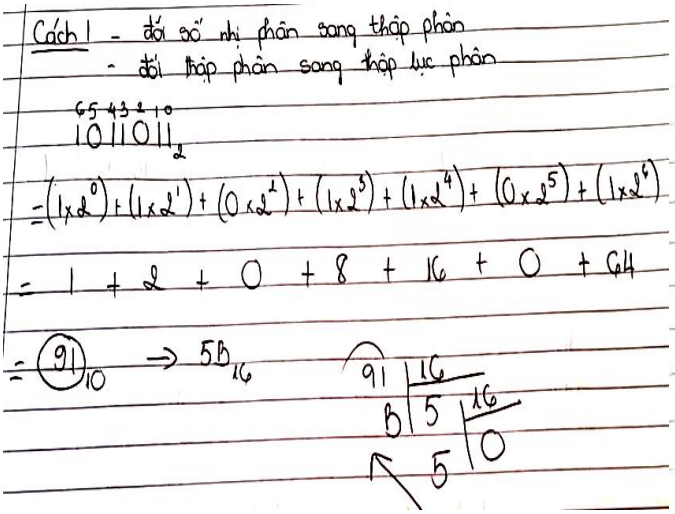

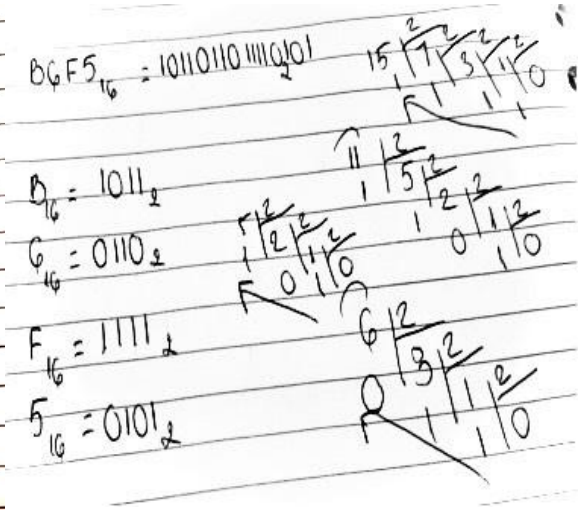

Cờ gửi (CF)

Có giá trị 1 khi có nhớ hoặc mượn từ bit MSB (tràn không dấu) trong phép cộng hoặc trừ, ngược lại là 0

Có thể thay đổi theo lệnh dịch hoặc quay

Cờ chẵn lẻ (PF)

Có giá trị 1 khi byte thấp của kết quả là chẵn

Có giá trị 0 khi byte thấp là lẻ

Một từ gọi là chẵn/lẻ khi số bit 1 của từ là chẵn/lẻ

Ví dụ: sau khi thực hiện một lệnh kết quả chứa trong AL là 11010010b thì PF=1

Cờ phụ (AF)

Có giá trị 1 khi có nhớ hoặc mượn từ 3 bit (tức có nhớ hoặc mượn từ 4 bit thấp), ngược lại bằng 0

Dùng trong các lệnh với số BCD

Thanh ghi cờ

Cờ không (ZF)

Có giá trị 1 khi kết quả bằng 0

Có giá trị 0 khi kết quả khác 0

Cờ dấu (SF)

Có giá trị 1 khi kết quả là âm (bit MSB là 1)

Có giá trị 0 khi kết quả là dương (bit MSB là 0)

Cờ tràn (OF)

Có giá trị 1 khi xảy ra trạng thái tràn tức giá trị (có dấu) vượt quá phạm vi giá trị cho phép

Ví dụ:

MOV AL, 80h
MOV BL, 80h
ADD AL, BL
	;AL=0
	;SF=0 vì MSB=0
	;PF=1 vì byte kết quả là chẵn
	;ZF=1 vì kết quả bằng không
	;CF=1 vì có nhớ
	;OF=1 vì có nhớ ra nhưng không nhớ vào MSB

