Chương 8 : Cấu trúc điều khiển và Vòng lặp

Mục tiêu

- **Biết cách mô phỏng cấu trúc điều khiển và vòng lặp như ở ngôn ngữ lập trình cấp cao.**
- **Nắm được các lệnh nhảy trong lập trình Assembly.**
- **Trên cơ sở đó, vận dụng để lập trình giải quyết 1 số bài toán.**

Nội dung

- ✓ Sự cần thiết của lệnh nhảy trong lập trình ASM.
- ✓ Lệnh JMP (Jump) : nhảy không điều kiện.
- ✓ Lệnh LOOP : cho phép lặp 1 công việc với 1 số lần nào đó.
- ✓ Các lệnh so sánh và luận lý.
- ✓ Lệnh lặp có điều kiện.
- ✓ Lệnh nhảy có điều kiện.
- ✓ Biểu diễn mô phỏng cấu trúc luận lý mức cao.
- ✓ Chương trình con.
- ✓ Một số chương trình minh họa.

Sự cần thiết của lệnh nhảy

- **Ở các chương trình viết bằng ngôn ngữ cấp cao thì việc nhảy (lệnh GoTo) là điều nên tránh nhưng ở lập trình hệ thống thì đây là việc cần thiết và là điểm mạnh của 1 chương trình viết bằng Assembly.**

- **Một lệnh nhảy ➔ CPU phải thực thi 1 đoạn lệnh ở 1 chỗ khác với nơi mà các lệnh đang được thực thi.**
- **Trong lập trình, có những nhóm phát biểu cần phải lặp đi lặp lại nhiều lần trong 1 điều kiện nào đó. Để đáp ứng điều kiện này ASM cung cấp 2 lệnh JMP và LOOP.**

Lệnh JMP （Jump）

- **Công dụng :Chuyển điều khiển không điều kiện**

Cú pháp : JMP đích

Nhảy gần (NEAR) : 1 tác vụ nhảy trong cùng 1 segment.

Nhảy xa (FAR) : 1 tác vụ nhảy sang segment khác.

Cać lệnh chuyển điều khiển

Chuyển điều khiển vô điều kiện

JMP [SORT | NEAR PTR | FAR PTR] DEST

Chuyển điều khiển có điều kiện

JConditional destination

Ex : JNZ nhãn đích ;

LỆNH LOOP

Công dụng : cho phép lặp 1 công việc với 1 số lần nào đó.
Mỗi lần lặp CX giảm đi 1 đơn vị. Vòng lặp chấm dứt khi CX =0.

Ex 1 : xuất ra màn hình 12 dòng gồm các ký tự A.

```
MOV CX, 12 * 80
MOV DL, 'A'
NEXT :
    MOV AH, 2
    INT 21H
LOOP NEXT
```

LOOP (tt)

Ex : có 1 Array A gồm 6 bytes, chép A sang array B – dùng SI và DI để lấy Offset

```
MOV SI, OFFSET A
MOV DI, OFFSET B
MOV CX, 6
MOVE_BYTE :
    MOV AL, [SI]
    MOV [DI], AL
    INC SI
    INC DI
LOOP MOVE_BYTE
A DB 10H,20H,30H,40H,50H,60H
B DB  6 DUP (?)
```

CÁC LỆNH LUẬN LÝ

Lưu ý về các toán tử LOGIC :
AND 2 Bit : kết quả là 1 khi và chỉ khi 2 bit là 1
OR 2 Bit : kết quả là 1 khi 2 Bit có bit là 1
XOR 2 Bit : kết quả là 1 chỉ khi 2 bit khác nhau
NOT 1 Bit : lấy đảo của Bit này

Lưu ý về thanh ghi cờ :

Cờ ZERO được lập khi tác vụ cho kết quả là 0.

Cờ CARRY được lập khi cộng kết quả bị tràn hay trừ phải mượn.

Cờ SIGN được lập khi bit dấu của kết quả là 1, tức kết quả là số âm.

