

คำสั่งโอนย้ายข้อมูล

- คำสั่ง MOV
 - การโอนย้ายข้อมูลระหว่างรีจิสเตอร์กับรีจิสเตอร์
 - การโอนย้ายข้อมูลกับหน่วยความจำ
 - การกำหนดค่าคงที่ให้กับหน่วยความจำ
- เครื่องมือในการพัฒนาโปรแกรมภาษาแอสเซมบลี : โปรแกรม DEBUG
 - คำสั่งทั่วไป
 - การทดลองโปรแกรมภาษาแอสเซมบลีด้วยโปรแกรม DEBUG
 - ตัวอย่างการทดลองโปรแกรมภาษาแอสเซมบลี

คำสั่ง MOV

- รูปแบบ MOV ปลายทาง,ต้นทาง

MOV reg, reg

MOV reg, mem

MOV mem, reg

MOV reg, imm

MOV mem, imm



reg : register

mem : memory

imm : immediate (ค่าคงที่)

- ข้อจำกัดของคำสั่ง MOV

- โอเปอเรนด์ทั้งสองตัวต้องมีขนาดเท่ากัน
- ไม่สามารถคัดลอก
 - ค่าคงที่ (immediate) ไปยังเซกเมนต์รีจิสเตอร์ได้โดยตรง
 - ข้อมูลจากหน่วยความจำไปยังหน่วยความจำได้โดยตรง
- ในการคัดลอกค่าคงที่ไปยังหน่วยความจำต้องระบุขนาดของหน่วยความจำด้วย


ตัวอย่างการใช้คำสั่ง MOV

MOV	AX, 100h	กำหนดค่า 100h ให้กับ AX จากนั้น คัดลอกไปให้ BX และ DX
MOV	BX, AX	
MOV	DX, BX	
MOV	AX, 1234h	กำหนดค่า 1234h ให้กับ AX และค่า 5678h ให้กับ DX จากนั้นคัดลอกค่า ใน DL ไปให้ AL และ DH ไปให้ BH
MOV	DX, 5678h	
MOV	AL, DL	
MOV	BH, DH	
MOV	AX, 1000h	กำหนดค่า 1000h ให้กับ AX จากนั้น คัดลอกข้อมูลจาก AX ไปยังหน่วย ความจำตำแหน่งที่ DS:100h และคัด ลอกข้อมูลกลับมายัง BX
MOV	[100h], AX	
MOV	BX, [100h]	
MOV	BYTE PTR [200h], 10h	กำหนดค่า 10h แบบไบต์ ให้กับหน่วยความจำที่ DS:200h และแบบเวิร์ดที่ DS:300h
MOV	WORD PTR [300h], 10h	
MOV	AX, 2300h	กำหนดค่า 2300h ให้กับ DS โดยผ่าน ทาง AX
MOV	DS, AX	

การโอนย้ายข้อมูลระหว่างรีจิสเตอร์

- การโอนย้ายข้อมูลระหว่างรีจิสเตอร์สามารถทำได้ถ้าขนาดของรีจิสเตอร์ทั้งคู่เท่ากัน
- คู่รีจิสเตอร์ 16 บิต กับ 8 บิต

MOV AX, 1000h



AX	AH	AL
1000h	10h	00h

MOV AL, 3Ah



AX	AH	AL
103Ah	10h	3Ah

MOV AH, AL



AX	AH	AL
3A3Ah	3Ah	3Ah

MOV AX, 234h



AX	AH	AL
234h	02h	34h

การโอนย้ายข้อมูลกับหน่วยความจำ

- การระบุตำแหน่งในหน่วยความจำ
 - โดยทั่วไป ในการระบุตำแหน่งในหน่วยความจำเรา จะระบุเฉพาะออฟเซตเท่านั้น โดยออฟเซตที่ระบุจะถูกนำไปประกอบกับค่าในเซกเมนต์ รีจิสเตอร์ที่เหมาะสม เช่นในการอ้างถึงข้อมูลออฟเซตจะถูกนำไปประกอบกับ DS เป็นต้น

- ตัวอย่าง

MOV	AX, 6789h	DS:100h	
MOV	DX, 1234h	DS:101h	
MOV	[100h], AX	DS:102h	
MOV	[102h], DX		
MOV	[104h], AH		
MOV	[105h], DL		
MOV	BX, [104h]		
MOV	CX, [103h]		
MOV	[106h], CL		