CT tính địa chỉ vật lý ĐCVL= Địa chỉ Đoạn*10h + Địa chỉ ô ***Địa chỉ vật lý = (Segment) x 10H + (offset)***	VD: Tính địa chỉ vật lý tương ứng với địa chỉ logic 0A25h:CD09h | 0A25h*10h+CD09h= 16F59

Instruction	Description	Condition	Opposite Instruction
JZ , JE	Jump if Zero (Equal).	ZF = 1	JNZ, JNE
JC , JB, JNAE	Jump if Carry (Below, Not Above Equal).	CF = 1	JNC, JNB, JAE
JS	Jump if Sign.	SF = 1	JNS
JO	Jump if Overflow.	OF = 1	JNO
JPE, JP	Jump if Parity Even.	PF = 1	JPO
JNZ , JNE	Jump if Not Zero (Not Equal).	ZF = 0	JZ, JE
JNC , JNB, JAE	Jump if Not Carry (Not Below, Above Equal).	CF = 0	JC, JB, JNAE
JNS	Jump if Not Sign.	SF = 0	JS
JNO	Jump if Not Overflow.	OF = 0	JO
JPO, JNP	Jump if Parity Odd (No Parity).	PF = 0	JPE, JP

Instruction	Description	Condition	Opposite Instruction
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1	JNE, JNZ
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0	JE, JZ
JA , JNBE	Jump if Above (>). Jump if Not Below or Equal (not <=).	CF = 0 and ZF = 0	JNA, JBE
JB , JNAE, JC	Jump if Below (<). Jump if Not Above or Equal (not >=). Jump if Carry.	CF = 1	JNB, JAE, JNC
JAE , JNB, JNC	Jump if Above or Equal (>=). Jump if Not Below (not <). Jump if Not Carry.	CF = 0	JNAE, JB
JBE , JNA	Jump if Below or Equal (<=). Jump if Not Above (not >).	CF = 1 or ZF = 1	JNBE, JA

Instruction	Description	Condition	Opposite Instruction
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1	JNE, JNZ
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0	JE, JZ
JG , JNLE	Jump if Greater (>). Jump if Not Less or Equal (not <=).	ZF = 0 and SF = OF	JNG, JLE
JL , JNGE	Jump if Less (<). Jump if Not Greater or Equal (not >=).	SF <> OF	JNL, JGE
JGE , JNL	Jump if Greater or Equal (>=). Jump if Not Less (not <).	SF = OF	JNGE, JL
JLE , JNG	Jump if Less or Equal (<=). Jump if Not Greater (not >).	ZF = 1 or SF <> OF	JNLE, JG

So sánh SRAM và DRAM:

1. Tính chất của bộ nhớ:
   - SRAM (Static Random Access Memory): Lưu trữ dữ liệu bằng cách sử dụng flip-flops và không yêu cầu làm mới định kỳ.
   - DRAM (Dynamic Random Access Memory): Lưu trữ dữ liệu dưới dạng điện tích trên các capacitor và cần được làm mới định kỳ vì điện tích sẽ giảm theo thời gian.
2. Tốc độ truy cập:
   - SRAM: Nhanh hơn so với DRAM vì không có quá trình làm mới định kỳ.
   - DRAM: Chậm hơn do cần làm mới định kỳ để duy trì dữ liệu.
3. Tính ổn định:
   - SRAM: Ổn định hơn do không cần làm mới dữ liệu.
   - DRAM: Cần duy trì và làm mới định kỳ, nếu không có quá trình làm mới, dữ liệu có thể bị mất.
4. Dung lượng và chi phí:
   - SRAM: Có dung lượng thấp hơn và chi phí cao hơn.
   - DRAM: Có thể sản xuất với dung lượng lớn hơn và chi phí thấp hơn.
5. Tiêu thụ năng lượng:
   - SRAM: Tiêu thụ năng lượng cao hơn do cấu trúc phức tạp hơn.
   - DRAM: Tiêu thụ năng lượng thấp hơn.