Lệnh AND

Cú pháp : AND Destination , Source

Công dụng :

Lệnh này thực hiện phép AND giữa 2 toán hạng, kết quả cuối cùng chứa trong toán hạng đích.

Dùng để xóa các bit nhất định của toán hạng đích giữ nguyên các bit còn lại.

Muốn vậy ta dùng 1 mẫu bit gọi là mặt nạ bit (MASK), các bit mặt nạ được chọn để sao cho các bit tương ứng của đích được thay đổi như mong muốn.

Lệnh AND

Ex1 : xoá bit dấu của AL, giữ nguyên các bit còn lại :
dùng AND với 01111111b làm mặt nạ
AND AL, 7FH

Ex2 :
MOV AL, '5' ; Đổi mã ASCII của số

AND AL, 0FH ; thành số tương ứng.

Ex3 :
MOV DL, 'a' ; Đổi chữ thường thành chữ hoa.

AND DL, 0DFH ; thành số tương ứng.

LỆNH OR

Công dụng : dùng để bật lên 1 số bit và giữ nguyên các bit khác.

Cú pháp : OR destination, source

Ex1 :

OR AL , 10000001b ; bật bit cao nhất và bit thấp nhất trong thanh ghi AL lên 1

Ex 2:

MOV AL , 5 ; đổi 0..9 thành ký số

OR AL , 30h ; ASCII tương ứng.

Ex 3:

OR AL , AL ; kiểm tra một thanh ghi có = 0.

Nếu : cờ ZF được lập ➔ AL =0

cờ SIGN được lập ➔ AL <0

cờ ZR và cờ SIGN không được lập ➔ AL >0

Việc xoá 1 thanh ghi

Ta có 3 cách để xoá 1 thanh ghi :
C1: MOV AX , 0
C2 : SUB AX, AX
C3 : XOR AX, AX

Mã lệnh 1 dài 3 bytes

Mã lệnh 2 và 3 dài 2 bytes

➔ Lệnh 2,3 hiệu quả hơn

Tuy nhiên các thao tác giữa ô nhớ và ô nhớ là không hợp lệ nên khi cần xoá 1 ô nhớ ta phải dùng lệnh 1 .

LỆNH XOR

Công dụng : dùng để tạo đồ họa màu tốc độ cao.

Cú pháp : XOR destination, source

Ex : lật bit cao của AL 2 lần

MOV AL , 00111011b ;

XOR AL, 11111111b ; AL = 11000100b

XOR AL, 11111111b ; AL = 00111011b

LỆNH TEST

Cú pháp : TEST destination, source

Công dụng : dùng để khảo sát trị của từng bit hay nhóm bit.

Test thực hiện giống lệnh AND nhưng không làm thay đổi toán hạng đích.

Ex : kiểm tra bit 13 trong DX là 0 hay 1

```
TEST DX, 2000h
JZ BitIs0
BitIs1 :  bit 13 is 1
BitIs0 :  bit 13 is 0
```

Để kiểm tra 1 bit nào đó chỉ cần đặt bit 1 vào đúng vị trí bit cần kiểm tra và khảo sát cờ ZF. (nếu bit kiểm là 1 thì ZF sẽ xoá, ngược lại ZF được lập.

MINH HỌA LỆNH TEST

Ex : kiểm tra trạng thái máy in. Interrupt 17H trong BIOS sẽ kiểm tra trạng thái máy in, sau khi kiểm tra AL sẽ chứa trạng thái máy in. Khi bit 5 của AL là 1 thì máy in hết giấy.

```
MOV AH, 2
INT 17h
TEST AL ,  00100000b ; Test bit 5, nếu bit 5 = 1➔ máy in hết giấy.
```

Lệnh TEST cho phép test nhiều bit 1 lượt.

MINH HỌA LỆNH TEST(tt)

Ex :viết đoạn lệnh thực hiện lệnh nhảy đến nhãn A1 nếu AL chứa số chẵn.

TEST AL, 1 ; AL chứa số chẵn ?

