บทที่ 5

การกำหนดและเคลื่อนย้ายข้อมูล

DEFINING AND MOVING DATA

วัตถุประสงค์

หลังจากที่ท่านศึกษาบทนี้ท่านจะมีความเข้าใจดังต่อไปนี้

- วิธีการกำหนดข้อมูลโปรแกรมภาษาแอสแซมบลี
- การกำหนดพื้นที่สำหรับการเคลื่อนย้ายข้อมูล
- การกำหนดค่าคงที่
- การกำหนดข้อมูลสตริง

CT 215 85

บทที่ 7

บทที่ 5
การกำหนดและเคลื่อนย้ายข้อมูล
DEFINING AND MOVING DATA

บทนำ

การศึกษาบทนี้ จะเป็นการกำหนดรายการข้อมูลและการเคลื่อนย้ายข้อมูล การกำหนดรายการข้อมูลอาจจะเป็นค่าคงที่หรือตัวแปร หรืออาจจะกำหนดในเทอมของความยาวที่แตกต่างกันเช่น ไบท์ เวิร์ด และ 2 เวิร์ด และศึกษาคำสั่งเทียม EQU ซึ่งจะนำมาใช้กำหนดค่าคงที่ของชื่อข้อมูล (NAME) อธิบายรายละเอียดของคำสั่ง MOV ซึ่งเป็นคำสั่งที่ถูกใช้บ่อยๆ ในการเคลื่อนย้ายข้อมูลระหว่างรีจีสเตอร์กับรีจีสเตอร์ หรือรีจีสเตอร์กับหน่วยความจำ ท่านสามารถกำหนดข้อมูลให้กับคำสั่ง สุดท้ายจะแนะนำพื้นฐานรูปแบบการอ้างแอดเดรสของภาษาแอสแซมบลี้

คำสั่งเทียมกำหนดข้อมูล (DATA DEFINITION DIRECTIVES)

จุดประสงค์ของ DATA SEGMENT ในโปรแกรม EXE สำหรับกำหนดค่าคงที่ Work areas และ input/output แอสแซมเบลอร์จะกำหนดรายการต่างๆตามคำสั่งเทียมที่กำหนดไว้ ตัวอย่างการใช้ DB และ DW รายการข้อมูลอาจจะไม่กำหนดค่า หรืออาจจะกำหนดค่าคงที่ ตัวอักขระ หรือค่าตัวเลข มีรูปแบบดังต่อไปนี้

[NAME] Dn EXPRESSION

NAME ถ้าโปรแกรมอ้างถึงรายการข้อมูลคำว่า NAME หมายถึงชื่อของข้อมูล
DIRECTIVE คำสั่งเทียมที่ใช้กำหนดรายการข้อมูลคือ DB , DW , DD , DF , DQ ,DT
EXPRESSION กฎเกณฑ์ในส่วนของ โอเปอร์แรนด์ อาจจะเป็น ? เพื่อเป็นตัวชี้ข้อมูลที่ไม่ได้กำหนด

```
FLD1    DB    ?
FLD2    DB    25
```

การกำหนดค่าคงที่จำนวนมากอาจใช้เครื่องหมาย , คั่นระหว่างข้อมูลดังนี้

```
FLD3        DB          11,12,13,14,15,16,.....
```

แอสแซมเบลอร์อาจกำหนดค่าคงที่ในไบท์ต่างๆตามรายชื่อของข้อมูลที่อ้างถึงใน FLD3 ค่าคงที่ไบท์แรกคือ 11 และถ้าเป็น FLD3 + 1 คือ 12 ดังตัวอย่าง

```
MOV         AL , FLD3 + 3
```

เป็นการโหลดค่า 14 ไปเก็บไว้ในรีจีสเตอร์ AL กฎเกณฑ์ในการทำสำเนาข้อมูลมีดังต่อไปนี้

```
[NAME]    Dn    repeat-count  DUP(expression).....
```

ตัวอย่าง การใช้คำสั่งกำหนดข้อมูล

```
DW    10 DUP(?)           ;TEN WORDS UNINITIALIZED
DB    5DUP(14)            ;FIVE BYTES CONTAINING HEX 0E0E0E0E0E0E
DB    3DUP(4DUP(8))       ;TWELVES 8s
```

ตัวอักขระสตริง (CHARACTER STRINGS)

ตัวอักขระสตริงที่ใช้ในข้อมูล ที่แสดงชื่อของคนสามารถกำหนดได้ดังนี้

```
DB    'CHARACTER STRING'
```

ค่าคงที่ตัวเลข (NUMERIC CONSTANTS)

ตัวเลขค่าคงที่ที่ใช้ในการคำนวนทางคณิตศาสตร์ และสำหรับการกำหนดแอดเดรสของหน่วยความจำ ค่าคงที่ตัวเลขจะไม่ต้องอยู่ภายใต้กำกับของ Single Quotes แอสแซมเบลอร์จะทำการเปลี่ยนค่าคงที่ที่เป็นตัวเลขเป็นเลขฐานสิบหกและเก็บค่าของไบท์ข้อมูลในลักษณะสลับไบท์ขวาไปซ้าย

DECIMAL รูปแบบของเลขฐานสิบคือเลข 0 - 9 ถ้ากำหนดในโปรแกรมจะต้องจะต้องตามด้วยอักษร D เช่น 125D หรือ 125 แอสแซมเบลอร์จะเปลี่ยนค่า 125 เป็นเลขฐานสิบหกดังนี้ 7DH

