“Basics of microcontroller”

***Lecture* 9.**

About the logical operator

Lecturer: Onon Otgonbaatar, MD.

Матрицан мэдээллийг хөрвүүлэх

XLAT команд нь матриц хэлбэрийн мэдээллийг хөрвүүлэхэд ашиглагдана. Жишээ нь: ASCII болон EBCDIC форматуудыг хооронд нь хөрвүүлэхэд, том үсгийг жижиг үрүү гэх мэт байж болно.
Дараах байдлаар ASCII болон EBCDIC форматуудыг хөрвүүлье. 0-9 хүртэлх тооны ASCII код нь 30-39 бол EBCDIC код нь F0-F9 юм. Хоосон тэмдэгтийн EBCDIC код нь 40h байдаг.
Энэ команд нь BX, AL регистрүүдийг шууд ашиглана. BX регистр хандах гэж буй таблицын эхний элементийн хаягийг агуулах ёстой. AL регистрт хөрвүүлэх гэж буй (ASCNO гэж нэр өгье) байтыг олгоно.

```
XLTBL  DB    47 DUP (40H)                          ; EBCDIC хоосон зайнууд
         DB    0F0H, 0F1H, 0F2H, 0F3H, …, 0F9H      ; EBCDIC 0-9
         DB    199 DUP (40H)                        ; EBCDIC хоосон зайнууд
       …..
       LEA          BX, XLTBL
       MOV          AL, ASCNO
       XLAT
```

XLAT команд нь AL регистрийн утгыг шилжлэг хаяг болгон ашиглаж, BX регистрийн утга дээр нэмнэ. Жишээ нь:
ASCNO-д 00 гэсэн утга байвал, хүснэгтийн хаяг нь XLTBL+0 болох ба XLAT команд нь AL регистр дахь 00 утгыг 40h утгаар солино. Хэрэв ASCNO нь 32h утгатай байвал, хүснэгтийн хаяг нь XLTBL+50 болно. Энэ хаяган дээр F2 (EBCDIC 2) гэсэн утга агуулагдаж байгаа ба үүнийг XLAT команд AL регистрт хийнэ.

- ASCII тоонуудыг EBCDIC форматад хөрвүүлэх жишээ программ

Цэгийн ASCII код 2E байдаг ба EBCDIC форматад 4B байна.
Хасах тэмдгийн ASCII код нь 2D байх ба EBCDIC формат нь 60h байна.
ASCNO нь –31.5 гэсэн утга агуулж байгаа ба ардаа хоосон тэмдэгттэй тул 6 удаа давталт хийе. Үүнийг 16-тын кодоор нь бичвэл 2D33312E3520 болоа ба прораммын төгсгөлд ASCNO нь 60F3F14BF540 утгатай байна.

```
Page 60, 132
Title XLATE  Translate ASCII to EBCDIC
.model small
.stack 100h
.data
       ASCNO      DB     '-31.5 '
       EBCNO      DB     6 DUP (' ')
       XLTAB      DB     45 DUP (40H)
                  DB     60H, 4BH
                  DB     5CH
                  DB     0F0H, 0F1H, 0F2H, 0F3H, 0F4H, 0F5H,
                         0F6H, 0F7H, 0F8H, 0F9H
                  DB     199    DUP (40H)
;-------------------------------------------------------------------------
.code
MAIN  proc far
       Mov    ax, @data
       Mov    ds, ax
       Lea    SI, ASCNO          ; ASCNO-ийн хаяг
       Lea    DI, EBCNO          ; EBCNO-ийн хаяг
```

```
        Mov     CX, 06              ; урт
        Lea     BX, XLTAB           ; хүснэгтийн хаяг
A20:
        Mov     AL, [SI]            ; ASCII тэмдэгтийг авна.
        Xlat                        ; хөрвүүлнэ.
        Mov     [DI], AL            ; EBCNO-д хадгална.
        Inc     SI
        Inc     DI
        Loop    A20                 ; 6 удаа давтана.
        Mov     ax, 4c00h
        Int 21h
        Ret
MAIN endp
        End MAIN
```

- 16-т ба ASCII өгөгдлүүдийг дэлгэцэнд хэвлэх программ
  Энэ программ нь бараг бүх ASCII тэмдэгтүүдийг тэдний 16-тын утгатай нь харуулна. Жишээ нь:
  53h кодтой ASCII тэмдэгт нь S үсэг бөгөөд програмд 53 S гэж харуулна. Бүхэл дэлгэц 16*16 матриц хэлбэрээр доорх байдлаар харагдана.