JZ A1 ; nếu đúng nhảy đến A1.

Lệnh CMP

Cú pháp : CMP destination , source

Công dụng : so sánh toán hạng đích với toán hạng nguồn
bằng cách lấy toán hạng đích – toán hạng nguồn.

Hoạt động : dùng phép trừ nhưng không có toán hạng
đích nào bị thay đổi.

Các toán hạng của lệnh CMP không thể cùng là các ô nhớ.

lệnh CMP giống hệt lệnh SUB trừ việc toán hạng đích không thay đổi.

LỆNH NHẢY CÓ ĐIỀU KIỆN

Cú pháp : Jconditional destination

Công dụng : nhờ các lệnh nhảy có điều kiện, ta mới mô phỏng được các phát biểu có cấu trúc của ngôn ngữ cấp cao bằng Assembly.

Phạm vi

- Chỉ nhảy đến nhãn có khoảng cách từ -128 đến +127 byte so với vị trí hiện hành.
- Dùng các trạng thái cờ để quyết định có nhảy hay không?

LỆNH NHẢY CÓ ĐIỀU KIỆN

Hoạt động

- để thực hiện 1 lệnh nhảy CPU nhìn vào các thanh ghi cờ.
- nếu điều kiện của lệnh nhảy thỏa, CPU sẽ điều chỉnh IP trỏ đến nhãn đích các lệnh sau nhãn này sẽ được thực hiện.

```
.....
MOV  AH,  2
MOV  CX,  26
MOV  DL,  41H
PRINT_LOOP  :
   INT  21H
   INC  DL
   DEC  CX
JNZ  PRINT_LOOP
MOV  AX,  4C00H
INT  21H
```

LỆNH NHẢY DỰA TRÊN KẾT QUẢ SO SÁNH CÁC TOÁN HẠNG KHÔNG DẤU.

Thường dùng lệnh CMP Opt1 , Opt2 để xét điều kiện nhảy hoặc dựa trên các cờ.

JZ	Nhảy nếu kết quả so sánh = 0
JE	Nhảy nếu 2 toán hạng bằng nhau
JNZ	Nhảy nếu kết quả so sánh là khác nhau.
JNE	Nhảy nếu 2 toán hạng khác nhau.
JA	Nhảy nếu Opt1 > Opt2
JNBE	Nhảy nếu Opt1 <= Opt2
JAE	Nhảy nếu Opt1 >= Opt2

LỆNH NHẢY DỰA TRÊN KẾT QUẢ SO SÁNH CÁC TOÁN HẠNG KHÔNG DẤU (ctn) .

JNC	Nhảy nếu không có Carry.
JB	Nhảy nếu Opt1 < Opt2
JNAE	Nhảy nếu Not(Opt1 >= Opt2)
JC	Nhảy nếu có Carry
JBE	Nhảy nếu Opt1<=Opt2
JNA	Nhảy nếu Not (Opt1 > Opt2)

LỆNH NHẢY DỰA TRÊN KẾT QUẢ SO SÁNH CÁC TOÁN HẠNG CÓ DẤU .

JG	Nhảy nếu Opt1>Opt2
JNLE	Nhảy nếu Not(Opt1 <= Opt2)
JGE	Nhảy nếu Opt1>=Opt2
JNL	Nhảy nếu Not (Opt1 < Opt2)
JL	Nhảy nếu Opt1 < Opt2
JNGE	Nhảy nếu Not (Opt1 >= Opt2)
JLE	Nhảy nếu Opt1 <= Opt2
JNG	Nhảy nếu Not (Opt1 > Opt2)

LỆNH NHẢY DỰA TRÊN CÁC CỜ .