So sánh các loại bộ nhớ ROM:

1. ROM (Read-Only Memory): Là loại bộ nhớ chỉ đọc, nghĩa là dữ liệu được lưu trữ không thay đổi trong quá trình sử dụng.
2. PROM (Programmable Read-Only Memory): Có thể được lập trình một lần bằng cách sử dụng thiết bị lập trình, nhưng sau đó không thể thay đổi nữa.
3. EPROM (Erasable Programmable Read-Only Memory): Có thể được lập trình và xóa nhiều lần. Để xóa dữ liệu, cần sử dụng ánh sáng UV.
4. EEPROM (Electrically Erasable Programmable Read-Only Memory): Tương tự như EPROM, nhưng có thể xóa dữ liệu điện tử mà không cần sử dụng ánh sáng UV.
5. Flash Memory: Một loại EEPROM tiên tiến, thường được sử dụng trong các thiết bị lưu trữ di động và thẻ nhớ. Nó có thể xóa và ghi dữ liệu theo các khối nhỏ, giảm tối đa quá trình xóa và ghi toàn bộ.
6. Mask ROM: Dữ liệu được lưu trữ trong quá trình sản xuất và không thể thay đổi sau đó.

So sánh các loại bộ nhớ ROM:

ROM (Read-Only Memory) là một loại bộ nhớ không thể ghi dữ liệu vào nó sau khi nó đã được lập trình.
Có nhiều loại bộ nhớ ROM như ROM không xóa được (Mask ROM), ROM có thể xóa được (EPROM), ROM chỉ có thể xóa được một lần (OTP ROM) và ROM có thể xóa được điện tử (EEPROM).
Mask ROM là ROM được lập trình trong quá trình sản xuất và không thể thay đổi nội dung sau khi được tạo ra.
EPROM (Erasable Programmable ROM) có thể xóa và lập trình lại bằng cách sử dụng ánh sáng tử ngoại để xóa dữ liệu trong chip.
OTP ROM (One-Time Programmable ROM) chỉ có thể được lập trình một lần và không thể thay đổi nội dung sau khi lập trình.
EEPROM (Electrically Erasable Programmable ROM) có thể xóa và lập trình lại điện tử bằng cách sử dụng điện áp.
Kiến trúc máy tính(Computer architecture): nghiên cứu những thuộc tính của một hệ thống mà người lập trình có thể nhìn thấy được, những thuộc tính quyết định trực tiếp đến việc thực thi một chương trình tính toán, xử lý dữ liệu.
• Cấu trúc máy tính(Computer organization): nghiên cứu về các thành phần chức năng và sự kết nối giữa chúng
để tạo nên một máy tính, nhằm thực hiện những chức năng và tính năng kỹ thuật của kiến trúc.
Các khối chức năng cơ bản của máy tính
• Đơn vị xử lý trung tâm (CPU): Khối điều khiển và xử lý dữ liệu
• Bộ nhớ (Memory): Khối lưu trữ dữ liệu
• Thiết bị nhập xuất (I/O): Khối chức năng cung cấp dữ liệu cho máy tính xử lý, hoặc phản ánh dữ liệu đã được xử lý do máy tính cung cấp
• Kênh liên kết hệ thống (Bus): Các kênh truyền dẫn cung cấp sự liên lạc và trao đổi dữ liệu giữa các khối trên
=>-RAM cho phép ghi và đọc dữ liệu tạm thời, trong khi ROM chỉ cho phép đọc dữ liệu. RAM là bộ nhớ dùng để lưu trữ tạm thời dữ liệu và chương trình đang hoạt động trong thời gian thực, trong khi ROM là bộ nhớ dùng để lưu trữ dữ liệu cố định. RAM có thể chỉnh sửa dữ liệu trong khi ROM không cho phép chỉnh sửa.

