“Basics of microcontroller”

Lecture 7.

8085 microprocessor command format, addressing modes, commands

Lecturer: Onon Otgonbaatar, MD.

ТЭМДЭГТТЭЙ ХИЙХ ҮЙЛДЛҮҮД

Тэмдэгттэй ажиллах үйлдлүүд их өргөн хэрэглэгддэг. Жишээ нь: Нэр болон бичлэгүүдийг үсгийн дарааллаар нь эрэмбэлэх гэх мэт.

REP / REPZ / REPE / REPNE / REPNZ – repeat string prefix –ийг тэмдэгттэй ажиллах үйлдлүүдэд их хэрэглэнэ.

REP нь CX регистрийн утгыг 0 болтол тэмдэгттэй хийх үйлдлийг давтана. CX-ийн утгыг өмнө нь зааж өгнө. DF(direction flag)-аар давталтын чиглэлийг зааж өгнө. DF=0 бол зүүнээс баруун тийш чиглэлээр давтана. CLD командаар DF-ийг 0 болгож болно. DF=1 бол баруунаас зүүн тийш чиглэлээр давтана. STD командаар DF-ийг 1 болгоно.

REPZ / REPE нь ZF битийн утга 0 байх юм бол тэмдэгттэй хийх үйлдлийг давтана. ZF битийн утга 0 биш эсвэл CX регистрийн утга 0 болох юм бол давталтаас гарна.

REPNZ / REPNE нь ZF битийн утга 0 биш байх юм бол тэмдэгттэй хийх үйлдлийг давтана. ZF битийн утга 0 эсвэл CX регистрийн утга 0 болох юм бол давталтаас гарна.

Тэмдэгттэй ажиллах үндсэн 5 команд ассемблерт байна.

Команд	Гишүүд (шууд бичигдэхгүй)	Байттай ажиллахдаа	Word-тэй ажиллахдаа
MOVS	DI, SI	MOVSB	MOVSW
LODS	AL, SI эсвэл AX, SI	LODSB	LODSW
STOS	DI, AL эсвэл DI, AX	STOSB	STOSW
CMPS	SI, DI	CMPSB	CMPSW
SCAS	DI, AL эсвэл DI, AX	SCASB	SCASW

1. **MOVS** (mov string)
   1 байт эсвэл 1 word өгөгдлийг санах ой дахь 1 байрлалаас нөгөөд шилжүүлнэ. ES:DI хос регистрээр эхний гишүүнийг хаяглана. DS:SI хос регистрээр 2 дахь гишүүнийг хаяглана. Энэ команд нь голдуу REP prefix-тэй хамт хэрэглэгдэнэ. Жишээ нь:

```
STRING1 DB      20 DUP('*')
STRING2 DB      20 DUP(' ')
        …
        CLD                             ; чиглэлийн төлөвийн бит DF=0
        MOV     CX, 20                  ; 20 гэсэн утгыг тоолуурт өгнө
        LEA     DI, STRING2             ; тэмдэгтүүдийг оруулах хаяг
        LEA     SI, STRING1             ; илгээх тэмдэгтүүдийн хаяг
        REP     MOVSB                   ; STRING1→STRING2 уруу хийнэ.
```

2. **LODS** (load string)
   Санах ойн нэг байтыг AL-д эсвэл 1 word-ийг AH-д ачаална. Санах ойн хаяг нь DS:SI хос регистрээр хаяглагдана. DF битээс хамаарч SI нь нэмэгдэх буюу хорогдоно. Byte-тай ажиллаж байвал 1-ээр, word-тэй ажиллаж байвал 2-оор өөрчлөгдөнө. Жишээ нь:
   LODSB гэсэн үйлдэл нь доорх 2 үйлдэлтэй тэнцэнэ.

```
MOV     AL, [SI]        ; AL регистрт SI регистрийн зааж байгаа
                          утгыг авна
INC     SI              ; SI регистрийг нэгээр нэмэгдүүлж, хаягийг
                          өөрчилнө
```