JCXZ	Nhảy nếu CX=0
JS	Nhảy nếu SF=1
JNS	Nhảy nếu SF =0
JO	Nhảy nếu đã tràn trị
JL	Nhảy nếu Opt1 < Opt2
JNGE	Nhảy nếu Not (Opt1 >= Opt2)
JLE	Nhảy nếu Opt1 <= Opt2
JNO	Nhảy nếu tràn trị
JP	Nhảy nếu parity chẳn
JNP	Nhảy nếu PF =0

CÁC VỊ DỤ MINH HỌA LỆNH NHẢY CÓ ĐK

Ex1 : tìm số lớn hơn trong 2 số
chứa trong thanh ghi AX và BX .
Kết quả để trong DX

```
MOV DX, AX          ; giả sử AX là số lớn hơn.
CMP DX, BX          ; IF AX >=BX then
JAE QUIT            ; nhảy đến QUIT
MOV DX, BX          ; ngược lại chép BX vào DX
QUIT :
  MOV AH,4CH
 INT 21H
.....
```

CÁC VÍ DỤ MINH HỌA LỆNH NHẢY CÓ ĐK

Ex1 : tìm số nhỏ nhất trong 3 số chứa trong thanh ghi AL BL và CL . Kết quả để trong biến SMALL

```
MOV    SMALL, AL
CMP  SMALL, BL
JBE  L1
MOV  SMALL, BL
L1  :
    CMP  SMALL, CL
  JBE    L2
MOV  SMALL, CL
L2  :  .  .  .
```

; giả sử AL nhỏ nhất

; nếu SMALL <= BL thì

Nhảy đến L1

; nếu SMALL <= CL thì

; Nhảy đến L2

; CL là số nhỏ nhất

Các lệnh dịch và quay bit

SHL (Shift Left) : dịch các bit của toán hạng đích sang trái

Cú pháp : SHL toán hạng đích ,1

Dịch 1 vị trí.

Cú pháp : SHL toán hạng đích ,CL

Dịch n vị trí trong đó CL chứa số bit cần dịch.

Hoạt động : một giá trị 0 sẽ được đưa vào vị trí bên phải nhất của toán hạng đích, còn bit msb của nó được đưa vào cờ CF

Các lệnh dịch và quay bit

Ex : DH chứa 8Ah, CL chứa 3.

SHL DH, CL ; 01010000b

? Cho biết kết quả của :

SHL 1111b, 3

MT thực hiện phép nhân bằng dịch trái

lệnh dịch phải SHR

Công dụng : dịch các bit của toán hạng đích sang bên phải.

Cú pháp : **SHR toán hạng đích , 1**

SHR toán hạng đích , CL ; dịch phải n bit trong đó CL chứa n

Hoạt động : 1 giá trị 0 sẽ được đưa vào bit msb của toán hạng đích, còn bit bên phải nhất sẽ được đưa vào cờ CF.

MT thực hiện phép chia bằng dịch phải

lệnh dịch phải SHR

```
Ex : shr 0100b, 1     ; 0010b = 2
```

Đối với các số lẻ, dịch phải sẽ chia đôi nó và làm tròn xuống số nguyên gần nhất.

```
Ex : shr 0101b, 1     ; 0010b = 2
```

Các phép nhân và chia tổng quát

Việc nhân và chia cho các số lũy thừa của 2 có thể thực hiện bằng lệnh dịch trái và dịch phải.

Để nhân và chia cho các số bất kỳ ta có thể kết hợp lệnh dịch và cộng.

Ex : nhân 2 số nguyên dương A và B bằng lệnh cộng và dịch bit.

Giả sử A = 111b và B = 1101b. Tính A*B

Các phép nhân và chia tổng quát

Thuật toán :

```
Tích = 0
Repeat
   If bit Lsb cuûa B baèng 1 Then
      tích = tích + A
   End If
   Dòch traùi A
   Dòch phaûi B
Until B =0
```

Các phép nhân và chia tổng quát

Giả sử A = 111b và B = 1101b. Tính A*B

-Vì bit lsb của B = 1

tích = tích +A = 111b

Dịch trái A : 1110b

Dịch phải B : 110b

Vì bit lsb của B = 0

Dịch trái A : 11100b

Dịch phải B : 11b

Vì bit lsb của B = 1

Tích =tích + A = 100011b

Dịch trái A : 111000b

Dịch phải B : 1b

Vì bit lsb của B =1

Tích = 100011b+111000b= 1011011b

Dịch trái A : 1110000b

Dịch phải B : 0b

Vì bit lsb của B = 0

Tích = 1011011b = 91d

Chương trình con

Có vai trò giống như chương trình con ở ngôn ngữ cấp cao.