การโอนย้ายข้อมูลกับหน่วยความจำ

MOV [100h],AX

67 89

MOV [102h],DX

12 34

MOV [104h],AH

67

MOV [105h],DL

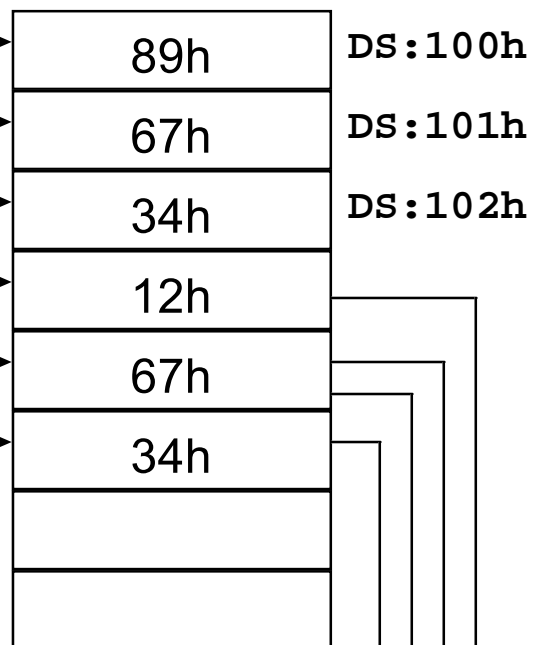
34

MOV BX,[104h]

34 67

MOV CX,[103h]

67 12



การโอนย้ายข้อมูลกับหน่วยความจำ

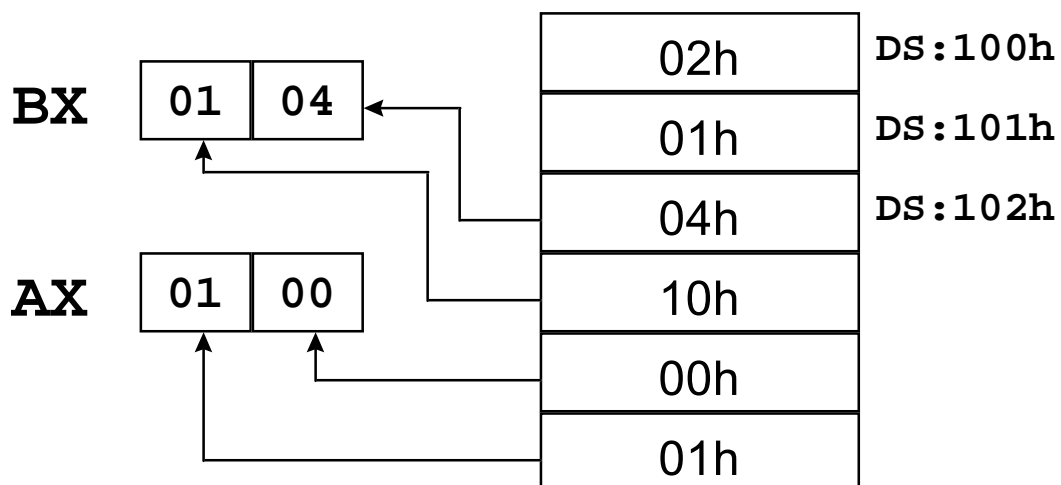
- ข้อสังเกตในการจัดเรียงลำดับไบนารีของข้อมูล
 - สังเกตว่าในการเก็บค่าในหน่วยความจำเมื่อเราเก็บค่าเป็น 16 บิต การเรียงไบนารีในหน่วยความจำจะเก็บค่าในไบนารีที่มีนัยสำคัญสูงไว้ในไบนารีที่มีแอดเดรสสูงกว่า และไบนารีที่มีนัยสำคัญต่ำไว้ในแอดเดรสที่มีแอดเดรสต่ำกว่า
 - เราเรียกว่าเป็นการเรียงแบบ **little endian** การเรียงข้อมูลแบบนี้ใช้ในหน่วยประมวลผลตระกูล Intel
 - ในหน่วยประมวลผลตระกูลอื่นเช่น SPARC หรือ MIPS จะเรียงไบนารีกลับกันการเรียงอีกแบบนี้เราเรียกว่าการเรียงแบบ **big endian**