HEXADECIMAL รูปแบบเลขฐานสิบหกคือ 0 -9 และ A - F ค่าของตัวเลขที่ใช้ในโปรแกรมจะต้องตามด้วยตัวอักษร H เช่น 12H , 4A6H แต่ถ้าตัวเลขฐานสิบหกที่ขึ้นต้นด้วยตัวอักษรก็ต้องมีเลข 0 นำหน้าตัวเลขนั้น เช่น 0A4H , 0D123H

BINARY รูปแบบของเลขฐานสอง จะมีตัวเลขอยู่ 2 ตัวคือ 0 กับ 1 และตามด้วยตัวอักษร B ปกติในการเขียนโปรแกรมภาษาแอสแซมบลี้เลขฐานสองจะใช้ในการเคลียบิตสำหรับการคำนวนของพีชคณิตบลูลีนในคำสั่ง AND , OR , XOR และ TEST เลขฐานสิบคือ 12 ฐานสิบหกคือ C และฐานสองคือ 1100B

OCTAL รูปแบบของเลขฐานแปดมีตัวเลข 8 ตัวคือ 0 - 7 ตามด้วยตัวอักษร O หรือ Q เช่น 23Q 12340

REAL แอสแซมเบลอร์จะกำหนดค่า REAL ในเลขฐานสิบ หรือค่าคงที่ในเลขฐานสิบหกตามด้วยตัวอักษร R เป็นรูปแบบของ floating-point format

เราจะต้องแน่ใจว่าค่าที่ใช้เป็นตัวอักขระหรือค่าคงที่ที่เป็นตัวเลข ตัวอักขระที่กำหนดเป็น DB '12' รหัส ASCII จะเป็น 3132H ค่าคงที่ที่เป็นตัวเลขจะกำหนดดังนี้ DB 12 จะเป็นไบนารีดังนี้ 0CH

ตัวอย่างโปรแกรมในรูป 5.1 แสดงการใช้คำสั่งเทียมในการกำหนดตัวอักขระสตริงและค่าที่เป็นตัวเลข แอสแซมเบลอร์จะเปลี่ยนเป็น Object file ที่อยู่ทางซ้ายดังตัวอย่าง โปรแกรมนี้เพียงแต่กำหนด DATA SEFMENT เท่านั้น ไม่ใช่เขียนสำหรับการเอ็กซีคิวส์

DEFINE BYTE : DB

คำสั่งเทียมที่ใช้กำหนดรายการข้อมูลที่ใช้มากที่สุดคือ Define Byte (DB) ซึ่งทำหน้าที่สร้างหน่วยความจำสำหรับการเก็บข้อมูลเป็นไบท์หรือกลุ่มของไบท์ ซึ่งมีกฎเกณฑ์ดังต่อไปนี้

```
[name]  DB   initialvalue [,initialvalue] . . . .
```

ตัวอย่าง 5.1

```
Mask    DB   01111101B      ;7DH
List    DB   10H,20H,41H,20H
Negat   DB   -185
```

สมมุติตัวแปร LIST ที่เก็บข้อมูลในตำแหน่ง 00H แสดงค่าออฟเซ็ทดังต่อไปนี้

Offset:	00	01	02	03
Values:	10	20	41	02

เราจะเห็นว่า LIST เป็นค่าออฟเซ็ทหรือแสดงตำแหน่งไบไบท์แรกของจำนวนใน LIST ส่วนไบท์อื่นๆเพิ่มขึ้นตามแอดเดรสของหน่วยความจำ ค่าคงที่อาจใช้ฐานที่แตกต่างกันได้ดังนี้

```
LIST    DB   10d,14h,41h,00000010b
BYTE1   DB   255            ;=FF
BYTE2   DB   128            ;=80H
BYTE3   DB   91             ;=5BH
BYTE4   DB   0              ;=00H
BYTE5   DB   -1             ;=FFH
BYTE6   DB   -91            ;=A5H
BYTE7   DB   -128           ;=80H
```

```
                        page 60,132
                TITLE   EXDEF (EXE)  Define data directives
                        .MODEL SMALL
                        .DATA
                ;       Define Byte - DB:
                ;       -----------------
0000  00                FLD1DB  DB  ?           ;Uninitialized
0001  20                FLD2DB  DB  32          ;Decimal constant
0002  20                FLD3DB  DB  20H         ;Hex constant
0003  59                FLD4DB  DB  01011001B   ;Binary constant
0004  000A[ 00 ]        FLD5DB  DB  10 DUP(0)   ;Ten zeros
000F  50 65 72 73 6F 6E FLD6DB  DB  'Personal Computer'
      61 6C 20 43 6F 6D
      70 75 74 65 72                            ;Character string
0019  33 32 36 35 34    FLD7DB  DB  '32654'     ;Numbers as chars
0024  01 4A 61 6E 02 46 FLD8DB  DB  01,'Jan',02,'Feb',03,'Mar'
      65 62 03 4D 61 72                         ;Table
```

```
                                ;       Define Word - DW:
                                ;       -----------------
0030  FFF0                      FLD1DW  DW  0FFF0H      ;Hex constant
0032  0059                      FLD2DW  DW  01011001B   ;Binary constant
0034  001F R                    FLD3DW  DW  FLD7DB      ;Address constant
0036  0003 0004 0007            FLD4DW  DW  3,4,7,8,9   ;Table of 5 constants
      0008 0009
0040  0005[ 0000]               FLD5DW  DW  5 DUP(0)    ;Five zeros

                                ;       Define Doubleword - DD:
                                ;       -----------------------
004A  00000000                  FLD1DD  DD  ?           ;Uninitialized
004E  00007F3C                  FLD2DD  DD  32572       ;Decimal value
0052  0000000E 00000031         FLD3DD  DD  14,49       ;Two constants
005A  00000001                  FLD4DD  DD  FLD3DB - FLD2DB
                                                        ;Diff betw addresses
005E  00005043                  FLD5DD  DD  'PC'        ;Character string

                                ;       Define Quadword - DQ:
                                ;       ---------------------
0062  0000000000000000          FLD1DQ  DQ  ?           ;Uninitialized
006A  474D000000000000          FLD2DQ  DQ  04D47H      ;Hex constant
0072  40E7FC000000000000       FLD3DQ  DQ  32572       ;Decimal constant

                                ;       Define Tenbytes - DT:
                                ;       ---------------------
007A  0000000000000000          FLD1DT  DT  ?           ;Uninitialized
      0000
0084  5634120000000000          FLD2DT  DT  123456      ;Decimal constant
      0000
008E  4350000000000000          FLD3DT  DT  'PC'        ;Character string
      0000
                                END
```