3. **STOS** (store string)
   AL регистрийн эсвэл AH регистрийн утгыг санах ойд хадгална. Санах ойн хаяг нь ES:DI хос регистрээр хаяглагдана. DF битээс хамаарч DI нь нэмэгдэх буюу хорогдоно. Byte-тай ажиллаж байвал 1-ээр, word-тэй ажиллаж байвал 2-оор өөрчлөгдөнө. Жишээ нь:
   REP STOS гэсэн үйлдэл нь доорх үйлдлүүдтэй тэнцэнэ.

```
JCXZ    LABEL2
LABEL1:
    MOV         [DI], AL
    INC/DEC     DI
    LOOP        LABEL1
LABEL2:
    …
```

4. **CMPS** (compare string)
   Санах ойн үүрэн дэх дурын урттай тэмдэгтүүдийг нэг байт эсвэл 1 word урттайгаар хооронд нь харьцуулна. DS:SI ба ES:DI гэсэн хос регистрүүдээр санах ойн үүрүүдийн хаягийг тодорхойлно. DF чиглэлийн битээс хамаарч DI, SI шилжлэг хаягууд нь нэмэгдэх буюу хорогдоно. DF нь 0 бол зүүнээс баруун тийш харьцуулан, DI, SI шилжлэг хаягуудыг нэмэгдүүлнэ. Харин 1 бол баруунаас зүүн тийш харьцуулан, шилжлэг хаягуудыг хорогдуулах замаар харьцуулна. Энэ үйлдэл нь AF, CF, OF, PE, SF, ZF флагуудад нөлөөлнө. REP CMPS гэсэн үйлдлээр хичнээн ч байт тоонуудыг харьцуулж болно.

5. **SCAS** (compare string)
   AL регистрийн эсвэл AH регистрийн утгыг санах ойн үүрэн дэх нэг байт эсвэл 1 word урттай утгуудтай харьцуулна.

Тэмдэгттэй ажиллах үйлдлүүдийн жишээ программ

```
.data
name1       db      ‘Assemblers’
name2       db      10 dup (‘ ’)
name3       db      10 dup (‘ ’)

.code
Begin  proc far
        mov         ax, @data
        mov         ds, ax
        mov         es, ax

        call        mvsbproc    ;дэд процедурыг дуудах хэсэг
        call        mvswproc
        call        lodsproc
        call        stosproc
        call        cmpsproc
        call        scasproc
```

```
        mov     ax, 4c00h       ; програмаас гарах хэсэг
        int     21h
        ret
Begin  endp

;--------- MOVSB     ------------
mvsbproc proc near
        cld
        lea     si, name1
        lea     di, name2
        mov     cx, 10          ; 10 байтыг name1-ээс name2 уруу хийнэ.
        rep     movsb
        ret
mvsbproc endp
;--------- MOVSW     ------------
mvswproc proc near
        cld
        lea     si, name2
        lea     di, name3
        mov     cx, 5           ; 5 word-ыг name2-оос name3 уруу хийнэ.
        rep     movsw
        ret
mvswproc endp
;--------- LODSW     ------------
lodsproc proc near
        cld
        lea     si, name1       ; AX регистрт name1-ийн эхний word-ийг
        lodsw                   ;       хийнэ.
        ret
lodsproc endp
;--------- STOSW     ------------
stosproc proc near
        cld
        lea     di, name3
        mov     cx, 5
        mov     ax, 2020h       ; хоосон тэмдэгтийг name3 уруу хийнэ.
        rep     stosw
        ret
stosproc endp
;--------- CMPSB     ------------
cmpsproc proc near
        cld
        mov     cx, 10
        lea     si, name1
        lea     di, name2
        repe    cmpsb           ; name1, name2-ийг харьцуулна
        jne     g20             ; тэнцүү биш бол g20 уруу үсэрнэ
        mov     bh, 01
g20:
        mov     cx, 10
        lea     si, name2
        lea     di, name3
```

```
        repe        cmpsb           ; name2, name3-ийг харьцуулна
        je          g30             ; тэнцүү бол g30 уруу үсэрч, гарна.
        Mov         bl, 02
g30:
        ret
cmpsproc endp
;--------- SCASB     ------------
scasproc proc near
        cld
        mov         cx, 10
        lea         di, name1
        mov         al, 'm'
        repne       scasb           ; name1-д m тэмдэгтийг хайна.
        jne         h20             ; олдохгүй бол h20 уруу үсэрнэ
        mov         ah, 03
h20:
        ret
scasproc endp
        End Begin
```