0	01	02	03	04	….	0E	0F
10	11	12	13	14	….	…	…
20	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
F0	F1	F2	F3	F4	…….	FE	FF

```
Page 60, 132
Title ASCII HEX  Conversion of ASCII characters to Hex
.model small
.stack 100h
.data
        DISPROW     DB      16 DUP(‘        ’), 13
        HEXCTR      DB      00
        XLATAB      DB      30H, 31H, 32H, 33H, 34H, 35H, 36H, 37H,
                            38H, 39H
                    DB      41H, 42H, 43H, 44H, 45H, 46H
;-------------------------------------------------------------------------------
.code
MAIN  proc near
        Mov     ax, @data
        Mov     ds, ax
        Mov     es, ax
        Call    CLRSCR              ; дэлгэц цэвэрлэнэ.
        Lea     SI, DISPROW
LOOPA:
        Call    TRANSHEX            ; хөрвүүлээд,
        Call    DISP                ; дэлгэцэнд хэвлэнэ.
        Cmp     HEXCTR, 0FFH        ; сүүлчийн 16-тын тоо FF мөн үү?
        Je      ENDING              ; тийм бол дуусгана.
        Inc     HEXCTR              ; үгүй бол нэгээр нэмэгдүүлнэ.
        Jmp     LOOPA
ENDING:
        Mov     ax, 4c00h
```

```
        Int     21h
        Ret
MAIN endp

TRANSHEX proc near              ; 16-т уруу хөрвүүлэх дэд процедур
        Mov     AH, 00
        Mov     AL, HEXCTR      ; 16-тын 2 байт тоог (AX)-д авна.
        Shr     AX, CL          ; баруун тийш шилжүүлнэ.
        Lea     BX, XLATAB      ; хүснгэтийн хаягийг тавина.
        Mov     CL, 04          ; шилжүүлэх утгыг тавина.
        Xlat                    ; 16-тыг хөрвүүлнэ.
        Mov     [SI], AL        ; зүүн талын тэмдэгтийг хадгална.
        Mov     AL, HEXCTR
        Shl     AX, CL          ; зүүн талын оронг шилжүүлж, хасна.
        Shr     AL, CL
        Xlat                    ; 16-тыг хөрвүүлнэ.
        Mov     [SI]+1, AL      ; баруун талын тэмдэгтийг хадгална.
        Ret
TRANSHEX endp

DISP proc near                  ; дэлгэцэнд хэвлэх дэд процедур
        Mov     AL, HEXCTR
        Mov     [SI]+3, AL
        Cmp     AL, 1AH         ; EOF (файлын төгсгөл) тэмдэгт мөн үү?
        Je      D20             ; тийм бол D20 уруу үсэрнэ.
        Cmp     AL, 07H         ; 08-аас бага буюу тэнцүү юу?
        Jb      D30             ; тийм бол зөв, D30 уруу үсэрнэ.
        Cmp     AL, 10H         ; 0F-ээс их буюу тэнцүү юу?
        Jae     D30             ; тийм бол зөв, D30 уруу үсэрнэ.
D20:
        Mov     BYTE PTR [SI]+3, 20H
D30:
        Add     SI, 05           ; мөрний дараагийн байрлал уруу нэмэгдэнэ.
        Lea     DI, DISPROW+80
        Cmp     DI, SI
        Jne     D40
        Mov     AH, 40H         ; дэлгэцэнд хэвлэх хүсэлт
        Mov     BX, 01          ; дэлгэц эзэмших
        Mov     CX, 81          ; бүтэн мөр
        Lea     DX, DISPROW
        Int     21H
        Lea     SI, DISPROW     ; мөрийг хэвлэх хүсэлт
D40:
        Ret
DISP endp

CLRSCR proc near                ; Дэлгэц цэвэрлэх дэд процедур
        Mov     AX, 0600H
        Mov     BH, 03          ; өнгө (BW дэлгэцэнд 07 байна)
        Mov     CX, 0000
        Mov     DX, 184FH
        Int 10H
        Ret
CLRSCR endp
        End MAIN
```

Матрицан мэдээллийг эрэмбэлэх

Матрицан өгөгдлийн уруудах буюу өгсөх дарааллаар нь эрэмбэлэх хэрэглээ их гардаг.