รูป 5-1 DEFINATIONS OF CHARACTER STRINGS AND NUMERIC VALUES

```
Segments and Groups:
                N a m e                 Length  Align   Combine Class

DGROUP . . . . . . . . . . . .  GROUP
   DATA . . . . . . . . . . . .  0098    WORD    PUBLIC  'DATA'
_TEXT . . . . . . . . . . . .  0000    WORD    PUBLIC  'CODE'

Symbols:
                N a m e                 Type    Value   Attr

FLD1DB . . . . . . . . . . . .  L BYTE  0000    _DATA
FLD1DD . . . . . . . . . . . .  L DWORD 004A    _DATA
FLD1DQ . . . . . . . . . . . .  L QWORD 0062    _DATA
FLD1DT . . . . . . . . . . . .  L TBYTE 007A    _DATA
FLD1DW . . . . . . . . . . . .  L WORD  0030    _DATA
FLD2DB . . . . . . . . . . . .  L BYTE  0001    _DATA
FLD2DD . . . . . . . . . . . .  L DWORD 004E    _DATA
FLD2DQ . . . . . . . . . . . .  L QWORD 006A    _DATA
FLD2DT . . . . . . . . . . . .  L TBYTE 0084    _DATA
FLD2DW . . . . . . . . . . . .  L WORD  0032    _DATA
FLD3DB . . . . . . . . . . . .  L BYTE  0002    _DATA
FLD3DD . . . . . . . . . . . .  L DWORD 0052    _DATA
FLD3DQ . . . . . . . . . . . .  L QWORD 0072    _DATA
FLD3DT . . . . . . . . . . . .  L TBYTE 008E    _DATA
FLD3DW . . . . . . . . . . . .  L WORD  0034    _DATA
FLD4DB . . . . . . . . . . . .  L BYTE  0003    _DATA
FLD4DD . . . . . . . . . . . .  L DWORD 005A    _DATA
FLD4DW . . . . . . . . . . . .  L WORD  0036    _DATA
FLD5DB . . . . . . . . . . . .  L BYTE  0004    _DATA   Length = 000A
FLD5DD . . . . . . . . . . . .  L DWORD 005E    _DATA
FLD5DW . . . . . . . . . . . .  L WORD  0040    _DATA   Length = 0005
FLD6DB . . . . . . . . . . . .  L BYTE  000E    _DATA
FLD7DB . . . . . . . . . . . .  L BYTE  001F    _DATA
FLD8DB . . . . . . . . . . . .  L BYTE  0024    _DATA
```

ค่าคงที่ ตัวอักขระ และสตริงอาจจะอยู่ในการกำหนดร่วมกันก็ได้

```
DB    10,'A',20H,'ABC'
```

ในหน่วยความจำจะจัดเก็บดังต่อไปนี้

```
VALUES:    0A  41   20   41   42   43
```

ข้อมูลในหน่วยความจำถ้าไม่ได้กำหนดข้อมูลให้ใช้เครื่องหมาย ? เป็นตัว Operator

```
count     DB    ?
```

ตัว Operator ที่กำหนดข้อมูลซ้ำกันคือ DUP ที่กำหนดข้อมูลในไบท์เดียวหรือแบบหลายๆไบท์ก็ได้ ตัวอย่างในการทำสำเนาข้อมูล 20 ไบท์ดังนี้

```
DB    20 dup(0)
DB    20 dup ('stack')
DB    50 dup ('*')
```

DEFINE WORD (DW)

ตัวอย่าง 5.2 การใช้ DW มีดังนี้

```
DW    0,0,0                ;DEFINE 3 WORDS OF STORAGE
DW    0,65535              ;LOWEST AND HIGHEST UNSIGNED VALUES
DW    -32768,+32767        ;LOWSET AND HIGHEST SIGNED VALUES
DW    256*2                ;CONSTANT EXPRESSION = 512
DW    100 DUP(?)           ;100 WORDS
DW    32767                ;7FFFH
DW    -1                   ;0FFFFH
DW    ' '                  ;0020H
DW    'OK'                 ;4F4BH
```