NHẬP XUẤT THẬP LỤC PHÂN (HEXA)

```
.MODEL SMALL
.DATA
    MSG1  DB 13,10,'Nhap mot
so:$'
    TB DB 13,10,'So hexa vua
nhap:$'
.STACK 100h
.CODE
;Nhap so thap luc phan chu vao BX
    XOR BX,BX
    MOV CL,4 ;DICH 4 BIT
    ;MOV DX,OFFSET MSG1
    ;MOV AH,9
    ;INT 21H
    MOV AH,9
    MOV AH,1
N2:
    INT 21h
    CMP AL,13
    JE N8
    CMP AL, 39h ;Ky tu la chu so
    JG Letter
    AND AL,0Fh ;Doi thanh tri
    JMP Shift  ;dua vao BX
Letter:
    SUB AL,37h ;Doi chu ra tri
Shift:
    SHL BX,CL
    OR BL,AL
    JMP N2
N8:
;Xuat so thap luc phan trong BX
    MOV AX,@DATA
    MOV DS,AX
    LEA DX,TB
    MOV AH,9
    INT 21h
    MOV CX,4
PrintHex:
    MOV DL,BH
    SHR DL,4
    CMP DL,9
    JG PrintLetter
    OR DL,30h
    MOV AH,2
    INT 21h
    JMP Exit
PrintLetter:
    ADD DL,37h
    MOV AH,2
    INT 21H
Exit:
    SHL BX,4
    LOOP PrintHex

    MOV AH,4Ch
    INT 21h
END
```

NHẬP XUẤT NHỊ PHÂN (BINARY)

```
.MODEL SMALL
.STACK 100h
.DATA
    CR EQU 13
    LF EQU 10
    MSG1 DB 'NHAP MOT SO NHI
PHAN: $'
    MSG2 DB CR,LF,'SO NHI PHAN DA
NHAP: $'
.CODE
    MOV AX, @DATA
    MOV DS, AX
    XOR BX, BX
    MOV AH, 9

    LEA DX, MSG1
    INT 21h
    MOV AH, 1
NHAP:
    INT 21h
    CMP AL, CR
    JE XUAT

    AND AL, 0Fh ;chuyen ky tu
thanh so
    SHL BX, 1
    OR BL, AL
    JMP NHAP
XUAT:
    MOV CX,16
    MOV AH,9
    LEA DX, MSG2
    INT 21H
    LAP:
        ROL BX,1
        JC XUAT1
        MOV DL, 30h
        MOV AH, 2
        INT 21h
        JMP THOAT1
    XUAT1:
        MOV DL, 31h
        MOV AH, 2
        INT 21h
    THOAT1:
        DEC CX
        CMP CX, 0
        JNE LAP

    MOV AH, 4CH
    INT 21h
END
```

NHẬP XUẤT THẬP PHÂN (DECIMAL)

```
.MODEL SMALL
.STACK 100h
.DATA
    TB1 DB 13,10,'NHAP SO THAP
PHAN: $'
    TB2 DB 13,10,'SO THAP PHAN VUA
NHAP: $'
.CODE
    MOV AX, @DATA
    MOV DS, AX
    CALL INDEC
    LEA DX, TB2
    MOV AH, 9
    INT 21h
    CALL OUTDEC
;NHAP
INDEC PROC
    PUSH SI
    PUSH BX
    PUSH CX
    PUSH DX
    @BEGIN:
        LEA DX, TB1
        MOV AH, 9
        INT 21h
        MOV AH, 1
        INT 21h
        XOR BX, BX
        XOR CX, CX
        CMP AL, '-'
        JE @MINUS
        CMP AL, '+'
        JE @PLUS
        JMP @REPEAT2
    @MINUS:
        MOV CX, 1
    @PLUS:
        INT 21h
    @REPEAT2:
        CMP AL, '0'
        JNGE @NOTDIGIT
        CMP AL, '9'
        JNLE @NOTDIGIT
        AND AL, 0Fh
        MOV SI, AX

        MOV AX, 10
        MUL BX
        MOV BX, SI
        XOR BH, BH
        ADD BX, AX

        MOV AH, 1
        INT 21h
        CMP AL, 13
        JNE @REPEAT2

        MOV AX, BX
        OR CX, CX
        JZ @EXIT
        NEG AX
    @EXIT:
        POP SI
        POP DX
        POP CX
        POP BX
        RET
    @NOTDIGIT:
        MOV AH, 2
        MOV DL, 10
        INT 21h
        MOV DL, 13
        INT 21h
        JMP @BEGIN

INDEC ENDP
;XUAT
OUTDEC PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    OR AX, AX
    ;CMP AX, 0
    JG @ENDIF_1
    PUSH AX
    MOV DL, '-'
    MOV AH, 2
    INT 21h
    POP AX
    NEG AX
    @ENDIF_1:
        XOR CX, CX
        MOV BX, 10
    @REPEAT_1:
        XOR DX, DX
        DIV BX
        PUSH DX
        INC CX
        OR AX, AX
        JNZ @REPEAT_1

        MOV AH, 2

    @FOR:
        POP DX
        OR DL, 30h
        INT 21h
        LOOP @FOR

        POP AX
        POP BX
        POP CX
        POP DX

        RET

OUTDEC ENDP
```