- Жишээ программ

Энэ программ нь хэрэглэгчийг гараас 30 тэмдэгт хүртэл урттай нэрсийг оруулахыг зөвшөөрнө. Бүх нэрс орсны дараа нэрсийг өгсөх дарааллаар нь байрлуулж, дэлгэцэнд хэвлэнэ.

```
page    60, 132
title   NMSORT      Sort names entered from Terminal
;--------------------
.model small
.stack
        dw 32 dup(?)
.data
        namepar     label   byte            ; ---name parameter list
        maxlendb    21                      ; уртын хамгийн дээд хэмжээ
        namelen     db      ?               ; оруулсан тэмдэгтийн тоо
        namefld     db      21 dup(' ')     ; нэр

        crlf    db      13, 10, '$'
        endaddr     dw      ?
        messg1      db      'Name? ', '$'
        namectr     db      00
        nametab     db      30 dup(20 dup(' '))     ;Name table
        namesav     db      20 dup(?), 13, 10, '$'
        swapped     db      00
.code
BEGIN proc far
        mov ax, @data
        mov ds, ax
        mov es, ax
        cld
        lea     di, nametab
        call    clrscr                  ; дэлгэц цэвэрлэх дэд процедур
        call    setcur                  ; курсор тавих дэд процедур
loopa:
        call    acceptname              ; нэр оруулах дэд процедур
        cmp     namelen, 00             ; нэр орсон уу?
        jz      A30                     ; үгүй бол эрэмбэлнэ.
        cmp     namectr, 30             ; 30 нэр орсон уу?
        je      A30                     ; тийм бол эрэмбэлнэ.
        call    storename               ; хүснэгтэд нэрсийг хадгалах дэд проц.
        jmp     loopa
A30:                                    ;---оролтын төгсгөл
        call    clrscr                  ; дэлгэц цэвэрлэх дэд процедур
        call    setcur                  ; курсор тавих дэд процедур
        cmp     namectr, 01             ; 1 нэр орсон эсвэл нэр ороогүй юу?
        jbe     A40                     ; тийм бол программаас гарна.
        call    sortname                ; хадгалсан нарсийг эрэмбэлэх дэд проц.
        call    dispname                ; эрэмбэлсэн нэрсийг дэлгэцэнд хэвлэх
A40:                                    ;--------------------программаас гарна.
```

```
        mov     ax, 4c00h
        int 21h
        ret
BEGIN endp
;       Accept name as input
;       --------------------
acceptname proc near
        mov     ah, 09
        lea     dx, messg1              ; дэлгэцэнд асуултыг хэвлэнэ.
        int 21h
        mov     ah, 0ah
        lea     dx, namepar             ; нэрийг хүлээж авна.
        int 21h
        mov     ah, 09
        lea     dx, crlf                ; return / line feed
        int 21h

        mov     bh, 00                  ;нэрийн дараагийн тэмдэгтийг цэвэрлэнэ.
        mov     bl, namelen             ; тэмдэгтийн тоог тавина.
        mov     cx, 21
        sub     cx, bx                  ; үлдсэн уртыг тооцно.
B20:
        mov     namefld[bx], 20h        ; хоосон тэмдэгт тавина.
        inc     bx
        loop    b20
        ret
acceptname endp
;       Store name in table
;       -------------------
storename proc near
        inc     namectr                 ; нэрийн тоог нэмэгдүүлнэ.
        cld
        lea     si, namefld
        mov     cx, 10
        rep     movsw                   ; нэрийг хүснэгтэд оруулна.
        ret
storename endp
;       Sort name in table
;       ------------------------
sortname proc near
        sub     di, 40                  ; зогсох хаягийг тавина.
        mov     endaddr, di
g20:
        mov     swapped, 00             ; хүснэгтийн эхлэлийг
        lea     si, nametab             ;тавина.
g30:
        mov     cx, 20                  ; харьцуулах урт
        mov     di, si
        add     di, 20                  ; харьцуулах дараагийн нэр
        mov     ax, di
        mov     bx, si
        repe    cmpsb                   ; дараагийн нэртэй харьцуулна.
        jbe     g40                     ; өөрчлөхгүй
        call    exchange                ; байрыг нь солино.
g40:
```

```
        mov     si, ax
        cmp     si, endaddr             ; хүснэгтийн төгсгөл мөн үү?
        jbe     g30                     ; үгүй бол үргэлжлүүлнэ.
        cmp     swapped, 00             ; солилцсон уу?
        jnz     g20                     ; тийм бол үргэлжлүүлнэ.
        Ret                             ; үгүй бол эрэмбэлэхийг зогсооно.
sortname endp
;       Exchange table entries
;       ----------------------
exchange proc near
        mov     cx, 10
        lea     di, namesav
        mov     si, bx
        rep     movsw                   ; доод элементийг хадгална.
        mov     cx, 10
        lea     si, namesav
        rep     movsw                   ;дээд элементийг доод элементэд хийнэ.
        mov     cx, 10
        lea     si, namesav
        rep     movsw                   ; хадгалснаа дээд элементэд хийнэ.
        mov     swapped, 01             ; солилцсоныг тэмдэглэнэ.
        ret
exchange endp
;       Display sorted names
;       --------------------
dispname proc near
        lea     si, nametab
K20:
        lea     di, namesav             ; хүснэгтийн эхлэл
        mov     cx, 10
        rep     movsw
        mov     ah, 09
        lea     dx, namesav
        int 21h                         ; дэлгэцэнд харуулна.
        dec     namectr                 ; сүүлийн нэр мөн үү?
        jnz     K20                     ; үгүй бол давт
        ret                             ; тийм бол программаас гарна.
dispname endp
;       Clear screen
;       ------------
clrscr proc near
        mov     ax, 0600h
        mov     bh, 61h                 ; өнгө (BW бол 07)
        sub     cx, cx
        mov     dx, 184fh
        int     10h
        ret
clrscr endp
;       Set cursor
;       ----------
setcur proc near
        mov     ah, 02
        sub bh, bh
        sub dx, dx                      ; курсорыг 00, 00 байрлалд тавина.
        int     10h
```