EQU DIRECTIVE

คำสั่งเทียม EQU ไม่ใช้เป็นตัวกำหนดรายการข้อมูล แต่ใช้เป็นตัวกำหนดค่าให้แอสแซมเบลอร์สำหรับใช้ร่วมกับคำสั่งอื่นๆ พิจารณาจากการใช้คำสั่ง EQU ที่กำหนดค่าใน Data segment

```
TIMES     EQU     10
```

หลังจากที่เรากำหนดค่า TIMES = 10 แล้ว เราก็สามารถใช้ตัวแปรที่กำหนดโดยคำสั่ง EQU ไปใช้ร่วมกับคำสั่งได้ดังนี้

```
FLELDA    DB    TIMES DUP(?)
```

มีการทำงานเหมือนกับคำสั่ง

```
FIELDA    DB    10 DUP(?)
```

หรือเราสามารถเขียนร่วมกับนิวรีมนิคของคำสั่งภาษาแอสแซมบลี้ดังนี้

```
COUNTER   EQU     50
          MOV     CX,COUNTER
```

คำสั่งการเคลื่อนย้ายข้อมูล (DATA TRANSFER INSTRUCTIONS)

คำสั่ง MOV

คำสั่ง MOV เป็นการทำสำเนาข้อมูลจากโอเปอร์รานด์ ไปยังที่อื่นๆ อาจจะเป็นข้อมูลขนาด 8 บิต 16 บิต 32 บิต คำสั่ง MOV เป็นคำสั่งในการเคลื่อนย้ายข้อมูลมีกฎเกณฑ์ดังนี้

```
MOV  destination,source
```

การเคลื่อนย้ายข้อมูลจากตัวส่งไปยังตัวรับ ดังนั้นโอเปอร์รานด์ของตัวส่งจะไม่มีการเปลี่ยนแปลง โอเปอร์รานด์ของตัวส่งอาจจะเป็น Immediate data , register , memory โอเปอร์รานด์ของตัวรับอาจจะเป็น register , memory ดังตัวอย่างต่อไปนี้

immediate data	to	register or memory
register	to	register
register	to	memory
memory	to	register

คำสั่ง MOV ไม่สามารถเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ ดังนั้นข้อมูลจะต้องเคลื่อนย้ายข้อมูลผ่านรีจีสเตอร์ ตัวอย่างการเคลื่อนย้ายข้อมูล Var1 ไปที่ Var2 เขียนได้ดังนี้

```
MOV  ax,var1
MOV  var2,ax
```

Register Operands. คำสั่ง MOV กลุ่มนี้เป็นการเคลื่อนย้ายข้อมูลระหว่างรีจีสเตอร์กับรีจีสเตอร์ ยกเว้นเซคเม้นต์รีจีสเตอร์ CS กับ IP

```
MOV  AX,BX
MOV  DL,AL
MOV  CX,AX
```

Immediate Operands คำสั่ง MOV กลุ่มนี้ใช้ในการเซ็ทค่าให้กับรีจีสเตอร์หรือหน่วยความจำดังนี้

```
MOV  BL,01          ;MOVE 8 BIT VALUE TO BL
MOV  BX,01          ;MOVE 16 BIT VALUE TO BX
MOV  BX,1000H       ;MOVE 16 BIT VALUE TO BX
MOV  TOTAL,1000H ;MOVE 16 BIT VALUE TO A VARIABLE
```

Direct Addressing ชื่อของตัวแปรอาจจะเป็นรหัสหนึ่งของโอเปอร์แรนด์ในคำสั่ง MOV เช่น

```
MOV         AL,COUNT
MOV         COUNT,BL
```

ถ้าท่านเก็บข้อมูลอยู่ในรูปอะเรย์ในการใช้แบบ Direct addressing ดังนี้

```
array     DB   10h,20h,30h,40h,50h
array:    +0   +1   +2   +3   +4
          10H  20H  30H  40H  50H
```

ฉนั้นการกำหนดตำแหน่งหน่วยความจำ array + 1 จะมีข้อมูลคือ 20H ดังตัวอย่างต่อไปนี้

```
MOV       al,array       ;contains = 10h
MOV       dl,array + 1   ;contains = 20h
MOV       bh,array + 3   ;contains = 30h
```

คำสั่ง XCHG

คำสั่ง XCHG เป็นการแลกเปลี่ยนข้อมูลระหว่างรีจีสเตอร์ 2 ตัว หรือรีจีสเตอร์กับตัวแปร มีกฎเกณฑ์ดังต่อไปนี้

```
XCHG      op1,op2
```

ตัวอย่าง 5.3

```
XCHG      AX,BX
XCHG      AH,AL
XCHG      VAR1,AX ;exchang memory operand with BX
```

การใช้คำสั่ง XCHG แลกเปลี่ยนข้อมูลโอเปอร์รานด์ของหน่วยความจำของ VALUE1 กับ VALUE2 สามารถเขียนคำสั่งได้ดังนี้

```
          MOV       AL,VALUE1
          XCHG      AL,VALUE2
          MOV       VALUE1,AL
          ....
          ....
VALUE1    DB        0AH
VALUE2    DB        14H
```

