Lý thuyết

Câu 1:

Trình bày các nội dung sau trong kiến trúc Intel x86 32bit.

- Cấu trúc một chương trình hợp ngữ.

- Danh sách các thanh ghi được sử dụng trong chương trình.

- Danh sách các cờ

Trả lời:

1. Cấu trúc chương trình

```
.386
.model flat,stdcall
.stack 4096
;///// function and procedure
.data
; declare variables here
.code
main PROC
        ; /////write your code here
        ;//////INVOKE ExitProcess,0
main ENDP
; //////(insert additional procedures here)
END main
```

**** Giải thích:***

- Lệnh **.386** xác định CPU tối thiểu cần thiết cho chương trình này (intel 386)

- Lệnh **.MODEL** hướng dẫn trình biên dịch tạo mã cho chế độ bảo vệ chương trình, STDCALL cho phép gọi các chức năng của MS-Window

- Hai chỉ thị **PROTO** tuyên bố cho các thủ tục được sử dụng bởi chương trình này

+ ExitProcess là chức năng của MS-Window tạm dừng chương trình hiện tại.

+ DumpRegs là một thủ tục từ thư viên irvine32 hiển thị các thanh ghi

- **INVOKE** là một lệnh biên dịch hợp ngữ gọi một thủ tục hay gọi hàm.

Chương trình này kết thúc bằng lệnh gọi hàm ExitProcess, truyền cho nó mã trả về bằng 0.

2. Danh sách các thanh ghi, chỉ tập trung vào 32 bit

3. Danh sách các cờ

- Overflow Flag (OF) :
- Direction Flag (DF)
- Interrupt Flag (IF)
- Trap Flag (TF)
- Sign Flag (SF)
- Zero Flag (ZF)
- Auxiliary Carry Flag (AF)
- Parity Flag (PF)
- Carry Flag (CF) Danh sách các cờ, ý nghĩa và tác động.

Câu 2:

Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: MOV, INC, DEC, ADD, SUB, NEG.

Trả lời:

*** MOV**: Lệnh MOV sao chép dữ liệu từ nguồn sang toán hạng đích. Nội dung của toán hạng đích thay đổi, nhưng toán hạng nguồn không thay đổi.

- Cú pháp: MOV *toán hạng đích, toán hạng nguồn*

Cụ thể:

MOV reg,reg

MOV mem,reg

MOV reg,mem

MOV mem,imm

MOV reg,imm

- Quy tắc:

+ Cả hai toán hạng phải có cùng kích thước.

+ Cả hai toán hạng không thể là toán hạng bộ nhớ.

+ CS, EIP, IP không thể là toán hạng đích.

+ Không thể di chuyển giá trị tức thời vào thanh ghi phân đoạn.

- Tác động vào cờ: Lệnh MOV không ảnh hưởng đến các cờ.

*** INC/DEC:** Lệnh INC (tăng)/DEC (giảm) tương ứng là toán hạng đích cộng 1/trừ 1.

- Cú pháp:

INC reg/mem

DEC reg/mem

- Tác động vào cờ:
+ OF, SF, ZF, AF, PF được thay đổi theo kết quả của giá trị toán hạng đích.

+ INC/DEC không ảnh hưởng đến cờ CF.

*** ADD:** Thêm toán hạng nguồn vào toán hạng đích có cùng kích thước, kết quả phép cộng lưu vào toán hạng đích, toán hạng nguồn không thay đổi.

- Cú pháp: ADD *toán hạng đích, toán hạng nguồn*.

- Tác động vào cờ: CF, AF, ZF, SF, OF, PF được thay đổi theo kết quả của giá trị toán hạng đích.

*** SUB:** Trừ toán hạng nguồn ra khỏi toán hạng đích, kết quả phép trừ được lưu vào toán hạng đích toán hạng nguồn không đổi.

- Cú pháp: SUB *toán hạng đích, toán hạng nguồn*.

- Tác động vào cờ: CF, AF, ZF, SF, OF, PF được thay đổi theo kết quả của giá trị toán hạng đích.