ret
setcur endp

Төрөл, урт, хэмжээг тодорхойлох оператор

Төрөл заах TYPE оператор нь байт, word гэх мэт төрлийг, LENGTH оператор нь DUP факторыг, SIZE оператор нь байтын тоог тодорхойлно.

```
TABLEX    DW    10 DUP(?)          ; 10 word-тэй матриц
MOV       AX, TYPE  TABLEX         ; AX=0002
MOV       BX, LENGTH  TABLEX       ; BX=000A (10)
MOV       CX, SIZE  TABLEX         ; CX=0014 (20)
```

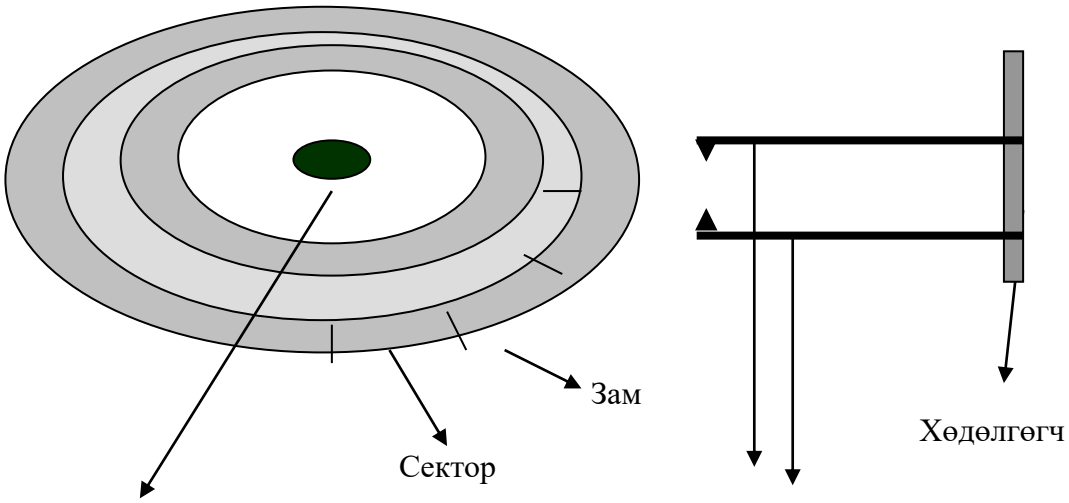
Эдгээрийг матрицаас хайлт хийх буюу эрэмбэлэх зэрэг үйлдлийг зогсоох зорилгоор ашиглаж болно. Жишээ нь:

```
CM    SI, SIZE  TABLEX
```

гэж нэмэгдэн өөрчлөгдөж байгаа SI регистр дэх матрицын хаягийг харьцуулж болно.