การกำหนดแอดเดรส ADDRESSING MODE

การกำหนดแอดเดรสของไมโครโปรเซสเซอร์ แบ่งออกเป็นประเภทใหญ่ 3 ส่วนคือ

-Immediate

-Register

-Memory --------Direct

Indirect--------Register Indirect

Based

Indexed

Based and Indexed

MODE	EXAMPLE	DESCRIPTION
Direct	op1	Effective address is
	bytelist	displacement
Register indirect	[bx]	Effective address is the
	[si]	contains of register
	[di]	
	[bp]	
Base	list[bx]	Effective address is the sum of
	[bp + 1]	a base register and a displacement
Indexed	list[si]	Effective address is the sum of
	[list + di]	an indexed register and a
	[di + 2]	displacement
Based indexed	[bx + si]	Effective address is the sum of
	[bx][si]	a base and an index register
	[bp + di]	
Based indexed with displacement	[bx + si +2]	Effective address is the sum of a base register,an index register and a displacement

IMMEDIATE ADDRESSING

รูปแบบการอ้างอิงแอดเดรสชนิดนี้ รีเจอร์รานตัวที่ 2 เป็นค่าคงที่ เพื่อส่งต่อไปยังรีเจอร์รานตัวที่ 1 อาจจะเป็นรีจีสเตอร์หรือหน่วยความจำก็ได้ ดังตัวอย่าง

```
SAVE    DB      ?
        ...
        ADD     CX,12           ;ADD 12 TO CX
        MOV     SAVE,25         ;MOVE 25 TO SAVE
        MOV     BX,OFFSET SAVE  ;LOAD OFFSET OF SAVE IN BX
```

REGISTER ADDRESSING

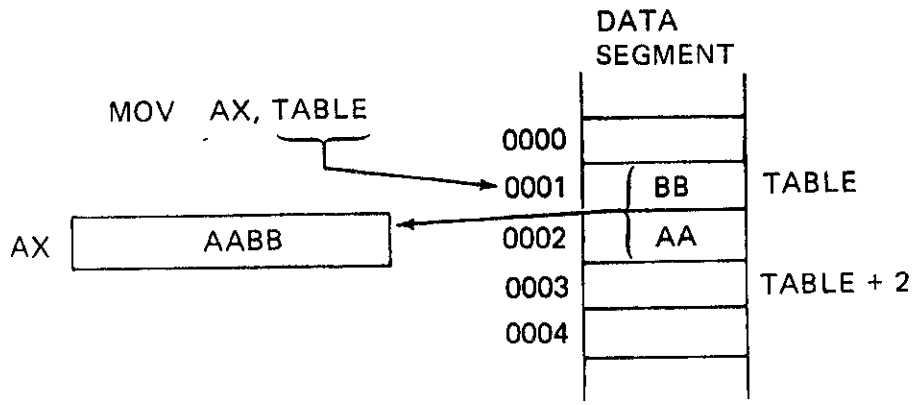
คำสั่งที่อ้างอิลแอดเดรสชนิดนี้เป็นการอ้างอิงการทำงานของรีจีสเตอร์ เป็นการประมวลผลที่เร็วที่สุด แต่ไม่การอ้างอิงถึงหน่วยความจำ ตัวอย่าง

```
MOV     CL,AH       ;MOVE CONTAINS OF AH TO CL
ADD     AX,BX       ;ADD CONTAINS BX TO AX
```

DIRECT MEMORY ADDRESSING

รูปแบบของการอ้างอิงแอดเดรสชนิดนี้จะอ้างถึงตำแหน่งของหน่วยความจำ ตัวอย่าง

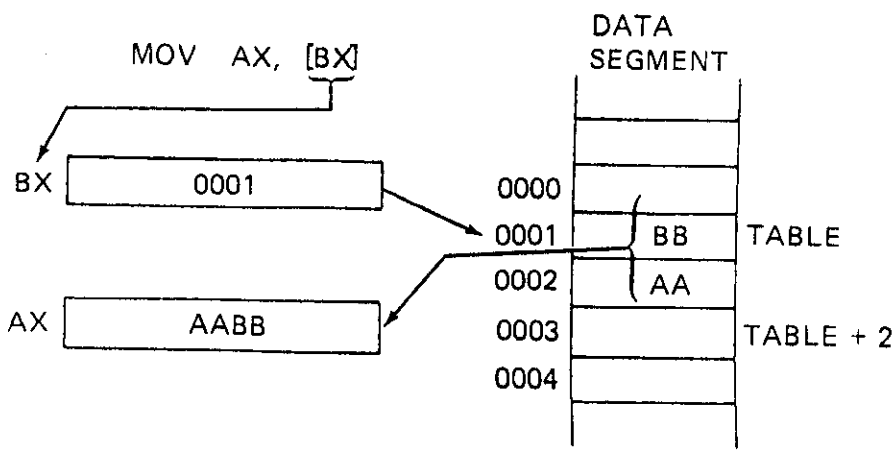
```
WORD1   DW      0
BYTE2   DB      0
        ...
        MOV   AX,WORD1      ;LOAD WORD1 IN AX
        ADD   BYTE2,CL      ;ADD CL TO BYTE2
```



รูป 5-2 **Direct addressing.**

REGISTER INDIRECT MODE

```
MOV  BX,[BX]
MOV  CX,[SI]
INC  [DI]
```



รูป 5-3 **Register indirect addressing.**

ตัวอย่าง 5.4

```
MOV  BX,OFFSET TABLE
MOV  AX,[BX]
MOV  BYTE PTR [BX],0
MOV  WORD PTR [BX],0
```

ตัวอย่าง 5.5 จงเขียนโปรแกรมในการบวกเลขเก็บผลลัพธ์ไว้ใน AX จากอะเรย์ข้อมูลต่อไปนี้

```
W    DW   10,20,30,40,50,60,70,80,90,100

          XOR  AX,AX          ;AX HOLDS SUM
          LEA  SI,W           ;SI POINT TO ARRAY W
          MOV  CX,10          ;CX = COUNTER
ADDNOS:
          ADD  AX,[SI]        ;SUM =SUM = element
          ADD  SI,2
          LOOP ADDNOS
```