*** NEG:** Lệnh NEG (phủ định) đảo ngược dấu của toán hạng đích bằng cách lấy bù hai.

- Cú pháp:

NEG reg

NEG mem

- Tác động vào cờ: CF, AF, ZF, SF, OF, PF được thay đổi theo kết quả của giá trị toán hạng đích.

Câu 3:
Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: PUSH, PUSHAD, PUSHFD, POP, POPAD, POPFD, RET

Trả lời:

* PUSH: Lệnh PUSH đầu tiên giảm ESP và sau đó sao chép toán hạng nguồn vào ngăn xếp. Toán hạng 16 bit làm cho ESP giảm đi 2. Toán hạng 32 bit làm cho ESP giảm 4.

- Cú pháp:

PUSH reg/mem16

PUSH reg/mem32

PUSH imm32

- Tác động vào cờ:

* PUSHAD: Lệnh PUSHAD đẩy tất cả các thanh ghi 32 bit lên ngăn xếp theo thứ tự sau: EAX, ECX, EDX, EBX, ESP (giá trị trước khi thực thi PUSHAD), EBP, ESI và EDI.

- Cú pháp: PUSHAD

- Tác động vào cờ:

* PUSHFD: Lệnh PUSHFD đẩy thanh ghi EFLAGS 32 bit vào ngăn xếp.

- Cú pháp : PUSHFD

- Tác động vào cờ:

* POP: Lệnh POP đầu tiên sao chép nội dung của phần tử ngăn xếp được ESP trỏ tới vào một toán hạng đích 16 hoặc 32-bit và sau đó tăng ESP. Nếu toán hạng là 16 bit, ESP được tăng thêm 2; nếu toán hạng là 32 bit, ESP được tăng thêm 4.

- Cú pháp:

POP reg / mem16

POP reg / mem32

- Tác động vào cờ:

* POPAD: Lệnh POPAD lấy các thanh ghi giống nhau ra khỏi ngăn xếp theo thứ tự ngược lại.

- Cú pháp: POPAD

- Tác động vào cờ:

* POPFD: POPFD đưa giá trị ngăn xếp vào EFLAGS

- Cú pháp: POPFD

- Tác động vào cờ:

* RET: Thủ tục sử dụng một lệnh RET (quay lại từ thủ tục) để đưa bộ xử lý trở lại điểm trong chương trình nơi thủ tục được gọi.

- Cú pháp: RET

- Tác động vào cờ:

Câu 4:

Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: XCHG, JMP, LOOP, truy cập Address (Địa chỉ trực tiếp và gián tiếp) .

* XCHG: Trao đổi nội dung 2 toán hạng nguồn và đích.

- Cú pháp:

XCHG reg,reg

XCHG reg,mem

XCHG mem,reg

- Tác động vào cờ:

* JMP: Lệnh nhảy không điều kiện, lệnh JUMP nhảy đến đích được xác định bởi một nhãn.

- Cú pháp: *JMP destination*

- Tác động vào cờ:

* LOOP: Thực hiện vòng lặp theo bộ đếm ECX, lặp lại một khối câu lệnh một số lần cụ thể. ECX được sử dụng như một bộ đếm tự động và được giảm dần mỗi khi vòng lặp lặp lại.

- Cú pháp: *LOOP destination*

- Tác động vào cờ:

* Truy cập địa chỉ (Địa chỉ trực tiếp và địa chỉ gián tiếp):

Câu 5:
Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: OFFSET, PTR, TYPE, SIZEOF, LENGTHOF

Trả lời:

* OFFSET:

- Toán tử OFFSET trả về địa chỉ ô nhớ tính bằng byte, của nhãn từ đầu phân đoạn dữ liệu kèm theo. Giá trị được trả về bởi OFFSET là một con trỏ.

```
.data
bVal BYTE ?
wVal WORD ?
.code
mov esi, OFFSET bVal ; ESI = 00404000h
mov esi, OFFSET wVal ; ESI = 00404001h
```

- OFFSET cũng có thể được áp dụng cho một toán hạng bù trừ trực tiếp.

```
.data
```

```
myArray WORD 1, 2, 3, 4, 5
.code
mov esi, OFFSET myArray + 4 ; ESI points to the third integer
```

* PTR:

- Toán tử PTR ghi đè kiểu mặc định của nhãn (biến), cung cấp tính linh hoạt để truy cập một phần của biến.