XUẤT NGÀY THÁNG NĂM

```
.MODEL SMALL
.STACK 100h
.DATA
    DAY DB 13,10,'NGAY : $'
    MONTH DB 13,10,'THANG: $'
    YEAR DB 13,10,'NAM: $'
.CODE
    INCLUDE 'emu8086.inc'

    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 2Ah
    INT 21h

    PUSH CX
    XOR BX, BX
    MOV BL, DH
    PUSH BX
    XOR BX, BX
    MOV BL, DL
    PUSH BX

    ;XUAT NGAY
    LEA DX, DAY
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    ;XUAT THANG
    LEA DX, MONTH
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    ;XUAT NAM
    LEA DX, YEAR
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    MOV AH, 4Ch
    INT 21h

DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS

END
```

XUẤT GIỜ PHÚT GIÂY

```
.MODEL SMALL
.STACK 100h
.DATA
    HOUR DB 13,10,'GIO : $'
    MINUTE DB 13,10,'PHUT: $'
    SECOND DB 13,10,'GIAY: $'
.CODE
    INCLUDE 'emu8086.inc'

    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 2Ch
    INT 21h

    XOR BX, BX
    MOV BL, DH
    PUSH BX
    XOR BX, BX
    MOV BL, CL
    PUSH BX
    XOR BX, BX
    MOV BL, CH
    PUSH BX

    ;XUAT GIO
    LEA DX, HOUR
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    ;XUAT PHUT
    LEA DX, MINUTE
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    ;XUAT GIAY
    LEA DX, SECOND
    MOV AH, 9
    INT 21h
    POP AX
    CALL PRINT_NUM

    MOV AH, 4Ch
    INT 21h
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

TÍNH CHU VI & DIỆN TÍCH HÌNH CHỮ NHẬT

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100h
.DATA
    TB1 DB 13,10,'NHAP DO DAI CUA
CHIEU DAI(ENTER DE TIEP TUC): $'
    TB2 DB 13,10,'NHAP DO DAI CUA
CHIEU RONG (ENTER DE TIEP TUC): $'
    TB3 DB 13,10,'CHU VI CUA HINH
CHU NHAT: $'
    TB4 DB 13,10,'DIEN TICH CUA HINH
CHU NHAT: $'

.CODE
    MOV AX, @DATA
    MOV DS, AX
    XOR BX, BX
    XOR CX, CX
    ;NHAP CHIEU RONG
    LEA DX, TB2
    MOV AH, 9
    INT 21h
    CALL NHAP
    MOV CX, BX
    PUSH BX
    PUSH CX

    ;NHAP CHIEU DAI
    LEA DX, TB1
    MOV AH, 9
    INT 21h
    CALL NHAP
    POP CX
    MOV DX, BX
    PUSH DX

    ;TINH CHU VI
    LEA DX, TB3
    MOV AH, 9
    INT 21h

    MOV AH, 0
    MOV AX, CX
    ADD AX, BX
    MOV CX, 2
    AND CX, 0Fh
    MUL CX
    CALL PRINT_NUM

    ;TINH DIEN TICH
    LEA DX, TB4
    MOV AH, 9
    INT 21h

    POP AX
    POP BX
    MUL BX
    CALL PRINT_NUM

    MOV AH, 4Ch
    INT 21h
;-------------------
```

```
;NHAP-----------
NHAP PROC
    @BEGIN:
        MOV AH, 1
        INT 21h
        XOR BX, BX
        XOR CX, CX
        CMP AL, '-'
        JE @MINUS
        CMP AL, '+'
        JE @PLUS
        JMP @REPEAT2
    @MINUS:
        MOV CX, 1
    @PLUS:
        INT 21h
    @REPEAT2:
        CMP AL, '0'
        JNGE @NOTDIGIT
        CMP AL, '9'
        JNLE @NOTDIGIT

        AND AL, 0Fh
        MOV SI, AX

        MOV AX, 10
        MUL BX
        MOV BX, SI
        XOR BH, BH
        ADD BX, AX

        MOV AH, 1
        INT 21h
        CMP AL, 13
        JNE @REPEAT2

        MOV AX, BX
        OR CX, CX
        JZ @EXIT
        NEG AX
    @EXIT:
        RET
    @NOTDIGIT:
        MOV AH, 2
        MOV DL, 10
        INT 21h
        MOV DL, 13
        INT 21h
        JMP @BEGIN
NHAP ENDP
;------------------------------

DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

SO SÁNH THỨ TỰ TRƯỚC SAU CỦA 2 KT