ตัวอย่าง 5.5.1 จงเขียน Procedure ในการ Reverse เวิร์ดข้อมูลของอะเรย์จำนวน N เวิร์ด หมายความว่าเวิร์ด Nth จะเป็นเวิร์ดแรกของอะเรย์ และเวิร์ด (N-1) จะมาเป็นเวิร์ดที่ 2 ของอะเรย์ และต่อๆไป ส่วนเวิร์ดแรกของอะเรย์จะมาเป็นเวิร์ด Nth โปรแกรมนี้จะใช้ SI เป็นตัวชี้อะเรย์ และ BX เก็บค่าจำนวน N เวิร์ด

```
;PROCEDURE REVERSE A WORD ARRAY
;INPUT : SI = OFFSET OF ARRAY
;        BX = NUMBER OF ELEMENTS
;OUTPUT : REVERSE ARRAY
          PUSH     AX        ;SAVE REGISTER
          PUSH     BX
          PUSH     CX
          PUSH     SI
          PUSH     DI
```

```
;MAKE DI POINT TO Nth WORD
        MOV     DI,SI       ;DI POINT TO 1st WORD
        MOV     CX,BX       ;CX = N
        DEC     BX          ;BX = BX - 1
        SHL     BX,1        ;BX = 2 X (N-1)
        ADD     DI,BX       ;DX POINT TO Nth WORD
        SHR     CX,1        ;CX = N/2 = NO. OS SWAPS TO DO
;SWAPS ELEMENTS
XCHG_LOOP:
        MOV     AX,[SI]     ;GET AN ELEMENT IN LOWER HALF
                            ;OF ARRAY
        HCHG    AX,[DI]     ;INSERT IN UPPER HALF
        MOV     [SI],AX     ;CPMPLETE EXCHANG
        ADD     SI,2        ;MOVE PTR
        ADD     DI,2        ;MOVE PTR
        LOOP    XCHG_LOOP
        POP  DI
        POP  SI
        POP  CX
        POP  BX
        POP  AX
        RET
REVERSE ENDP
```

BASED AND INDEX MODES

การอ้างอิงแอดเดรสชนิดนี้ ค่าของออฟเซ็ตแอดเดรสของโอเปอรานด์เป็นการรวมค่าของ Displacement กับข้อมูลในรีจีสเตอร์ดังต่อไปนี้เช่น

A - เป็นค่าออฟเซ็ตแอดเดรสของตัวแปร

-2 - เป็นค่าคงที่ + หรือ -

A + 2 - เป็นค่าออฟเซ็ตแอดเดรสของตัวแปรและค่าคงที่ (+ หรือ -)

ถ้าสมมุติค่าของ A เป็นตัวแปรจะได้กฎเกณฑ์ดังนี้

[Register + Displacement]

[Displacement + register]

[Register] + displacement

Displacement + [Register]

Displacement[Register]

Reg หมายถึงค่าของรีจีสเตอร์เช่น BX,BP,SI,DI ส่วนรีจีสเตอร์ BX,SI,DI ใช้กับหน่วยความจำในส่วนของ Data segment และ รีจีสเตอร์ BP ใช้กับหน่วยความจำในส่วนของ สแตค ซึ่ง B ย่อมาจาก BASED และค่า I ย่อมาจาก INDEXED

ตัวอย่าง 5.6 ถ้าเรากำหนดค่าของ W เป็นค่าของเวิร์ดอะเรย์ และค่าของรีจีสเตอร์ BX มีค่าเท่ากับ 4 เราสามารถเขียนคำสั่งได้ดังนี้

```
MOV     AX,W[BX]
```

ค่าของ Displacment คือค่าออฟเซ็ตแอดเดรสของตัวแปร W การเขียนคำสั่งจะประกอบดังต่อไปนี้

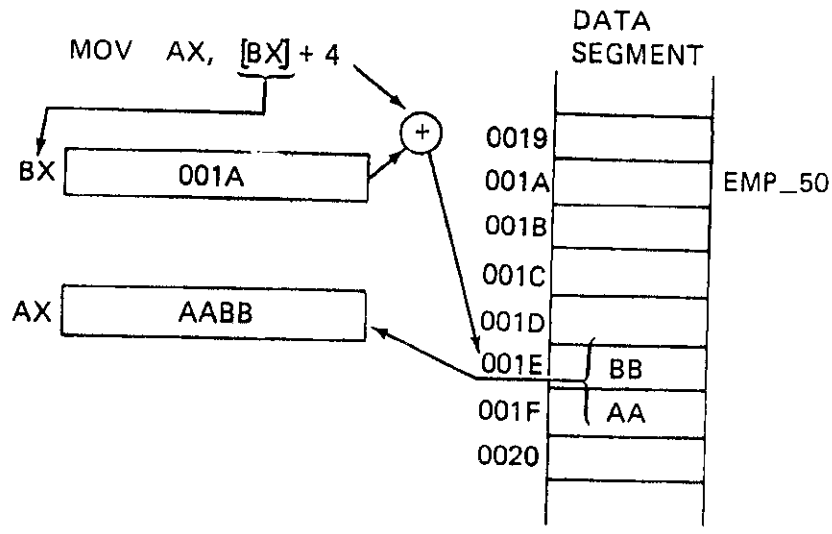
```
MOV     AX,[W+BX]
MOV     AX,[BX+W]
MOV     AX,W+[BX]
MOV     AX,[BX]+W
```