Тэмдэгтийг хайж олоод орлуулж солих жишээ программ

Ямар нэг тэмдэгтийг өөр тэмдэгтээр солих шаардлага программд их гардаг. Жишээ нь: Paragraph-ийн эхлэл ба хуудасны төгсгөл зэргийг арилгах хэрэгтэй үе гардаг.
SCASB командыг ашиглан ampersand(&)-ийг олоод хоосон тэмдэгтээр солиё. Доорх жишээнд STRING+8 гэсэн хаягт "&" тэмдэгт маань байна. SCASB үйлдлээр эрж олоход DI регистрт STRING+9 гэсэн утга байх болно. Иймээс DI регистрийн утгыг 1-ээр хорогдуулаад хоосон тэмдэгтээ байрлуулах хаягаа тодорхойлно.

```
STRLEN      EQU     15                  ; тэмдэгтүүдийн урт
STRING      DB      'The time&is now'   ; тэмдэгтүүд
        …
        CLD                             ; DF=0
        MOV         AL, '&'             ; хайх тэмдэгт
        MOV         CX, STRLEN          ; тэмдэгтүүдийн урт
        LEA         DI, STRING          ; тэмдэгтүүдийн эхлэх хаяг
        REPNE       SCASB               ; хайлт
        JNZ         K20                 ; олсон уу?
        DEC         DI                  ; тийм бол хаягийг зас
        MOV         BYTE PTR[DI], 20H   ; хоосон тэмдэгтээр солино.
K20:
        RET
```

Тэмдэгтүүдийг олшруулах жишээ программ

MOVS командыг ашиглан ***--- гэсэн 6 тэмдэгтийг DISAREA талбарыг дуустал олшруулъя.

```
PATTERN     DB      '***---'
DISAREA     DB      42      DUP(?)
        …
        …
        CLD
```

```
        MOV         CX, 21
        LEA         DI, DISAREA
        LEA         SI, PATTERN
        REP         MOVSW
```

Программ биелэхдээ MOVSW үйлдлээр PATTERN-ийн (**) гэсэн эхний word утгыг DISAREA-ийн эхний word-д хийнэ. Дараа нь (*-) гэсэн 2 тэмдэгт, дараа нь (--) гэсэн 2 тэмдэгтүүдийг хийнэ. Эдгээрийн дараа DI регистр DISAREA+6 гэсэн утгатай болно. SI регистр нь PATTERN+6 гэсэн утга агуулна. Дараагаар нь дахин DISAREA-ийг DISAREA+6 хаягт, дараа нь DISAREA+2-ийг DISAREA+8 хаягт, DISAREA+4-ийг DISAREA+10 хаягт гэх мэтээр дэс дараалан хуулна. Ингэсээр DISAREA талбарыг дуустал дээрх 6 тэмдэгтийг хуулаад программ дуусна.

Тэмдэгтүүдийг баруун талд нь зэрэгцүүлэх жишээ программ

```
TITLE   EXRIGHT (COM) Right adjust displayed names
.MODEL SMALL
.STACK
      ORG    100H
.DATA
      NAMEDSP DB     'COMPUTER SCIENCE', 13, 10, '$'
.CODE
;                  Үндсэн программ
;-------------------------------------------------------------
MAIN    PROC    NEAR
      MOV      AX, @DATA
      MOV      DS, AX

      MOV      AX, 0600H
      SUB      DX, DX

      CALL     SCROLLSCR          ; дэлгэц цэвэрлэн, горим тавина.
      CALL     SETCURSOR          ; курсорыг байрлуулна.
      CALL     RIGHTADJ           ; баруун талд нь зэрэгцүүлнэ.

      MOV     AX, 4C00H           ; программаас гарах хэсэг
      INT      21H
      RET
MAIN    ENDP
; баруун талд нь зэрэгцүүлэх дэд процедур
;-------------------------------------------------------
RIGHTADJ  PROC  NEAR
        MOV         CX, 0010H
        MOV         DX, 004FH           ; 80 багана
        SUB         DX, CX              ; 80 – 16 буюу эхлэх координат
        CALL        SETCURSOR
        MOV         AH, 09
        LEA         DX, NAMEDSP
        INT         21H
        JMP         ENDING
ENDING:
        RET
RIGHTADJ ENDP
```

```
; дэлгэц цэвэрлэн, горим тавих дэд процедур
;-------------------------------------------------------------
SCROLLSCR PROC NEAR
        MOV         BH, 30
        MOV         CX, 00
        MOV         DX, 184FH
        INT         10H
        RET
SCROLLSCR ENDP
; курсорыг байрлуулах дэд процедур
;-------------------------------------------------
SETCURSOR  PROC NEAR
        MOV         AH, 02
        SUB         BH, BH
        INT         10H
        RET
SETCURSOR  ENDP
      END MAIN
```