ASM có 2 dạng chương trình con : dạng FAR và dạng NEAR.

Lệnh gọi CTC nằm cùng đoạn bộ nhớ với CTC được gọi

Lệnh gọi CTC nằm khác đoạn bộ nhớ với CTC được gọi

BIỂU DIỄN CẤU TRÚC LOGIC MỨC CAO

Dù Assembly không có phát biểu IF, ELSE, WHILE, REPEAT, UNTIL,FOR,CASE nhưng ta vẫn có thể tổ hợp các lệnh của Assembly để hiện thực cấu trúc logic của ngôn ngữ cấp cao.

Cấu trúc IF Đơn giản

Phát biểu IF sẽ kiểm tra 1 điều kiện và theo sau đó là 1 số các phát biểu được thực thi khi điều kiện kiểm tra có giá trị true.

Cấu trúc logic

```
IF (OP1=OP2)
 <STATEMENT1>
 <STATEMENT2>
ENDIF
```

HIỆN THỰC BẰNG ASM

```
CMP OP1,OP2
JNE  CONTINUE
 <STATEMENT1>
  <STATEMENT2>
CONTINUE : ....
```

Cấu trúc IF với OR

Phát biểu IF có kèm toán tử OR

Cấu trúc logic

```
IF (A1>OP1) OR
(A1>=OP2) OR
(A1=OP3) OR
(A1<OP4)
 <STATEMENT>
ENDIF
```

HIỆN THỰC BẰNG ASM

```
CMP A1,OP1
JG  EXCUTE
CMP A1,OP2
JGE  EXCUTE
CMP A1,OP3
JE EXCUTE
CMP A1,OP4
JL EXCUTE
JMP CONTINUE
EXCUTE : <STATEMENT>
CONTINUE : .....
```

Cấu trúc IF với AND

Phát biểu IF có kèm toán tử AND

Cấu trúc logic

```
IF (A1>OP1) AND
(A1>=OP2) AND
(A1=OP3) AND
(A1<OP4)
 <STATEMENT>
ENDIF
```

HIỆN THỰC BẰNG ASM

```
CMP A1,OP1
JNG  CONTINUE
CMP A1,OP2
JL  CONTINUE
CMP A1,OP3
JNE CONTINUE
CMP A1,OP4
JNL CONTINUE
<STATEMENT>
JMP CONTINUE
CONTINUE : ....
```

CHÚ Ý : khi điều kiện có toán tử AND, cách hay nhất là dùng nhảy với điều kiện ngược lại đến nhãn, bỏ qua phát biểu trong cấu trúc Logic.

VÒNG LẶP WHILE

Cấu trúc WHILE

Cấu trúc logic

DO WHILE (OP1<OP2)
<STATEMENT1>
<STATEMENT2>
ENDDO

HIỆN THỰC BẰNG ASM

```
DO_WHILE :
   CMP OP1, OP2
   JNL ENDDO
   <STATEMENT1>
   <STATEMENT2>
   JMP DO_WHILE
ENDDO : .....
```

VÒNG LẶP WHILE CÓ LỒNG IF

Cấu trúc WHILE có lồng IF

Cấu trúc logic

```
DO WHILE (OP1<OP2)
<STATEMENT>
IF (OP2=OP3) THEN
<STATEMENT2>
<STATEMENT3>
ENDIF
ENDDO
```

HIỆN THỰC BẰNG ASM

```
_WHILE :
   CMP OP1, OP2
   JNL WHILE_EXIT
   <STATEMENT1>
  CMP OP2,OP3 ; phần If
   JNE ELSE ; không thỏa If
  <STATEMENT2> ; thỏa If
  <STATEMENT3>
  JMP  ENDIF; thỏa If nên
                  bỏ qua Else
ELSE : <STATEMENT4>
ENDIF : JMP  _WHILE
WHILE_EXIT : .....
```