Ví dụ:

```
.data
myDouble DWORD 12345678h
.code
mov ax, myDouble ; error: size mismatch
mov ax, WORD PTR myDouble ; AX = 5678h
mov al, BYTE PTR myDouble ; AL = 78h
mov al, BYTE PTR [myDouble+1] ; AL = 56h
mov al, BYTE PTR [myDouble+2] ; AL = 34h
mov ax, WORD PTR [myDouble+2] ; AX = 1234h
```

- PTR cũng có thể được sử dụng để kết hợp các phần tử của kiểu dữ liệu nhỏ hơn và chuyển chúng vào một toán hạng lớn hơn. CPU sẽ tự động đảo ngược các byte.

```
.data
myBytes BYTE 12h, 34h, 56h, 78h
.code
mov ax, WORD PTR [myBytes] ; AX = 3412h
mov ax, WORD PTR [myBytes+2] ; AX = 7856h
mov eax, DWORD PTR myBytes ; EAX = 78563412h
```

* TYPE:

- Toán tử TYPE trả về kích thước, tính bằng byte, của một phần tử duy nhất của khai báo dữ liệu.

.data

```
var1 BYTE ?
var2 WORD ?
var3 DWORD ?
var4 QWORD ?
.code
mov eax, TYPE var1 ; 1
mov eax, TYPE var2 ; 2
mov eax, TYPE var3 ; 4
mov eax, TYPE var4 ; 8
```

* SIZEOF:

- Toán tử SIZEOF trả về một giá trị tương đương với việc nhân LENGTHOF với TYPE.

```
.data                                SIZEOF
Byte1 BYTE 10,20,30                  ; 3*1 = 3
Array1 WORD 30 DUP(?) , 0, 0         ; 32*2 = 64
Array2 WORD 5 DUP(3 DUP(?))          ; 15*2 = 30
Array3 DWORD 1, 2, 3, 4              ; 4*4 = 16
digitStr BYTE “12345678”, 0          ; 9*1 = 9
.code
mov ecx, SIZEOF array1 ; ECX = 64
```

* LENGTHOF:

- Toán tử LENGTHOF đếm số phần tử trong một khai báo dữ liệu.

```
.data                                LENGTHOF
Byte1 BYTE 10,20,30                  ; 3
Array1 WORD 30 DUP(?) , 0, 0         ; 30 + 2 = 32
Array2 WORD 5 DUP(3 DUP(?))          ; 5*3 = 15
```

```
Array3 DWORD 1, 2, 3, 4          ; 4
digitStr BYTE “12345678”, 0      ; 9
.code
mov ecx, LENGTHOF array1 ; ECX = 32
```

Câu 6:

Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: AND, OR, XOR, NOT, TEST, CMP

* AND:

- Phép toán AND thực hiện giữa toán hạng nguồn và toán hạng đích:

AND destination, source

- Các toán hạng có thể là 8 16 hoặc 32 bits và phải có kích thước giống nhau. Trong phép toán AND, quy tắc sau được áp dụng: Nếu cả hai bit bằng 1, bit kết quả là 1; mặt khác, nó là 0. Kết quả được ghi vào toán hạng đích.

- Lệnh AND cho phép xóa 1 hoặc nhiều bit trong toán hạng mà không làm ảnh hưởng đến các bit khác.

- Phép toán AND có thể được sử dụng kiểm tra một số là số chẵn hay không bằng cách AND bit thấp nhất của toán hạng với 1. Nếu như kết quả là 0, toán hạng là số chẵn:

```
MOV AX, value
AND AX, 1
```

- Cú pháp:

AND reg,reg

AND reg,mem

AND reg,imm

AND mem,reg

AND mem,imm

- Tác động vào cờ: Phép toán AND luôn luôn xóa các cờ OF và CF. Các cờ SF, ZF, PF được điều chỉnh phù hợp với kết quả được gán cho toán hạng đích.

```
mov al,10101110b
and al,11110110b ; result in AL = 1010011
```

* OR:

- Phép toán OR thực hiện giữa toán hạng nguồn và toán hạng đích:

OR destination, source

- Các toán hạng có thể là 8 16 hoặc 32 bits và phải có kích thước giống nhau. Đối với hai toán hạng, bit đầu ra là 1 khi có ít nhất một trong hai bit có giá trị đầu vào là 1.