ДИСК

Диск гэдэг нь өгөгдлийг хадгалах зориулалттай хамгийн өргөн хэрэглэгддэг төхөөрөмж юм. Файлтай ажиллахын тулд дискний зохион байгуулалттай танилцах хэрэгтэй. Дискний гадаргуу нь олон тооны тойрог замуудаас бүрдэнэ. Зам бүр секторт хуваагдах бөгөөд сектор бүр 512 байтаас тогтоно. Жишээ нь: 5.25 инчийн диск нь 00-39 хүртэл дугаартай 40 ширхэг замтай, зам бүр нь 8-9 сектортой, 1 сектор нь 512 байт багтаамжтай.



Эргүүлэх гол Унших, бичих толгой

Дискэн дээрх өгөгдөлүүд нь файлын хэлбэрээр хадгалагдана.

Дискний багтаамж:

Өргөн хэрэглэгддэг дискнүүдийн багтаамжийг авч үзвэл:

Дискний төрөл	1 тал дээрх замын тоо	1 зам дээрх секторын тоо	1 сектор дахь байтын тоо	Дискний 2 тал дахь нийт хэмжээ (bytes)
DOS 2.0-оос өмнөх	40	8	512	327.680
DOS 2.0 –ийн дараах	40	9	512	368.640
Өндөр (high density)	80	15	512	1.228.800
3.5 inch	80	9	512	737.280

Хатуу дискний хувьд:

Дискний төрөл	1 тал дээрх замын тоо	1 зам дээрх секторын тоо	1 сектор дахь байтын тоо	Дискний 4 тал дахь нийт хэмжээ (bytes)
10 megabytes	306	17	512	10.653.696
20 megabytes	614	17	512	21.377.024

Дискний тал(толгой), замын дугаарууд нь 0-ээс, секторын дугаар 1-ээс эхэлнэ.

Directory (Сан)

Дискэнд хадгалагдаж байгаа мэдээллүүдийг бүртгэн зохион байгуулж байхын тулд DOS нь тодорхой хэдэн секторуудыг өөртөө хэрэглэдэг. Хатуу болон уян дискүүд нь өөрсдийн багтаамжаас хамаарч янз бүрээр зохион байгуулагддаг. 2 талтай, зам бүртээ 9 сектортой дискний хувьд:

Тал	Зам	Сектор	Агуулагдах мэдээ
0	0	1	Boot record
0	0	2-3	FAT (file allocation table)
0	0	4-7	Directory (сан)
1	0	1-3	Directory (сан)
1	0	4-9	Data files (өгөгдөл файлууд)
1	0	4→	Data files (өгөгдөл файлууд)

Өгөгдлүүд 1-р талын 0-р замын 3-9-р секторууд дээрээс эхлэнэ. Систем нь 0-р талын 1-р зам дээр, дараа нь 1-р талын 1-р зам, дараа нь 0-талын 2-р зам дээр гэх мэтээр бичлэгүүдийг хадгалдаг. Энэ мэдээллийг хадгалах онцлог нь дискний толгойн хөдөлгөөнийг бага байлгах зорилготой.

Хэрэв FORMAT/S гэж бичээд уян дискийг форматлах гэвэл DOS нь IBMBIO.COM, IBMDOS.COM файлуудаа дискний эхний секторт байрлуулна.