หรือ

```
MOV     AX,[SI+2]
```

การใช้งานของรหมดนี้ สมมุติว่าค่าของ ALPHA มีดังต่อไปนี้

```
ALPHA   DW   0123H,0456H,0489H,0ABCDH
```



รูป 5-4 **Base relative addressing.**

ซึ่งค่าของ ALPHA กำหนดอยู่ในหน่วยความจำที่เรียกว่า DATA SEGMENT (DS) และกำหนดค่าของรีจีสเตอร์ต่างๆ ดังต่อไปนี้

```
BX = 2              Offset address 0002 = 1084H
SI = 4              Offset address 0004 = 2BACH
DI = 1
```

จากคำสั่งต่อไปนี้ จงหาผลลัพธ์ที่ได้จากการเอ็กซีคิวส์คำสั่ง

```
a)   MOV  AX,[ALPHA+BX]      ;AX=0456H
b)   MOV  BX,[BX+2]          ;BX=2BACH
c)   MOV  CX,ALPHA[SI]       ;CX=0789H
d)   MOV  AX,-2[SI]          ;AX=1084H
e)   MOV  BX,[ALPHA+3+DI)    ;BX,0789H
```

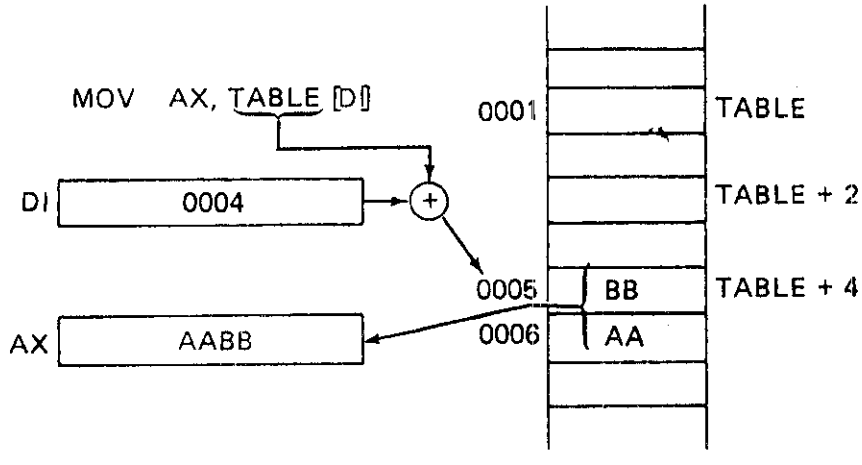
ตัวอย่าง 5.7 จงเขียนโปรแกรมในการบวกเลขเก็บผลลัพธ์ไว้ใน AX จากอะเรย์ข้อมูลต่อไปนี้ โดยใช้ BASE MODE.

```
W     DW    10,20,30,40,50,60,70,80,90,100

            XOR   AX,AX          ;AX HOLDS SUM
            XOR   BX,BX          ;CLEAR BASE REGISTER
            MOV   CX,10          ;SET COUNTER
ADDNOS:
            ADD   AX,W[EX]       ;SUM =SUM+ELEMENT
            ADD   BX,2           ;INDEX NEXT ELEMENT
            LOOP  ADDNOS
```

INDEXED ADDRESSING

การอ้างอิลแอดเดรสชนิดนี้จะใช้เทคนิคคอมพิวเตอร์เซคเม้นต์ ออฟเซ็ทแอดเดรส โดยใช้รีจีสเตอร์ BX , DI , SI และ BP ซึ่งอยู่ในวงเล็บเป็นค่าของอินเดค รีจีสเตอร์ BX , DI , SI ทำงานร่วมกับรีจีสเตอร์ DS รีจีสเตอร์ BP ทำงานร่วมกับรีจีสเตอร์ SS ตัวอย่าง



รูป 5-5 Direct Indexed addressing.

```
DATAFLD     DB      ?
            ...
            MOV  BX,OFFSET DATAFLD   ;LOAD BX WITH OFFSET
            MOV  [BX],0              ;MOVE 0 TO DATAFLD
            MOV      DX,ARRAY[BX]
            MOV      DX,[DI+ARRAY]
            MOV      DX,[ARRAY+SI]
            MOV      AX,[BP+2]
            MOV      DX,2[SI]
            MOV      AL,[BP][SI]
            MOV      DX,[BX+SI]
            ADD      CX,[DI][BX]
            MOV      CX,ARRAY[BP+SI]
```

ตัวอย่าง 5.8 การแทนค่าตัวอักษรตัวเล็กให้เป็นตัวใหญ่โดยการใช้วิธีการอ้างอิงแบบ INDEXED MODE

```
MSG     DB   'this is a message'

        MOV  CX,17
        XOR  SI,DI
TOP:
        CMP  MSG[SI],' '
        JE   NEXT
        AND  MSG[SI],0DFH
NEXT:   INC  SI
        LOOP TOP
```