Логик үйлдлүүд (Boolean operations)

Логик үйлдлүүд нь микросхемд чухал үүрэгтэйн адил мөн программын логикт ч их хэрэглэгддэг. Программд хэрэглэдэг логик командууд: AND, OR, XOR, TEST, NOT
AND, OR, XOR, TEST командууд нь битүүдийг тавих болон цэвэрлэх, ASCII өгөгдлүүдийг арифметикт оролцуулах зэрэгт голчлон ашигладаг ба регистр буюу санах ой дахь 1 байт эсвэл 1 word өгөгдлүүдтэй л ажиллана. 2 гишүүнтэй логик үйлдлийн үр дүнд CF, OF, PF, SF, ZF (AF хамаарахгүй) зэрэг төлөвийн битүүдэд нөлөөлнө.

AND: БА үйлдэл

Харгалзах бит 1	0	0	1	1
Харгалзах бит 2	0	1	0	1
Үр дүн	0	0	0	1

OR: БУЮУ үйлдэл

Харгалзах бит 1	0	0	1	1
Харгалзах бит 2	0	1	0	1
Үр дүн	0	1	1	1

XOR: Зөвхөн БУЮУ үйлдэл

Харгалзах бит 1	0	0	1	1
Харгалзах бит 2	0	1	0	1
Үр дүн	0	1	1	0

TEST: энэ нь хэдийгээр AND үйлдэлтэй ижил мэт боловч битүүдийг өөрчлөхгүй, зөвхөн флаг битүүдийг өөрчилдөг.

Харгалзах бит 1	0	0	1	1
Харгалзах бит 2	0	1	0	1
Үр дүн	0	0	0	1

Жишээ нь:

AL=1100 0101
BH= 0101 1100 байг.

```
1. AND      AL, BH      ; Үр дүн AL=0100 0100
2. OR       BH, AL      ; Үр дүн BH=1101 1101
3. XOR      AL, AL      ; Үр дүн AL=0000 0000
4. AND      AL, 00      ; Үр дүн AL=0000 0000
5. AND      AL, 0FH     ; Үр дүн AL=0000 0101
6. OR       CL, CL      ; Үр дүн SF ба ZF-т нөлөөлнө.
```

Доорх байдлаар нөхцөл шалгах үйлдлүүдийг хийж болно.

- OR үйлдлийг ашиглан нөхцөл шалгах жишээ:

```
OR    CX, CX            ; CX –ийг 0 эсэхийг шалгана.
JZ …  эсвэл  JS …       ; хэрэв 0 бол үсэр эсвэл сөрөг бол үсэр.
```

- TEST үйлдлийг ашиглан нөхцөл шалгах жишээ:

```
TEST        BL, 1111 0000B
JNZ         ….          ;Энэ нь BL-ийн ахлах 4 бит нь 0 эсэхийг шалгана.

TEST        AL, 0000 0001B
JNZ         ….          ;Энэ нь AL-ийг сондгой буюу тэгш тоог шалгана.

TEST        DX, 0FFH
JZ          ….          ;Энэ нь DX-ийн 0-тэй тэнцүү эсэхийг шалгана.
```

NOT үйлдэл нь регистр буюу санах ойн утгыг 0-ийг нь 1-ээр, 1-ийг нь 0 болгож урвуулна. Флаг битүүдэд нөлөөлөхгүй.