Бүх файлууд секторын эхлэл дээр байрлана. DOS нь файл бүрт зориулж дискний 0-р зам дээр сан (англиар-directory, оросоор-каталог) үүсгэнэ. Сан бүр нь нэр, огноо, хэмжээ, файлын дискэн дээрх байрлал зэргийг агуулна. Сан нь 32 байтаас бүрдэх ба форматыг нь бичвэл:

Байт	Зориулалт
0-7	Файлын нэр 1 дэх байт нь файлын төлөвийг илэрхийлнэ: 00-файлыг огт хэрэглээгүй, E5-файл устгагдсан, 2E-дэд директор
8-10	Файлын өргөтгөл
11	Файлын төрлийг тодорхойлох атрибут: 00-энгийн файл 01-зөвхөн уншиж болох файл 02-‘hidden’ буюу нууцлагдсан файл 04-DOS-ийн системийн файл 08-volume label 10h-дэд директор 20h-хатуу дискний архив файл
12-21	DOS-д нөөцлөгдсөн
22-23	Файлын үүссэн буюу сүүлд өөрчилсөн цаг-минут-секунд (2-тоор) цццццммм.ммммссссс
24-25	Файлын үүссэн буюу сүүлд өөрчилсөн жил-сар-өдөр (2-тоор) жжжжжжжс.сссөөөөө
26-27	Файлын эхлэх кластер Саны сүүлчийн 2 сектортой ижил байна. Эхний өгөгдлийн файл нь 002 гэсэн кластераас эхэлнэ.
28-31	Файлын хэмжээг байтаар илэрхийлнэ. Файлыг үүсгэхдээ DOS нь энэ хэмжээг тооцоод хадгална.

FAT (File Allocation Table) буюу файлын байрлалын хүснэгт

FAT гэдэг нь дискний орон зайг файлуудад хуваарилах зориулалттай хүснэгт юм. Хэрэв шинээр файл үүсгэх эсвэл байгаа файлаа засварлах бол DOS нь FAT-ийн утгуудыг засна. Boot-ийн бичлэг нь 1-р сектор дээр байх ба FAT нь 2-р сектороос эхэлнэ. FAT нь кластер бүрийн утгыг агуулна. Кластер гэдэг нь 1 талтай дискний хувьд 1 сектор, 2 талтай дискний хувьд 2 сектор, олон талтай дискний хувьд нэг сегмент болно. Иймд 2 талтай диск нь 2 FAT-ийг агуулна. FAT-ийн эхний байт нь гадаад төхөөрөмжийн төрлийг заана.

FE	Нэг талтай, 8 сектортой
FC	Нэг талтай, 9 сектортой
FF	Хоёр талтай, 8 сектортой
FD	Хоёр талтай, 9 сектортой
F9	Өндөр багтаамжтай, 1.2 meg
F8	Hard disk

2 ба 3 дахь байтууд нь FFFF гэсэн утгыг агуулна. Зарим төрлийн төхөөрөмжүүдийн эхлэх ба дуусах секторын дугааруудыг харуулбал:

Төхөөрөмж	Boot	FAT	Directory	Sectors/ Cluster
Нэг талтай, 8 сектортой	1	2-3	4-7	1
Нэг талтай, 9 сектортой	1	2-5	6-9	1
Хоёр талтай, 8 сектортой	1	2-3	4-10	2
Хоёр талтай, 9 сектортой	1	2-5	6-12	2
Өндөр багтаамжтай, 1.2 meg	1	2-15	16-29	1
XT hard disk	1	2-17	18-49	8
AT hard disk	1	2-83	84-115	4

4 дэх байтаас 12 битээр секторын дугаарыг заах ба DOS 3.0-оос хойшхи дээр 16 бит байж болно. Эхний 3 байт утга нь сектор 000 ба 001–т хамаарна. Эхний өгөгдлийн файл нь 002 сектор дээрээс эхэлнэ. FAT бүр 3 ширхэг 16-тын утгуудыг (12 бит) агуулах ба эдгээр нь доорх форматаар секторын байдлыг үзүүлнэ.

000	Энэ кластер нь одоогоор хэрэглэгдээгүй.
NNN	Файлын дараагийн кластерийн дугаар
FF7	Хэрэглэж болохгүй, эвдрэлтэй кластер
FFF	Файлын сүүлийн кластер

Жишээ нь: Дискэн дээр PAYROLL.ASM гэсэн ганц файл 002, 003, 004 гэсэн кластеруудыг эзэлсэн байна гэж үзье. Сан дотор энэ файл нь PAYROLL гэдэг нэртэй, ASM гэсэн өргөтгөлтэй, энгийн файл гэдгийг нь илэрхийлсэн 00 гэсэн утгатай, үүсгэсэн он-сар-өдөр, 002 гэсэн эхлэх секторын дугаар, файлын хэмжээг байтаар илэрхийлсэн байна. FAT нь доорх байдалтай байна.

