегменттэй холбогдох (link) үед STACK, COMMON, PUBLIC, AT, MEMORY гэх мэт төрлүүдийн аль нэгийг зааж өгнө. Жишээ нь:
  Name SEGMENT PARA STACK гэж stack сегментийг тодорхойлж өгч болно. Өөр программуудтай холбогдоогүй программуудад үүнийг тодорхойлохгүй.
- **class type**
  Клас төрлийг нь apostroph (‘ ‘) дотор бичих ба хамтрах (link) сегментүүдтэй грүп болгоход хэрэглэнэ. Жишээ нь:
  Name SEGMENT PARA STACK ‘Stack’

5. PROC

Код сегмент нь программын биелэгдэх кодоос гадна нэг буюу хэд хэдэн процедуруудыг агуулна. Зөвхөн нэг процедуртай сегментийн жишээ:

```
Segname     SEGMENT     PARA
Procname    PROC    FAR
            …
            …
            RET
Procname    ENDP
Segname     ENDS
```

Far гэсэн операнд нь процедурыг программын эхлэл хэсэг гэдгийг DOS-н программ ачаалагчид зааж өгнө. Endp нь заасан процедуртай ижил нэртэй байх ба процедурын төгсгөлийг заана.

6. ASSUME

Микропроцессор нь SS регистрийг stack-ийг хаяглахад, DS регистрийг data segment-ийг хаяглахад, CS регистрийг code segment-ийг хаяглахад хэрэглэнэ. Программист нь assembler –т сегмент бүрийн үүргийг зааж өгөх ёстой. Ийм зорилготой хийсвэр үйлдэл бол assume бөгөөд код сегмент дотор доорх байдлаар бичигддэг.

ASSUME SS:stackname, DS:datasegname, CS:codesegname

ES:nothing гэж бичих нь ES-ийг бичээгүйтэй ижил ба уг регистрийг ашиглаагүй гэсэн үг.

7. END

End үйлдэл нь бүхэл программыг дуусгана.

END [procname] ; операнд нь хоосон байж болно.
ENDS [segname] ; сегментийг дуусгана.
ENDP [procname]; процедурыг дуусгана.

Программын толгой (Program initialization)

Биелэгдэх боломжтой программын 2 төрөл байна. EXE, COM
DOS нь Ассемблерийн EXE программыг эхлүүлэхдээ дараах шаардлагуудыг тавина.

1. Аль сегмент нь аль сегмент регистртэй нэгдэхийг нь заана.
2. Программын биелэлт эхлэх үед DS регистр дэх хаягийг stack-д хадгална. Санах ойд программ биелэгдэхийн яг өмнө PSP (program segment prefix) гэж нэрлэгддэг 256 байт (100h) хэмжээтэй санах ойн хэсэг үүсдэг. DOS-ын ачаалагч программ нь DS регистрийг PSP-ийн эхлэлийг тавихдаа ашигдана. Программ нь stack уруу “push” хийн энэ хаягийг хадгална. Сүүлд нь DOS уруу буцахдаа “ret”-ийг ашиглана.
3. Stack-д 0 гэсэн хаягийг хадгална.
   Систем нь stack дахь дараагийн утгыг (өөрөөр хэлбэл offset хаяг) 0 байхыг шаарддаг. Энэ зорилгоор sub команд ах-ийг цэвэрлэн, push командаар stack-д хадгална.
4. DS-т өгөгдлийн сегментийн хаягийг хадгална.
   DOS-ын ачаалах программ нь stack-ийн хаягийг SS-д, code segment-ийн хаягийг CS-д хадгалсан. Ачаалагч программ DS-ийг өөр зорилгоор ашигласан учир DS-ийг 2 ширхэг MOV команд ашиглан анхны утгыг нь тодорхойлно.

Дээрхийг бичвэл:

```
STACKSG     SEGMENT PARA STACK      Stack       ‘Stack’
DATASG      SEGMENT PARA ‘Data’
CODESG    SEGMENT   PARA ‘Code’
      BEGIN       PROC FAR
1.          ASSUME    CS:CODESG, DS:DATASG, SS:STACKSG
2.          PUSH DS
3.          SUB       AX, AX
            PUSH AX
4.          MOV       AX, DATASG
            MOV       DS, AX
            …
            …
5.          RET
      BEGIN       ENDP
      CODESG      ENDS
            END       BEGIN
```

Push ds гэж stack-д хийсэн хаягийг ret командаар хаягаа авч DOS уруу буцаж, программаас гарна. Өөр нэг өргөн хэрэглэгддэг гарах үйлдэл (exit) бол INT20H команд юм.