VÒNG LẶP REPEAT UNTIL

Cấu trúc REPEAT UNTIL

Cấu trúc logic

REPEAT

<STATEMENT1>

<STATEMENT2>

<STATEMENT3>

UNTIL (OP1=OP2) OR (OP1>OP3)

Bằng nhau thoát Repeat

HIỆN THỰC BẰNG ASM

REPEAT :

<STATEMENT1>

<STATEMENT2>

<STATEMENT3>

TESTOP12:

CMP OP1, OP2

JE ENDREPEAT

TESTOP13 :

CMP OP1, OP3

JNG **REPEAT**

ENDREPEAT : .....

Cấu trúc CASE

Cấu trúc logic

```
CASE INPUT OF
  ‘A’ : Proc_A
  ‘B’ : Proc_B
  ‘C’ : Proc_C
  ‘D’ : Proc_D
End ;
```

HIỆN THỰC BẰNG ASM

```
CASE : MOV AL, INPUT
CMP  AL, ‘A’
JNE TESTB
CALL PROC_A
JMP  ENDCASE
TESTB :
  CMP AL, ‘B’
 JNE TESTC
CALL PROC_B
JMP  ENDCASE
TESTC :
CMP AL, ‘C’
JNE TESTD
CALL  PROC_C
JMP  ENDCASE
TESTD : CMP  AL, ‘D’
JNE  ENDCASE
CALL  PROC_D
ENDCASE : ……..
```

LooKup Table

Rất hiệu quả khi xử lý phát biểu CASE là dùng bảng OFFSET chứa địa chỉ của nhãn hoặc của hàm sẽ nhảy đến tuỳ vào điều kiện.

Bảng Offset này được gọi Lookup Table rất hiệu quả khi dùng phát biểu Case có nhiều trị lựa chọn.

LooKup Table

```
Case_table  db  'A'      ; giá trị tìm kiếm
Dw  Proc_A               Địa chỉ các procedure
                         giả sử  ở địa chỉ 0120
Db 'B'
Dw  Proc_B               giả sử ở địa chỉ 0130
Db 'C'                   giả sử ở địa chỉ 0140
Dw  Proc_C
Db 'D'                   giả sử ở địa chỉ 0150
Dw  Proc_D
```

'A'	0120	'B'	0130	'C'	0140	'D'	0150

Cấu trúc lưu trữ của CaseTable như sau

LooKup Table

Case :

```
MOV AL, INPUT
MOV BX, OFFSET CASE_TABLE
MOV CX, 4 ; lặp 4 lần số entry của table
TEST :
  CMP  AL, [BX]  ; kiểm tra Input
 JNE  TESTAGAIN ; không thỏa kiểm tra tiếp
CALL WORD  PTR [BX+1]  ; gọi thủ tục tương ứng
JMP  ENDCASE
TESTAGAIN : ADD BX , 3 ; sang entry sau của CaseTable
LOOP TEST
ENDCASE : ..........
```

Chương trình con

Cấu trúc CTC :

```
TênCTC   PROC <Type>
      ; các lệnh
      RET
TênCTC  ENDP
```

CTC có thể gọi 1 CTC khác hoặc gọi chính nó.

CTC được gọi bằng lệnh CALL <TenCTC>.

CTC gần (near) là chương trình con nằm chung segment với nơi gọi nó.

CTC xa (far) là chương trình con không nằm chung segment với nơi gọi nó.

Kỹ thuật lập trình

- Hãy tổ chức chương trình ➔ các chương trình con ➔ đơn giản hoá cấu trúc luận lý của CT làm cho CT dễ đọc, dễ hiểu , dễ kiểm tra sai sót..