FAT	FDF	FFF	003	004	FFF	000	000	000
Сектор	0	1	2	3	4	5	6	end

Эхний 2 байт нь 000 ба 001 секторууд дээрх сан байрлана. Энэ файлыг санах ойд оруулахын тулд систем нь доорх үйлдлүүдийг хийнэ.

1. DOS нь дискэнд хандаад, PAYROLLнэртэй ASM өргөтгөлтэй файлыг сангаас хайна.
2. DOS нь сангаас файлын эхний сектор (002)-ыг авааД, энэ секторын утгуудыг санах ой дахь хэсэгт дамжуулна.
3. 2 дахь секторын хувьд DOS нь 002 секторт харгалзах FAT-ын мэдээлэлд хандана. Дээрх жишээнд энэ утга нь 003 бөгөөд энэ нь уг файл 3 дугаар сектор дээр үргэлжилж байгааг заана. DOS нь энэ 3-р секторын утгыг санах ойн хэсэгт дамжуулна.

4. 3 дахь секторын хувьд DOS нь 003 секторт харгалзах FAT-ын мэдээлэлд хандана. Дээрх жишээнд энэ утга нь 004 буюу энэ нь уг файл 4-р сектор дээр үргэлжилж байгааг заана. DOS нь энэ 4-р секторын утгыг санах ойн хэсэгт дамжуулна.
5. DOS нь 004 секторт харгалзах FAT-ын мэдээлэлд хандахад энэ нь FFFh гэсэн утгатай буюу өөр мэдээлэл уг файлд байхгүйг илэрхийлнэ.

- Сан дотор файл бүрийн эхлэх кластерийн дугаар байрлана.
- FAT нь нэмэлт кластер бүрийн байрлалыг 16-тын 3 оронтой тоогоор заана.
- Хамгийн багадаа FAT нь FFFh гэсэн тоог агуулах ба эхний кластераас илүү гарах мэдээлэл байхгүйг заана.

Жишээ:

15 дугаар кластер дээрээс эхлэх файлыг сан зааж байна. Эхний FAT-ын бичлэгийг олохдоо:

1. 15*1.5=22.5 гэж үржүүлнэ.
2. FAT –ийн offset 22 ба 23 дахь байтуудад хандана. Тэд нь Fx, FF гэсэн утгуудтай байна гэж үзье.
3. Байтуудыг урвуулаад FF Fx болгоно.
4. 15 нь сондгой тоо учир эхний 3 орон болох FFF-ийг авна. Энэ нь өөр кластер байхгүй болохыг заана.

Одоо 4 кластер эзэлдэг 15-р кластераас эхлэх файлыг үзье:

FAT-ийн 22 дугаар кластераас эхлээд утгууд нь 6x 01 17 80 01 FF xF байдалтай байна гэе.

Эхний FAT-ийн оролт нь 15*1.5=22.5 учир 22 ба 23-р байтуудыг аваад байрыг нь солиход, 016x болно. 15 нь сондгой тоо учир эхний 3 оронг аваад, файлын 2 дахь кластер нь 016 гэсэн үг.

3 дахь кластерыг олохдоо: 16*1.5=24 учир 24, 25 байтууд 17, 80 гэсэн утгуудтай. 16 нь тэгш тоо учир 8017 гэсэн тооны сүүлийн 3 орон болох 017-г авч 3 дэх кластераа олно.

4 дахь кластерыг дээрх маягаар олоход 25, 26 байтууд 80, 01 гэсэн утгуудтай. 17 нь сондгой тоо учир 0180 гэсэн тооны эхний 3 орон болох 018-г авч 4 дэх кластераа олно.

5 дахь кластерийг дээрх маягаар олоход 27, 28-р offset байрлалуудад FF, xF утгууд байна. 18 нь тэгш тоо учир сүүлийн 3 орон болох FFF-г авч сүүлийн кластер болохыг нь мэднэ.

Хэрэв дискний төрлийг тодорхойлох хэрэгтэй бол FAT-ыг шууд шалгаж болно. Эсвэл DOS-ын 1BH, 1CH функцүүдийг хэрэглэнэ.

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