- Lệnh OR được sử dụng để đặt 1 hoặc nhiều bit trong toán hạng mà không làm ảnh hưởng đến bit khác.

- Lệnh OR được sử dụng để chuyển đổi một byte nhị phân thể hiện của chữ số

thập phân thành chữ số thập phân ASCII tương đương của nó bằng cách thiết lập bit 4 và bit 5:

```
MOV AL, 9 ; AL = 00001001
OR AL, 00110000 ; AL = 00111001
```

- Cú pháp:

OR reg,reg

OR reg,mem

OR reg,imm

OR mem,reg

OR mem,imm

- Tác động vào cờ: Phép toán OR luôn xóa cờ CF và OF. Các cờ SF, OF và PF được điều chỉnh phù hợp với kết quả được gán cho toán hạng đích

```
mov al,10101110b
or al,11110110b ; result in AL = 11111111b.
```

* XOR:

- Phép toán XOR thực hiện thao tác OR chọn lọc giữa mỗi cặp bit cùng vị trí

trong hai toán hạng và lưu kết quả vào toán hạng đích:

XOR destination, source

- Các toán hạng có thể là 8 16 hoặc 32 bits và phải có kích thước giống nhau. Đối với mỗi bit cùng vị trí trong toán hạng, áp dụng: nếu cả hai bit giống nhau (cả 0 hoặc 1) thì kết quả là 0; ngược lại có một trong hai bit là 1 kết quả là 1.

- Kiểm tra cờ chẵn lẻ: là một hàm được thực hiện trên một số nhị phân bằng cách đếm số lượng bit 1 có trong số đó; nếu kết quả là chẵn cờ PF bật, ngược lại nếu kết quả là lẻ thì clear cờ PF.

- Cú pháp:

XOR reg,reg

XOR reg,mem

XOR reg,imm

XOR mem,reg

OR mem,imm

- Tác động vào cờ: Lệnh XOR luôn xóa cờ OF và CF. Các cờ SF, ZF và PF được điều chỉnh phụ thuộc vào kết quả được gán cho toán hạng đích.

* NOT:

- Phép toán NOT cho phép đảo ngược tất cả các bit trong toán hạng.

- Cú pháp:

NOT reg

NOT mem

- Tác động vào cờ:

```
mov al,11110000b
not al ; AL = 00001111b
```

* TEST:

- Lệnh TEST thực hiện tương tự như lệnh AND tuy nhiên các bit trong toán hạng không bị thay đổi.

- Lệnh TEST đặc biệt có ý nghĩa để kiểm tra các bit riêng lẻ trong toán hạng có được đặt hay không.

- Cú pháp:

TEST reg,reg

TEST reg,mem

TEST reg,imm

TEST mem,reg

- Tác động vào cờ: Lệnh TEST luôn xóa các cờ OF và CF; tác động đến các cờ SF, ZF và PF dựa trên giá trị được gán cho toán hạng đích nếu thực hiện sự thay đổi.

* CMP:

- Lệnh CMP thực hiện phép trừ ngụ ý toán hạng nguồn từ toán hạng đích, cả hai toán hạng đều không được sửa đổi:

CMP destination, source

-

- Cú pháp:

- Tác động vào cờ:

+ Khi thực hiện lệnh CMP các cờ OF, SF, ZF, CF, AF và PF thay đổi theo giá trị mà toán hạng đích sẽ có nếu phép trừ thực tế xảy ra.