Дээрх бичиглэлийг хялбар байдлаар бичвэл доорх маягаар тодорхойлж болно. Цаашдаа жишээ программуудыг дараах хялбар бичиглэлээр бичиж явна.

```
.Stack  100H
.Data
.Code
        BEGIN       PROC FAR
                MOV         AX, @Data
                MOV         DS, AX
                …
                …
                RET
        BEGIN       ENDP
        END         BEGIN
```

Жишээ программ

```
                PAGE 60,132
TITLE EXASM1 (EXE) Example register operations
;-------------------------------------------------------------------------
.Model small
.Stack
        ORG 100h
;-------------------------------------------------------------------------
.Code
BEGIN       PROC            FAR
             MOV            AX, 0123H    ; ах-д 0123h утгыг ачаална.
             ADD            AX, 0025H    ; ах-н утган дээр 25h-г нэмнэ
             MOV            BX, AX       ; ах-н утгыг вх-д хийнэ
             ADD            BX, AX       ; вх дээр ах-н утгыг нэмнэ
             MOV            CX, BX       ; вх- н утгыг сх-д хийнэ
             SUB            CX, AX       ; сх-ах ялгаварыг ах-д олгоно
             SUB            AX, AX       ; ах-д 0 утга хийнэ.
             NOP
             RET                         ; DOS уруу буцна
BEGIN       ENDP                        ; процедурын төгсгөл
             END            BEGIN        ; программын төгсгөл
```

Энэ программ нь stack segment, code segment-ийг агуулна. Код сегмент нь программын кодуудыг агуулна. Программд өгөгдөл тодорхойлоогүй тул DS регистрт өгөгдлийн сегментийн хаягийг олгож өгөх шаардлагагүй.

Өгөгдлийг тодорхойлох

Өгөгдлийн сегментийн зорилго нь тогтмолууд, ажлын талбхрууд, оролт гаралтын ажлын хэсгүүдийг тодорхойлох юм. Өгөгдөл тодорхойлох хийсвэр үйлдлүүд (директив) байдаг.

Өгөгдөл тодорхойлох формат:

```
[name]      Dn      expression
[name]      Dn      repeat-count DUP(expression)
```

Dn-ийн оронд доорх командууд байна.

DB	define byte	- Байтыг тодорхойлох
DW	define word	- 1 үгийг буюу 2 байт утгыг тодорхойлох
DD	define doubleword	- 4 байт утгыг тодорхойлох
DQ	define quadword	- 8 байт утгыг тодорхойлох
DT	define tenbytes	- 10 байт утгыг тодорхойлох

Мөн тогтмолыг тодорхойлох өөр нэг арга бол команд дотор шууд утга өгч болно. Жишээ нь: mov al, 20h гэх мэт

Жишээ:

FLD1		DB	25; тогтмол утга зарлах
FLD2		DB	? ; анхны утгыг нь олгоогүй тогтмол зарлах
FLD3		DB	11, 12, 13, 14, 15, …; олон тогтмол утгыг зарлах
			FLD3 нь 11 гэсэн утгаа авна, mov al, fld3+3 гэвэл al-д 14 гэсэн утгыг олгоно.
DW	10	DUP(?);	анхны утгыг нь олгоогүй 10 байт
DB	5	DUP(14);	0E гэсэн 16-тын утгыг агуулсан 5 байт
DB	3	DUP(4 DUP(8));	16 ширхэг 8-уудыг үүсгэнэ.

Хавсралт 2 – Тэмдэгтүүд болон тоон утгуудыг тодорхойлох нь

<u>Тоон тогтмолууд</u>

Энэ нь арифметик утгууд ба санах ойг хаяглахад хэрэглэгдэнэ.

Decimal (10-т):
0…9 хүртэлх тоонуудаас бүрдэх ба ард нь D үсэг бичих буюу эсвэл бичихгүй байж болно.