THE PTR Operator

การใช้โอเปอร์รานของคำสั่งจะต้องเป็นชนิดเดียวกัน ในคำสั่งต่างๆเราจะเห็นว่าถ้ามีขนาดเป็นไบท์ ต้องมีขนาดไบท์เท่ากันหรือมีขนาดเป็นเวิร์ดก็ต้องเป็นเวิร์ดเหมือนกัน หรือค่าของโอเปอร์รานด์อาจจะเป็นค่าคงที่ก็ได้ดังตัวอย่าง

```
MOV  AX,1        ;AX = WORD
MOV  BH,5        ;BH = BYTE
```

ถ้าเราใช้คำสั่งชนิดไบท์ จะเห็นว่าคำสั่งชนิดนี้ไม่สามารถคอมไพล์ได้เช่น

```
MOV  [BX],1          ;ILLEGAL
```

คำสั่งชนิดเราไม่ได้บอกตัวรับว่าข้อมูลที่จะส่งเป็นไบท์หรือเวิร์ดในตัวชี้ BX ถ้าท่านต้องการบอกตัวรับท่านสามารถเขียนคำสั่งได้ดังนี้

```
MOV  BYTE PTR [BX],1
```

หรือ

```
MOV  WORD PTR [BX],1
```

ตัวอย่าง 5.9 จากโปรแกรมเราสามารถเขียนโปรแกรมโดยแทนที่ตัวอักษร t ด้วยตัวอักษร T

```
MSG      DB   'this is a message'
```

การใช้วิธี Indirect mode

```
LEA  SI,MSG              ;SET POINTER TO MSG
MOV BYTE PTR [SI],'T' ;REPLACE 't' BY 'T'
```

การใช้วิธี Index mode

```
XOR  SI,SI           ;CLEAR SI
MOV  MSG[SI],'T'     ;REPLACE 't' BY 'T'
```

บทสรุป

- ชื่อของรายการข้อมูลจะต้องไม่เป็นคำสงวน
- DB เป็นคำสั่งเทียมที่กำหนดข้อมูลเป็นไบท์ ที่เป็นตัวเลขหรือตัวอักษรก็ได้

- ค่าคงที่เลขฐานสิบและเลขฐานสองจะกำหนดค่าที่แตกต่างกัน พิจารณาจากการบวกเลขฐานสิบกับเลขฐานสิบหกดังนี้

```
ADD   AX,25      ;ADD 25
ADD   AX,25H     ;ADD 37
```

- DW , DD และ DQ เก็บข้อมูลที่เป็นตัวเลขในรูปของภาษาเครื่องสลับกัน
- DB คือการกำหนดข้อมูลสำหรับการประมวลผลครึ่งไบท์เช่น AL,BL ส่วน DW สำหรับข้อมูลขนาด 1 เวิร์ด เช่น AX , BX ส่วน DD สำหรับ extended register เช่น EAX
- คำสั่งที่ใช้มี 4 ชนิดคือ register , immediate , direct , memory และ indexed addressing.

แบบฝึกหัด

5-1. จงอธิบายความหมายของคำต่อไปนี้

a) DW b) DD c) DB d) DQ

5-2. จงกำหนดค่าคงที่ต่อไปนี้ในรายการข้อมูลชื่อ FLDA ถึง FLDE

a) กำหนดข้อมูลเลขฐานสิบหกขนาด 4 ไบท์เทียบเท่าเลขฐานสิบคือ 115

b) กำหนดข้อมูลเลขฐานสิบหกขนาด 1 ไบท์เทียเท่าเลขฐานสิบหกคือ 25

c) ข้อมูลขนาด 2 ไบท์ไม่ต้องกำหนดข้อมูล

d) กำหนดข้อมูลเลขฐานสอง 1 ไบท์ เทียบเท่าเลขฐานสิบคือ 25

e) ใช้ DW กำหนดค่าต่อไปนี้ 16,19,20,27,30

5-3. แสดงค่าเลขฐานสิบหกของข้อมูลต่อไปนี้

a) DB '25' b) DB 24 c) DB 'ABC'

5-4. จงแทนรหัสภาษาเครื่องของข้อมูลต่อไปนี้

a) DB 26H b) DW 2645H c) DD 25733AH d) DQ 25733AH

5-6. จงเขียนการกำหนดข้อมูลให้กับสตริงต่อไปนี้

'MYFILE.DTA' , 'MY NAME IS DEBUG'

5-7. จงเขียนคำสั่งเทียมในการกำหนดค่า 500 เวิร์ด ข้อมูลแต่ละค่ามีข้อมูล 1000H

5-8. จงตรวจสอบว่าค่าของ AX มีค่าอะไรหลังจากการเอ็กซีคิวส์

```
            MOV   AX,ARRAY1
            INC   AX
            ADD   AH,1
            SUB   AX,ARRAY1
            ...
ARRAY1      DW    10H,20H
ARRAY2      DW    30H,40H
```

5-9. จากคำสั่งต่อไปนี้เมื่อมีการเอ็กซัคิวส์ ให้ใส่ค่าเลขฐานสิบหกในโอเปอร์รานด์ทางขวามือ

```
            MOV   AX,ARRAY1       ; AX =
            XCHG  ARRAY2,AX       ;AX =
            DEC   AX
            SUB   ARRAY2,2        ;ARRAY2 =
            MOV   BX,ARRAY2
            ADD   AH,BL           ; AX =
            ...
ARRAY1      DW     20H,10H
ARRAY2      DW     30H,40H
```

----------**********----------