+ Khi so sánh các toán hạng không dấu, các cờ ZF và CF chỉ ra các mối quan hệ sau đây giữa các toán hạng:

CMP Result	ZF	CF
Destination < source	0	1
Destination > source	0	0
Destination = source	1	0

+ Khi so sánh hai toán hạng có dấu, các cờ SF, ZF và OF chỉ ra các mối quan hệ sau đây giữa các toán hạng:

CMP Result	Flags

Destination < source	SF ≠ OF
Destination > source	SF = OF
Destination = source	ZF = 1

Câu 7 + 8:
Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau:

7. JE, JNE, JC, JZ, JNC, JP, LOOPZ, LOOPNZ, LOOPE, LOOPNE.

8. JG, JL, JNG, JA, JB, JNA, LOOPZ, LOOPNZ, LOOPE, LOOPNE

Lệnh	Mô tả	Trạng thái cờ	Ghi chú
JE	Nhảy nếu trái = phải		So sánh giá trị
JNE	Nhảy nếu trái != phải		So sánh giá trị
JC	Nhảy nếu cờ CF = 1	CF = 1	Giá trị cờ
JZ	Nhảy nếu nguồn = đích (=JE)	ZF = 1	Giá trị cờ
JNC	Nhảy nếu cờ CF = 0	CF = 0	Giá trị cờ
JP	Nhảy nếu cờ PF = 1	PF = 1	Giá trị cờ
LOOPZ	Lặp nếu đích = nguồn và ECX > 0	ZF = 1	
LOOPNZ	Lặp nếu đích != nguồn và ECX > 0	ZF = 0	
LOOPE	Lặp nếu trái = phải và ECX > 0	ZF = 1	
LOOPNE	Lặp nếu trái != phải và ECX > 0	ZF = 0	
JG	Nhảy nếu trái > phải		So sánh có dấu
JL	Nhảy nếu trái < phải		So sánh có dấu
JNG	Nhảy nếu trái <= phải		So sánh có dấu
JA	Nhảy nếu trái > phải		So sánh không dấu
JNA	Nhảy nếu trái <= phải		So sánh không dấu
JB	Nhảy nếu trái < phải		So sánh không dấu

Câu 9:
Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau:
SHL, SHR, SAL, SAR, ROL, ROR, RCL, RCR

* SHL:

- Shift Left - Dịch chuyển trái logic trên toán hạng đích, bit cao nhất được chuyển tới cờ Carry, bit thấp nhất = 0.

- Cấu trúc:

SHL reg,imm8

SHL mem,imm8

SHL reg,CL

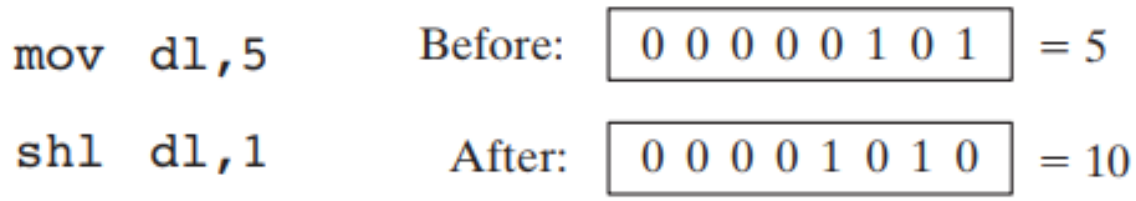
SHL mem,CL

```
mov  bl,8Fh                    ; BL = 10001111b
shl  bl,1                      ; CF = 1, BL = 00011110b
```

- Tác động vào cờ: Khi một toán hạng bị dịch chuyển sang trái nhiều lần, cờ Carry chứa bit cuối cùng được dịch ra khỏi bit quan trọng nhất (MSB) và OF.

```
mov  al,10000000b
shl  al,2                      ; CF = 0, AL = 00000000b
```

- Phép nhân theo chiều bit: SHL có thể thực hiện như phép nhân với lũy thừa 2. Chuyển bất kỳ toán hạng nào sang trái n bit sẽ nhân toán hạng với 2^n.



* SHR:

- Shift Right - Dịch chuyển phải logic trên toán hạng đích, bit cao nhất được chuyển tới cờ Carry, bit thấp nhất = 0.

- Cấu trúc:

SHR reg,imm8

SHR mem,imm8

SHR reg,CL

SHR mem,CL

```
mov  al,0D0h                   ; AL = 11010000b
shr  al,1                      ; AL = 01101000b, CF = 0
```

- Tác động vào cờ: 6 cờ trạng thái.