Hexadecimal (16-т):
0…9, A, B, C, D, E, F хүртэлх тоонуудаас бүрдэх ба ард нь H үсэг бичнэ.

Binary (2-т):
0 ба 1 гэсэн 2 утгыг л авах ба ард нь B үсэг бичнэ.

Octal (8-т):
0…7 хүртэлх тоонуудаас бүрдэх ба ард нь Q үсэг бичнэ.

Жишээ нь: 12D = 0CH = 1100B = 14Q
125D = 7DH = 01111101B = 175Q

DB 00H…0FFH байхаас 00H…7FH, 80H…0FFH буюу +127…-128
DW 0000H…0FFFFH байхаас 0000H…7FFFH, 8000H…0FFFFH буюу +32767…-32768
DD 00000000H…0FFFFFFFFH байхаас 0H…7FFFFFFFH, 80000000H…0FFFFFFFFH буюу +2147483647…-2147483648

<u>Шууд утгаа авдаг командууд:</u>

- Move and compare (зөөх ба харьцуулах): mov, cmp
- Arithmetic: adc, add, sbb, sub
- Logical: and, or, test, xor
- Shift (шилжүүлэх): rcl, rcr, rol, ror, shl, sar, shr

EQU директив

Энэ нь өгөгдөл тодорхойлохгүй боловч тоон утгыг тодорхойлдог хийсвэр үйлдэл юм.
Data segment –д
TIMES EQU 10
гэсэн команд байвал TIMES гэдэг үг командад дурдагдах бүрт программ 10 гэсэн тоон утгыг авна.
Жишээ 1:
FIELD DB TIMES DUP(?) гэсэн команд нь
FIELD DB 10 DUP(?) гэсэн командтай ижил юм.

Жишээ 2:
```
COUNTR   EQU  5
         …
         …
         MOV CX, COUNTR;      mov cx, 5 гэсэнтэй ижил
```

EQU нь олон командууд COUNTR гэж тодорхойлогдсон утгыг хэрэглэх үед сайн талтай. Энэ утга нь өөрчлөгдөхгүй, өөрчлөх тохиолдолд EQU дотор л өөрчлөлт хийнэ. Мөн доорх байдлаар тэмдэгт нэрүүдийг ижилсгэж болно.
TP EQU TOTALPAY; TOTALPAY data segment-д тодорхойлогдсон байна.
MPY EQU MUL; MPY-ийг хэрэглэх үед MUL гэсэн үржих командыг ашиглана.

COM өргөтгөлтэй файл

Өргөн хэрэглэгддэг COM файлын нэг жишээ бол COMMAND.COM юм. Turbo Assembler программыг хэрэглэх үед COM өргөтгөлтэй файлд хөрвүүлэхдээ TLINK /T EXAMPLE.OBJ гэж бичвэл EXAMPLE.EXE файлыг бус EXAMPLE.COM файлыг шууд үүсгэнэ. Мөн EXE2BIN.COM нэртэй DOS-ын программ нь EXE файлыг COM файл уруу хөрвүүлдэг. EXE ба COM файлуудын ялгаа:

1. Программын хэмжээ
   COM файл нь хамгийн ихдээ 64к байт буюу 1 сегментийн хэмжээгээр хязгаарлагддаг бол EXE программ нь ямар ч хэмжээтэй байж болно. COM файл нь өөрийнхөө EXE файлаас үргэлж бага хэмжээтэй байдаг. Нэг шалтгаан нь 512 байт (header block) толгой блок COM файлд байдаггүй, харин EXE файлтай ажиллахад энэ блок хэрэглэгддэг.
2. Stack segment
   EXE программ нь stack segment-ийг тодорхойлдог ба COM программ нь stack-ийг автоматаар үүсгэдэг. Иймээс COM файлд хөрвүүлэх зорилготой assembler программыг бичих үедээ stack-ийг орхиж болно.
3. Data segment
   EXE программ data segment-ийг голдуу тодорхойлдог ба DS регистрийн анхны утгыг нь олгоно. COM программ code segment дотроо өгөгдлөө тодорхойлох ёстой.
4. Анхны утга олгох (Initialization)
   EXE программ DS регистрт анхны төлөвийг нь тотооно. COM программд stack ба data segment-үүд байхгүй учираас дээрх үйлдлийг хийдэггүй. COM программ эхлэх үед бүх сегмент регистрүүд PSP (program segment prefix)-ийн хаягийг агуулна. PSP нь 256 байт хэмжээтэй санах ойн блок ба COM болон EXE программыг санах ойд эхлүүлэхийн өмнө DOS оруулдаг. Учир нь PSP-ийн эхлэлээс 100h offset утгаас

эхлэн хаяглалт явагдана. Code segment-ийн бичлэгийн дараа ORG 100H гэсэн хийсвэр үйлдэл бичигддэг.