Жижиг үсгүүдийг том үсэг болгох

Том үсгүүд A-аас Z нь 41h-ээс 5Ah хүртэл кодтой. Жижиг үсгүүд a-аас z нь 61h-ээс 7Ah хүртэл кодтой. Ялгаа нь 5 дахь бит нь том үсэг дээр 0, жижиг үсэг дээр 1 болдог. Жишээ нь:

Бит	7 6 5 4 3 2 1 0
A үсэг	0 1 0 0 0 0 0 1
A үсэг	0 1 1 0 0 0 0 1
Z үсэг	0 1 0 1 1 0 1 0
z үсэг	0 1 1 1 1 0 1 0

Жишээ программ

```
Title   CASE Change lower case to uppercase
.model small
.stack  100h
.data
        TITLEX      db      ‘Change to upper case letters’
.code
MAIN        proc            near
            mov             ax,@data
            mov             es,ax
            mov             ds,ax
```

```
        lea     bx, TITLEX+1    ; өөрчлөх эхний тэмдэгт
        mov     cx, 31          ; өөрчлөх тэмдэгтийн тоо
b20:
        mov     ah, [bx]        ; TITLEX-ийн тэмдэгтийг авна.
        cmp     ah, 61h         ; том үсэг биш бол
        jb      b30             ; бол дараагийн
        cmp     ah, 7Ah         ; тэмдэгт
        ja      b30             ; уруу шилжинэ.
        and     ah, 11011111b   ; жижиг үсгийг том үсэг болгоно
        mov     [bx], ah
b30:
        inc     bx              ; дараагийн тэмдэгтийг сонгоно.
        loop    b20             ; 31 удаа давтана.
        ret
MAIN    endp
        End     MAIN
```

Шилжүүлэх ба эргүүлэх үйлдлүүд

- Шилжүүлэх үйлдэл

SHR(shift unsigned right) - баруун тийш шилжүүлэх
SHL(shift unsigned left) - зүүн тийш шилжүүлэх
SAR(shift arithmetic right) - баруун тийш шилжүүлэх
SAL(shift arithmetic left) - зүүн тийш шилжүүлэх

Жишээ 1:

```
Mov     cl, 03
Mov     ax, 1011 0111b          ; 1011 0111
Shr     ax, 1                   ; 0101 1011 CF-д шилжиж гарсан бит
Shr     ax, cl                  ; 0000 1011 CF-д шилжиж гарсан бит
                                нь орох ба шилжсэн нь 0-ээр дүүргэгдэнэ.
```

Жишээ 2:

```
Mov     cl, 03
Mov     ax, 1011 0111b          ; 1011 0111
Sar     ax, 1                   ; 1101 1011 CF-д шилжиж гарсан бит
Sar     ax, cl          ; 1111 1011 CF-д шилжиж гарсан бит нь орох ба
                        шилжсэн нь Sign битээр дүүргэгдэнэ.
```

Хэрэв зүүн тийш шилжүүлбэл (SAL) 0-ээр дүүргэгдэнэ. Баруун тийш үед тооныхоо тэмдгийг хадгалах зорилгоор SF-ээр дүүргэгдэнэ. (сөрөг, эерэг тэмдгээ авна)

Зүүн тийш шилжилт нь утгаа 2-оор үржихэд, баруун тийш шилжилт нь утгаа 2-т хуваахад хэрэглэгдэх ба эдгээр үйлдэл нь үржих, хуваах үйлдлүүдээс хамаагүй хурдан байдаг. Сондгой тоог хуваахад бага утгаа авах ба CF-ийг 1 болгодог. Жишээ нь: 5/2=2 ба CF=1 болно гэсэн үг.