```
.MODEL SMALL
.STACK 100H
.DATA
PR1 DB 10,13,'NHAP KI TU DAU TIEN:$'
PR2 DB 10,13,'NHAP KI TU THU HAI:$'
PR3 DB 10,13,'KI TU 1 TRUOC KI TU 2$'
PR4 DB 10,13,'KI TU 1 SAU KI TU 2$'
PR5 DB 10,13,'KI TU 1 BANG KI TU 2$'
.CODE
MOV AX,@DATA
MOV DS,AX

KIEMTRAKT1:

MOV DX,OFFSET PR1
MOV AH,9
INT 21H

MOV AH,1
INT 21H
MOV CL,AL

CMP CL,'A'
JL KIEMTRAKT1
CMP CL,'Z'
JG KIEMTRAKT1
JMP KIEMTRAKT2

KIEMTRAKT2:

MOV DX,OFFSET PR2
MOV AH,9
INT 21H

MOV AH,1
INT 21H

MOV CH,AL

CMP CH,'A'
JL KIEMTRAKT2
CMP CH,'Z'
JG KIEMTRAKT2
JMP CHECK

CHECK:
CMP CL,CH
JL TRUOC
JG SAU
JE BANG

TRUOC:
MOV DX,OFFSET PR3
JMP GREETING
SAU:
MOV DX,OFFSET PR4
JMP GREETING
BANG:
MOV DX,OFFSET PR5
JMP GREETING
GREETING:
MOV AH,9
INT 21H

END
```

XUẤT 5 KT TRƯỚC/SAU

```
.MODEL SMALL
.STACK 100H
.DATA

PR1 DB 10,13,'NHAP KI TU:$'
PR2 DB 10,13,'5 KI TU DUNG
TRUOC:$'
PR3 DB 10,13,'5 KI TU DUNG
SAU:$'

.CODE
MOV AX,@DATA
MOV DS,AX

MOV DX,OFFSET PR1
MOV AH,9
INT 21H

MOV AH,1
INT 21H

MOV BL,AL

MOV DX,OFFSET PR2
MOV AH,9
INT 21H
SUB BL,6

MOV CX,5

PRINTLOOP1:

MOV AH,2
INC BL
MOV DL,BL
INT 21H
LOOP PRINTLOOP1

MOV DX,OFFSET PR3
MOV AH,9
INT 21H
ADD BL,1

MOV CX,5
PRINTLOOP:

MOV AH,2
INC BL
MOV DL,BL
INT 21H
LOOP PRINTLOOP


END
```