การโอนย้ายข้อมูลกับหน่วยความจำ

- เราสามารถระบุออฟเซตของหน่วยความจำทางอ้อมได้โดยผ่านทางรีจิสเตอร์ BX

```
MOV AX,102h
MOV BX,100h
MOV CX,4004h
MOV DX,1201h
MOV [BX],AX
MOV [BX+2],CX
MOV [BX+3],DX
MOV [BX+4],BX
MOV BX,[102h]
MOV AX,[BX]
```

AX = ?



การกำหนดค่าให้กับหน่วยความจำ

- ลองสังเกตคำสั่งต่อไปนี้

MOV [100h],10h

- การคัดลอกค่าไปยังหน่วยความจำจะเป็นแบบ
 - 16 บิต (คัดลอก 0010h) หรือ
 - 8 บิต (คัดลอก 10h)
- ในการเขียนค่าคงที่ลงในหน่วยความจำเราจะต้องระบุขนาดของหน่วยความจำด้วย

MOV WORD PTR [100h],10h

MOV BYTE PTR [100h],10h

- สังเกตว่าความกำกวมนี้ไม่เกิดในกรณีของ

MOV [100h],AX

การใช้โปรแกรม DEBUG

ในการทดลองโปรแกรมภาษาแอสเซมบลี

- คำสั่งทั่วไป

- คำสั่ง ? : แสดงรายการคำสั่ง
- คำสั่ง R (register) : จัดการกับรีจิสเตอร์

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0100  NV UP EI PL NZ NA PO NC
12AF:0100  5F                POP      DI
-R
CX 0000
:100
-R
AX=0000  BX=0000  CX=0100  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0100  NV UP EI PL NZ NA PO NC
12AF:0100  5F                POP      DI
-
```

- คำสั่ง D (dump) : แสดงค่าในหน่วยความจำ

```
-D100
12AF:0100  C7 06 16 00 6E 30 26 C7-06 18 00 C7 40 26 8B 06  ....n0&.....@&...
12AF:0110  08 00 26 03 06 0C 00 26-3B 06 06 00 34 00 9E 12  ..&.....&i...4...
12AF:0170  C2 02 00 90 C8 04 00 00-57 56 8B 7E 0A 57 E8 57  ....WV.~.W.W
-
```

การใช้โปรแกรม DEBUG :

คำสั่งสำหรับการแปลโปรแกรม

- คำสั่ง A (assemble) : สั่งให้โปรแกรม DEBUG แปลโปรแกรกลงในแอดเดรสที่ระบุ

```
-A100
12AF:0100 mov ax,10
12AF:0103 mov bx,20
12AF:0106 mov [200],ax
12AF:0109 mov [202],bx
12AF:010D mov bx,204
12AF:0110 mov cx,1234
12AF:0113 mov [bx],cx
12AF:0115 int 20
12AF:0117
-
```

ข้อสังเกต ตัวเลขต่าง ๆ ในโปรแกรม DEBUG จะเป็นเลขฐาน 16 ทั้งหมด ไม่ต้องพิมพ์ h ระบุหลังตัวเลข

- คำสั่ง U (unassemble) : สั่งให้โปรแกรม DEBUG แสดงโปรแกรมที่อยู่ในแอดเดรสที่ระบุ