- Эргүүлэх үйлдэл

ROR(rotate right) - баруун тийш эргүүлэх
ROL(rotate left) - зүүн тийш эргүүлэх
RCR(rotate with carry right) - баруун тийш CF-ийг оролцуулан эргүүлэх
RCL(rotate with carry left) - зүүн тийш CF-ийг оролцуулан эргүүлэх

Жишээ нь:

```
Mov     cl, 03
Mov     bx, 1011 0111b      ; 1011 0111
Ror     bx, 1               ; 1101 1011
Ror     bx, cl              ; 0111 1011
```

Rcr командын хувьд шилжиж гарсан бит нь CF-д очин, CF-д байсан бит хамгийн ахлах бит дээр очино.

```
Shl     ax, 1
Rcl     dx, 1 энэ 2 командаар DX:AX-ийн хосыг 2-оор үржүүлнэ.
```

Арифметик үйлдлүүд(+,-,/,*)

Нэмэх, хасах:

ADD – 2 утгыг нэмээд үр дүнг нь эхний операндад хадгална.
ADD AX, BX гэсэн команд нь AX+BX = AX үйлдлийг хийнэ.

ADC – 2 утгыг CF-тай хамт нэмээд үр дүнг нь эхний операндад хадгална. ADC AX, BX гэсэн команд нь AX+BX+Carry flag = AX үйлдлийг хийнэ.

SUB – 2 утгыг хасаад үр дүнг нь эхний операндад хадгална.
SUB AX, BX гэсэн команд нь AX-BX = AX үйлдлийг хийнэ.

SBB – 2 утгыг CF-тай хамт хасаад үр дүнг нь эхний операндад хадгална. SBB BX, CX гэсэн команд нь BX-CX-Carry flag = BX үйлдлийг хийнэ.

Дээрх арифметик үйлдлүүд нь доорх операндууд дээр хийгдэнэ.

Эхний операнд		2 дахь операнд
Регистр	–	Регистр
Регистр	–	Санах ой
Регистр	–	Шууд утга
Санах ой	–	Регистр
Санах ой	–	Шууд утга

Жишээ 1:
AH=1010 0011; BL=0111 1010; Carry flag=1
ADC AH, BH командын дараа

```
AH              1010 0011
BL        +     0111 1010
CF        +           1
Үр дүн:   =     0001 1101;          Carry flag=1
```

Жишээ 2:

```
ADD   AX, 20H       ; AX регистерийн утга+20h=AX регистерийн үр дүн үлдэнэ
```

Үржих / хуваах:

MUL (unsigned integer multiply**)** – Байт операндын хувьд AL регистрийн утгыг 8 бит регистр буюу санах ойн утгаар үржээд 16 бит үр дүнг AX регистрт хийнэ. Word операндын хувьд AX регистрийн утгыг 16 бит регистр буюу санах ойн утгаар үржээд 32 бит үр дүнг DX:AX регистрийн хосд хийнэ. MUL BL нь (BL)*(AL)=(AX) үйлдлийг хийнэ.

IMUL (signed integer multiply**)** – MUL үйлдэлтэй бараг ижил боловч ялгаа нь signed тоонууд дээр үйлдэл хийнэ.

Signed word нь дараах хязгаарт байна. +65535 → +32767 … -32768
Unsigned word нь дараах хязгаарт байна. +65535 →0 … +65535

DIV (unsigned integer devide**)** – Байт операндын хувьд AX регистрийн утгыг 8 бит регистр буюу санах ойн утганд хуваагаад 8 бит үр дүнг AL регистрт, үлдэгдлийг нь AH регистрт хийнэ. Word операндын хувьд DX:AX регистрийн хосд байгаа 32 бит утганд 16 бит регистр буюу санах ойн утгыг хуваагаад 16 бит үр дүнг AX регистрт, үлдэгдлийг нь DX регистрт хийнэ. DIV BL нь (AX)/(BL)=(AL) ; үлдэгдэлийг нь AH-д оруулах үйлдлийг хийнэ.

IDIV (signed integer devide**)** – DIV үйлдэлтэй бараг ижил боловч ялгаа нь signed тоонууд дээр үйлдэл хийнэ.

CBW (Convert byte to word) – AL регистрийн signed байт утгыг AX регистрт хийх ба AL-ийн ахлах бит буюу 7 дахь битээр AH-ийн битүүдийг дүүргэнэ. Флаг битүүдэд нөлөөлөхгүй.

CWD (Convert word to doubleword) – AX регистрийн signed 2 байт утгыг DX:AX регистрт хийх ба AX-ийн ахлах бит буюу 15 дахь битээр DX-ийн битүүдийг дүүргэнэ. Флаг битүүдэд нөлөөлөхгүй.

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