- Phép chia theo bit: SHR có thể thực hiện như phép chia với lũy thừa 2. Chuyển một số nguyên không dấu sang phải n bit: chia toán hạng cho 2^n

```
mov  dl,32
shr  dl,1
```

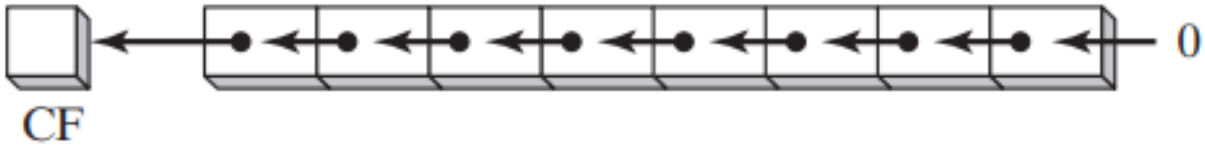
Before: 0 0 1 0 0 0 0 0 = 32

After: 0 0 0 1 0 0 0 0 = 16

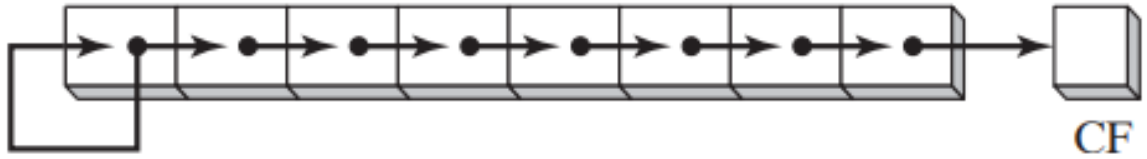
* SAL/SAR:

- Tương tự SHR/SHL chỉ khác ở chổ bit cuối cùng được đẩy ra ngoài sẽ được sao chép vào cờ CF.

SAL:



SAR:

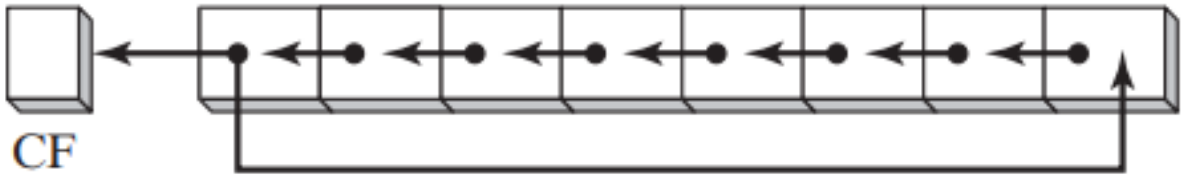


- Tác động vào cờ: 6 cờ trạng thái.

- Nếu dịch trái n bits thì nhân thêm 2^n, dịch phải n bits thì chia cho 2^n

* ROL:

- ROL (rotate left): di chuyển từng bit sang trái. Bit cao nhất được sao chép vào cờ Carry và vị trí bit thấp nhất



- Cấu trúc:

ROL reg,imm8

ROL mem,imm8

ROL reg,CL

ROL mem,CL

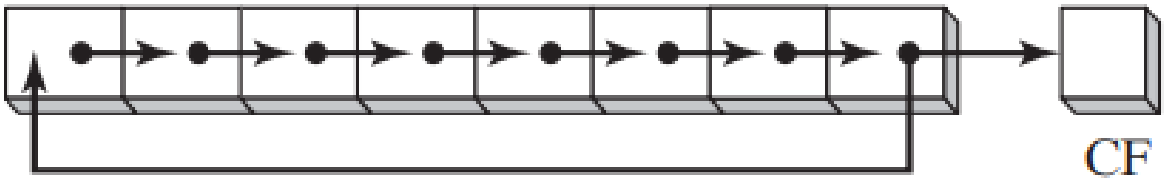
```
mov  al,40h                    ; AL = 01000000b
rol  al,1                      ; AL = 10000000b, CF = 0
rol  al,1                      ; AL = 00000001b, CF = 1
rol  al,1                      ; AL = 00000010b, CF = 0
```

- Tác động vào cờ: 6 cờ trạng thái.