```
-U100
12AF:0100 B81000      MOV     AX,0010
12AF:0103 BB2000      MOV     BX,0020
12AF:0106 A30002      MOV     [0200],AX
12AF:0109 891E0202    MOV     [0202],BX
12AF:010D BB0402      MOV     BX,0204
12AF:0110 B93412      MOV     CX,1234
12AF:0113 890F        MOV     [BX],CX
12AF:0115 CD20        INT     20
      ...
12AF:011F 12FE        ADC     BH,DH
-
```

การใช้โปรแกรม DEBUG :

คำสั่งสำหรับการติดตามการทำงาน

- คำสั่ง G (go) : คำสั่งเริ่มการทำงาน
 - โปรแกรมจะเริ่มทำงานที่แอดเดรส CS:IP จนกระทั่งจบโปรแกรม (มีคำสั่งสั่งให้จบการทำงาน เช่นในตัวอย่างมีคำสั่ง INT 20h)
- คำสั่ง T (trace) : คำสั่งตามรอยการทำงาน
 - โปรแกรมจะทำงานไป 1 คำสั่งแล้วจะกลับมาที่ DEBUG เพื่อรับคำสั่งอื่น ๆ ต่อไป
- คำสั่ง P (proceed) : คำสั่งให้ทำงานจนถึงบรรทัดถัดไป
 - โปรแกรมจะทำงานไปจนกระทั่งถึงบรรทัดถัดไป แล้วจะกลับมาที่ DEBUG เพื่อรับคำสั่งอื่น ๆ ต่อไป
 - ใช้ในกรณีมีการเรียกโปรแกรมย่อยหรือมีการเรียกใช้บริการของระบบ

หมายเหตุ เราควรพิจารณาค่าใน CS และ IP ก่อนสั่งให้โปรแกรมทำงาน เราสามารถกำหนดค่าให้กับรีจิสเตอร์ทั้งสองได้โดยคำสั่ง RCS และ RIP

ตัวอย่างการทดลองโปรแกรม

- โปรแกรมตัวอย่าง

```
MOV    AX,5678h
MOV    BX,204h
MOV    CX,1234h
MOV    [200h],CX
MOV    [202h],AH
MOV    [203h],AL
MOV    [BX],AX
MOV    DX,[202h]
MOV    [204h],10h
```

ตัวอย่างการทดลองโปรแกรม (1)

ป้อนโปรแกรม

-A100

14FF:0100 *mov ax,5678*

14FF:0103 *mov bx,204*

14FF:0106 *mov cx,1234*

14FF:0109 *mov [200],cx*

14FF:010D *mov [202],ah*

14FF:0111 *mov [203],al*

14FF:0114 *mov [bx],ax*

14FF:0116 *mov dx,[202]*

14FF:011A *mov [204],10*

^ Error

14FF:011A *mov word ptr [204],10*

14FF:0120

-

-U

14FF:0100 B87856 MOV AX,5678

14FF:0103 BB0402 MOV BX,0204

14FF:0106 B93412 MOV CX,1234

14FF:0109 890E0002 MOV [0200],CX

14FF:010D 88260202 MOV [0202],AH

14FF:0111 A20302 MOV [0203],AL

14FF:0114 8907 MOV [BX],AX

14FF:0116 8B160202 MOV DX,[0202]

14FF:011A C70604021000 MOV WORD PTR [0204],0010

-

ใช้คำสั่ง A (assemble) ป้อน
โปรแกรม ถ้าโปรแกรมมีข้อผิดพลาด
DEBUG จะแจ้งให้ผู้ใช้
ทราบ

เรียกดูโปรแกรมที่ป้อน
ไปอีกครั้ง

ตัวอย่างการทดลองโปรแกรม (2)

สั่งให้โปรแกรมทำงาน

-A120

14FF:0120 int 20

14FF:0122

-U100

14FF:0100 B87856 MOV AX,5678

14FF:0103 BB0402 MOV BX,0204

...

14FF:011A C70604021000 MOV WORD PTR [0204],0010

-U11A

14FF:011A C70604021000 MOV WORD PTR [0204],0010

14FF:0120 CD20 INT 20

...

-

-G

Program terminated normally

-D200

14FF:0200 34 12 56 78 10 00 DA EB-04 9D F8 EB 02 9D F9 89 4.Vx.....

14FF:0210 36 8A DB 5E 5F 5A 59 C3-53 51 52 57 56 9C E8 6E 6..^_ZY.SQRWV..n

14FF:0220 00 83 3E 7A DB 40 7D 57-8A F7 8B 1E 7A DB FF 06 ..>z.@}W....z...

14FF:0230 7A DB B8 BA D8 E8 95 00-C7 47 07 00 00 F6 C6 01 z.....G.....

14FF:0240 74 03 89 6F 07 89 4F 05-88 77 02 8B 36 84 DB 89 t.o.o..O..w..6...

14FF:0250 37 03 36 5C D8 2B F7 89-77 03 8B 36 8A DB 89 77 7.6\..+..w..6...w

14FF:0260 09 8B F7 8B 3E 84 DB 03-F9 3B 3E 80 DB 7D 15 2B>....i>...}.+

14FF:0270 F9 FC F3 A4 B0 00 AA 89-3E 84 DB 9D F8 EB 0A B8>.....

-R

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=14FF ES=14FF SS=14FF CS=14FF IP=0100 NV UP EI PL NZ NA PO NC

14FF:0100 B87856 MOV AX,5678

-

ถ้าเราจะใช้คำสั่ง G (g) เพื่อให้โปรแกรมทำงาน เราจะต้องใส่คำสั่งที่สั่งให้โปรแกรมจบการทำงาน (INT 20h) ลงไปท้ายโปรแกรมด้วย

สั่งให้โปรแกรมทำงาน และให้แสดงค่าในหน่วยความจำ

พิจารณาค่าในรีจิสเตอร์

ตัวอย่างการทดลองโปรแกรม (3)

ให้โปรแกรมทำงานทีละบรรทัด

- **T**