- Đầu CTC hãy cất trị thanh ghi vào Stack bằng lệnh PUSH để lưu trạng thái hiện hành.

- Sau khi hoàn tất công việc của CTC nên phục hồi lại trị các thanh ghi lúc trước đã Push bằng lệnh POP .
- Nhớ trình tự là ngược nhau để trị của thanh ghi nào trả cho thanh ghi nấy.

- Đừng tối ưu quá CT vì có thể làm cho CT kém thông minh, khó đọc.

Kỹ thuật lập trình (tt)

- Cố gắng tổ chức chương trình cho tốt ➔ phải thiết kế được các bước chương trình sẽ phải thực hiện.

- Kinh nghiệm : khi vấn đề càng lớn thì càng phải tổ chức logic chương trình càng chặt chẽ.

- Bằng sự tổ hợp của lệnh nhảy ta hoàn toàn có thể mô phỏng cấu trúc điều khiển và vòng lặp.

SUMMARY

- ✓ Có thể mô phỏng cấu trúc logic như ngôn ngữ cấp cao trong Assembly bằng lệnh JMP và LOOP.
- ✓ các lệnh nhảy : có điều kiện và vô điều kiện.
- ✓ Khi gặp lệnh nhảy, CPU sẽ quyết định nhảy hay không bằng cách dựa vào giá trị thanh ghi cờ.
- ✓ các lệnh luận lý dùng để làm điều kiện nhảy là AND, OR, XOR, CMP . . .
- ✓Bất cứ khi nào có thể, hãy tổ chức chương trình thành các chương trình con ➔ đơn giản được cấu trúc luận lý của chương trình.

Câu hỏi

1. Giả sử DI = 2000H, [DS:2000] = 0200H. Cho biết địa chỉ ô nhớ toán hạng nguồn và kết quả lưu trong toán hạng đích khi thực hiện lệnh MOV DI, [DI]
2. Giả sử SI = 1500H, DI=2000H, [DS:2000]=0150H . Cho biết địa chỉ ô nhớ toán hạng nguồn và kết quả lưu trong toán hạng đích sau khi thực hiện lệnh ADD AX, [DI]
3. Có khai báo A DB 1,2,3
   Cho biết trị của toán hạng đích sau khi thi hành lệnh MOV AH, BYTE PTR A.
4. Có khai báo B DB 4,5,6
   Cho biết trị của toán hạng đích sau khi thi hành lệnh MOV AX, WORD PTR B.

Bài tập LẬP TRÌNH

Bài 1 : Có vùng nhớ VAR1 dài 200 bytes trong đoạn được chỉ bởi DS.

Viết chương trình đếm số chữ 'S' trong vùng nhớ này.

Bài 2 : Có vùng nhớ VAR2 dài 1000 bytes. Viết chương trình chuyển đổi các chữ thường trong vùng nhớ này thành các ký tự hoa, các ký tự còn lại không đổi.

Bài 3 : Viết chương trình nhập 2 số nhỏ hơn 10.

In ra tổng của 2 số đó.

Bài tập LẬP TRÌNH

Bài 4 : Viết chương trình nhập 2 số bất kỳ.

In ra tổng và tích của 2 số đó. Chương trình có dạng sau :

Nhập số 1 : 12

Nhập số 2 : 28

Tổng là : 40

Tích là : 336

Bài 5 : Viết chương trình nhập 1 ký tự. Hiển thị 5 ký tự kế tiếp trong bộ mã ASCII.

Ex : nhập ký tự : a

5 ký tự kế tiếp : b c d e f

Bài tập LẬP TRÌNH

Bài 6 : Viết chương trình nhập 1 ký tự. Hiển thị 5 ký tự đứng trước trong bộ mã ASCII.

Ex : nhập ký tự : f

5 ký tự kế tiếp : a b c d e

Bài 7 : Viết chương trình nhập 1 chuỗi ký tự.

In chuỗi đã nhập theo thứ tự ngược.

Ex : nhập ký tự : abcdef

5 ký tự kế tiếp : fedcba