- Trao đổi nhóm bit: Bạn có thể sử dụng ROL để trao đổi nửa trên (bit 4–7) và nửa dưới (bit 0–3) của một byte

* ROR:

- ROR (rotate right) dịch chuyển từng bit sang phải và sao chép bit thấp nhất vào cờ Carry và vị trí bit cao nhất



- Cấu trúc:

ROR reg,imm8

ROR mem,imm8

ROR reg,CL

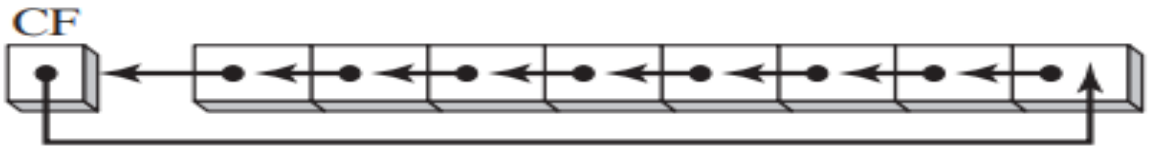
ROR mem,CL

```
mov  al,01h                    ; AL = 00000001b
ror  al,1                      ; AL = 10000000b, CF = 1
ror  al,1                      ; AL = 01000000b, CF = 0
```

- Tác động vào cờ: 6 cờ trạng thái.

* RCL:

- RCL (rotate carry left) dịch chuyển từng bit sang trái, sao chép cờ Carry sang LSB và sao chép MSB vào cờ Carry



- Cấu trúc:

RCL reg,imm8

RCL mem,imm8

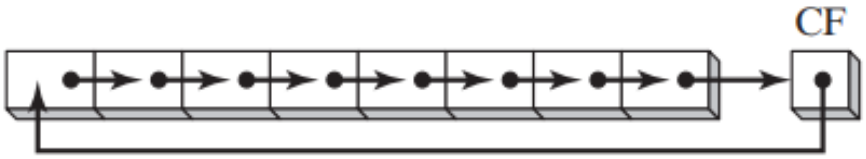
RCL reg,CL

RCL mem,CL

```
clc                 ; CF = 0
mov  bl,88h         ; CF,BL = 0 10001000b
rcl  bl,1           ; CF,BL = 1 00010000b
rcl  bl,1           ; CF,BL = 0 00100001b
```

* RCR:

- RCR (rotate carry right) dịch chuyển từng bit sang phải, sao chép cờ Carry vào MSB và sao chép LSB vào cờ Carry



- Cấu trúc:

RCR reg,imm8

RCR mem,imm8

RCR reg,CL

RCR mem,CL

```
stc                 ; CF = 1
mov  ah,10h         ; AH, CF = 00010000 1
rcr  ah,1           ; AH, CF = 10001000 0
```

Câu 10:
Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: MUL, IMUL, DIV, IDIV, CBW, CWD, CDQ

* MUL:

- Lệnh MUL (nhân không dấu) có ba phiên bản: phiên bản đầu tiên nhân toán hạng 8 bit bằng thanh ghi AL. Phiên bản thứ hai nhân toán hạng 16 bit bằng AX và phiên bản thứ ba nhân toán hạng 32 bit bằng EAX.

- Số nhân và cấp số nhân phải luôn có cùng kích thước và sản phẩm có kích thước gấp đôi kích thước của chúng.

Số nhân	Cấp số nhân	Kết quả
AL	reg/mem8	AX
AX	reg/mem16	DX:AX
EAX	reg/mem32	EDX:EAX

- Cấu trúc:

MUL reg/mem8

MUL reg/mem16

MUL reg/mem32

```
mov eax,12345h
mov ebx,1000h
mul ebx                 ; EDX:EAX = 0000000012345000h, CF = 0
```

* IMUL:

- Lệnh IMUL (nhân có dấu) thực hiện phép nhân số nguyên có dấu. Không giống như hướng dẫn MUL, IMUL giữ nguyên dấu hiệu của sản phẩm. Nó thực hiện điều

này bằng cách mở rộng bit cao nhất của nửa dưới của sản phẩm vào các bit trên của sản phẩm.

- Tập lệnh x86 hỗ trợ ba định dạng cho lệnh IMUL: một toán hạng, hai toán hạng và ba toán hạng.