```
AX=5678 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0103 NV UP EI PL NZ NA PO NC
14FF:0103 B0402          MOV     BX,0204
```

- **T**

```
AX=5678 BX=0204 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0106 NV UP EI PL NZ NA PO NC
14FF:0106 B93412          MOV     CX,1234
```

- **T**

```
AX=5678 BX=0204 CX=1234 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0109 NV UP EI PL NZ NA PO NC
14FF:0109 890E0002        MOV     [0200],CX
DS:0200=1234
```

- **T**

```
AX=5678 BX=0204 CX=1234 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=010D NV UP EI PL NZ NA PO NC
14FF:010D 88260202        MOV     [0202],AH
DS:0202=56
```

- **T**

```
AX=5678 BX=0204 CX=1234 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0111 NV UP EI PL NZ NA PO NC
14FF:0111 A20302          MOV     [0203],AL
DS:0203=78
```

- **T**

...

- **T**

```
AX=5678 BX=0204 CX=1234 DX=7856 SP=FFEE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0120 NV UP EI PL NZ NA PO NC
14FF:0120 CD20           INT     20
```

- **P**

Program terminated normally

-

ใช้คำสั่ง P เพราะเป็นเรียกใช้บริการจากระบบ

ตัวอย่างการทดลองโปรแกรม (4)

การใช้คำสั่ง T ในการติดตามคำสั่ง INT 20h

-T

AX=5678 BX=0204 CX=1234 DX=7856 SP=FFFE BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=14FF IP=0120 NV UP DI PL NZ NA PO NC
14FF:0120 CD20 INT 20

-T

AX=5678 BX=0204 CX=1234 DX=7856 SP=FFE2 BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=00C9 IP=0FA8 NV UP DI PL NZ NA PO NC
00C9:0FA8 90 NOP

-T

AX=5678 BX=0204 CX=1234 DX=7856 SP=FFE2 BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=00C9 IP=0FA9 NV UP DI PL NZ NA PO NC
00C9:0FA9 90 NOP

-T

AX=5678 BX=0204 CX=1234 DX=7856 SP=FFE2 BP=0000 SI=0000 DI=0000
DS=14FF ES=14FF SS=14FF CS=00C9 IP=0FAA NV UP DI PL NZ NA PO NC
00C9:0FAA E8DB00 CALL 1088

-G

Program terminated normally

-

สังเกตว่าจะมีการกระโดดไป
ทำงานที่ตำแหน่งอื่น

เราสามารถใช้คำสั่ง G เพื่อให้
โปรแกรมทำงานจนจบได้

การบ้าน

- จงทดลองโปรแกรมต่อไปนี้ใน DEBUG แล้วบันทึกค่าของรีจิสเตอร์ AX BX CX และ DX รวมทั้งค่าในหน่วยความจำตั้งแต่ตำแหน่งที่ 200h ถึง 210h หลังการทำงานของโปรแกรม

```
MOV    AX,1234h
MOV    BX,202h
MOV    CX,9999h
MOV    [BX],CX
MOV    [BX+2],AX
MOV    DX,[BX+1]
MOV    AX,A12h
MOV    [BX+5],AX
MOV    BYTE PTR [BX+7],2h
MOV    BX,[BX+6]
MOV    [BX],AL
```