5. Хөрвүүлэлт (Conversion)
   EXE ба COM программуудыг үүсгэхдээ 2-уулангийнх нь ассемблер программд хөрвүүлэлт хийн OBJ файл, эндээс EXE файлыг үүсгэнэ. EXE файлыг хэрэглэх хүсэлтэй бол эндээс шууд ашиглах ба COM файл мэтээр ажиллуулах гэвэл No stack segment гэсэн мэдээлэл өгнө. Харин
   TLINK /T CALC.OBJ гэхэд COM программ нь обьект программаасаа үүснэ.
   Мөн өөрөөр, EXE2BIN CALC, CALC.COM гэж exe-гээс com файлд хөрвүүлсний дараа Com файл үүснэ. Com файлд өөр нэр өгч болно.
   Com файл үүссэний дараа OBJ, EXE гэсэн файлуудыг устгаж болно.

COM программын жишээ:

```
                PAGE 60,132
TITLE EXCOM1    COM program to move and add
.Code
        ASSUME  CS:CODESG, DS:CODESG, SS:CODESG,
                        ES:CODESG
        ORG 100H
BEGIN:  JMP     MAIN
;-------------------------------------------------------------------
FLDA    DW      250
FLDB    DW      125
FLDC DW ?
;-------------------------------------------------------------------
MAIN    PROC NEAR
        MOV     AX, FLDA
        ADD     AX, FLDB
        MOV     FLDC, AX
        RET
MAIN    ENDP
CODESG  ENDS
        END     BEGIN
```

Дээрх программд:

- Stack ба data segment байхгүй
- ASSUME үйлдэл нь код сегментийн эхлэлээс offset-үүдийг эхлүүлэхийг заана. CS регистр нь энэ эхлэлийн хаягийг агуулах ба мөн PSP-ийн хаяг гэсэн үг.
- ORG 100h үйлдэл нь биелэлтийн эхний хаягийн offset хаягийг тавина. Программ ачаалагч нь энэ хаягийг командын заагчид хадгална.

Энэ программыг хөрвүүлэхдээ:

```
        tasm excom1.asm
```

гэж assembler файлаас object файлаа үүсгэнэ.

```
        tlink excom1.obj
```

гэж excom1.exe буюу биелэгдэх боломжтой файлыг үүсгэнэ. Хэрэв

```
        tlink/t excom1.obj
```

бичвэл excom1.com файл нь үүснэ.

Exe2bin excom1, excom1.com гэж com файлаа үүсгэж мөн болно.

Del excom1.obj, excom1.exe гэж зөвхөн com файл үүсгэх хэрэгтэй байсан бол бусад хэрэггүй файлуудаа устгаж болно.

Энэ программын хувьд exe ба com программууд нь тус тус 788 байт, 20 байт хэмжээтэй. Exe, Com программуудыг debug-аар ажиллуулан өгөгдөл ба командуудыг харахдаа:

td excom1.exe эсвэл td excom1.com

Com файл нь стек, data сегментүүдийг агуулдаггүй ба DS регистрийн ч анхных нь төлөвийг тогтоож өгдөггүй. DOS com программын стекийг тодорхойлдог.

Textbook

1. Peter Abel, “IBM PC Assembler language and programming”, USA, 1987
2. Jim Mischel, "Macro magic with Turbo Assember ”, USA, 1993
3. William C.Runnion, “Structured programming in Assembly Language for the IBM PC”, Boston, 1988
4. Thomas A.Wadlow, “Memory resident programming on the IBM PC”, USA, 1987
5. Robert S.Lai, “Writing MS-DOS device drivers”, USA, 1987
6. E. Majigsuren, “IBM assembly language”, MGL, 2003
7. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay, “The 8051 microcontroller and Embedded Systems Using Assembly and C”,USA, 2007