- Ở định dạng một toán hạng, số nhân và cấp số nhân có cùng kích thước và tích gấp đôi kích thước của chúng

- Cấu trúc:

+ Dạng 1: nhân 1 toán hạng

*IMUL reg/mem8 ; AX = AL * reg/mem8*

*IMUL reg/mem16 ; DX:AX = AX * reg/mem16*

*IMUL reg/mem32 ; EDX:EAX = EAX * reg/mem32*

+ Dạng 2: nhân 2 toán hạng.

IMUL reg16,reg/mem16

IMUL reg16,imm8

IMUL reg16,imm16

IMUL reg32,reg/mem32

IMUL reg32,imm8

IMUL reg32,imm32

+ Dạng 3: nhân 3 toán hạng

IMUL reg16,reg/mem16,imm8

IMUL reg16,reg/mem16,imm16

IMUL reg32,reg/mem32,imm8

IMUL reg32,reg/mem32,imm32

* DIV:

- Lệnh DIV (chia không dấu) thực hiện phép chia số nguyên không dấu 8 bit, 16 bit và 32 bit. Thanh ghi đơn hoặc toán hạng bộ nhớ là ước số.

Số bị chia	Số chia	Thương	Phần dư
AX	reg/mem8	AL	AH
DX:AX	reg/mem16	AX	DX
EDX:EAX	reg/mem32	EAX	EDX

- Cấu trúc:

DIV reg/mem8

DIV reg/mem16

DIV reg/mem32

* IDIV:

- Lệnh IDIV (phép chia có dấu) thực hiện phép chia số nguyên có dấu, sử dụng các toán hạng giống như DIV. Trước khi thực hiện phép chia 8 bit, số bị chia (AX) phải được mở rộng hoàn toàn bằng dấu. Phần còn lại luôn có cùng dấu với số bị chia.

```
.data
wordVal SWORD -5000
.code
mov ax,wordVal          ; dividend, low
cwd                     ; extend AX into DX
mov bx,+256             ; divisor
idiv bx                 ; quotient AX = -19, rem DX = -136
```

* CBW: Lệnh CBW (chuyển đổi byte thành từ) mở rộng bit dấu của AL thành AH, giữ nguyên dấu của số.

```
.data
byteVal SBYTE -101      ; 9Bh
.code
mov al,byteVal          ; AL = 9Bh
cbw                     ; AX = FF9Bh
```

* CWD: Lệnh CWD (chuyển từ thành từ kép) mở rộng bit dấu của AX thành DX:

```
.data
wordVal SWORD -101      ; FF9Bh
.code
mov ax,wordVal          ; AX = FF9Bh
cwd                     ; DX:AX = FFFFFF9Bh
```

* CDQ: Lệnh CDQ (chuyển đổi từ kép thành bốn từ) mở rộng bit dấu của EAX thành EDX

```
.data
dwordVal SDWORD -101 ; FFFFFF9Bh
.code
mov eax,dwordVal
cdq                          ; EDX:EAX = FFFFFFFFFFFFFF9Bh
```

Câu 11:

Giải thích hoạt động, tác động vào các cờ, cho ví dụ minh hoạ của các lệnh sau: COORD, DUP, PROC, MACRO

* COORD: được xác định trong Windows API xác định X và Y tọa độ màn hình. Trường X có độ lệch bằng 0 so với phần đầu của cấu trúc và độ lệch của trường Y là 2

```
COORD STRUCT
 X WORD ? ; offset 00
 Y WORD ? ; offset 02
COORD ENDS
```

* DUP:

* PROC:

- Lệnh PROC khai báo một thủ tục với một danh sách tuỳ chọn của các tham số.

- Cú pháp:

```
label PROC parameter_list
```

- parameter_list là danh sách các tham số được tách nhau bằng dấu “,”. Mỗi tham số có cú pháp: paramName: type (type có thể là BYTE, SBYTE, WORD,.. hoặc nó có thể là con trỏ trỏ đến một trong những loại này)

- Các tham số có thể ở trên một dòng hoặc ở trên nhiều dòng, khi xuống dòng nếu muốn chưa kết thúc việc khai báo thì phải có dấu “,” đi sau.

* MACRO: