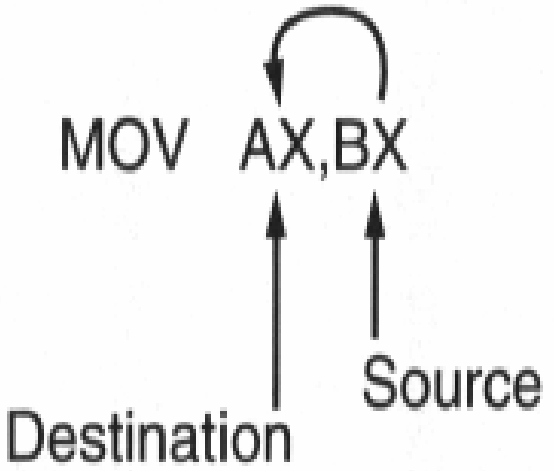
Mode Pengalamatan Data

- **Gambar 3.1** Intruksi MOV yang menggambarkan sumber, tujuan dan aliran data.
- **Gambar 3.2** menggambarkan semua variasi yang mungkin dari mode pengalamatan data dengan menggunakan instruksi MOV.
- Pengalamatan Register : MOV CX,DX or
  MOV ECX,EDX
- Pengalamatan Segera : MOV AL,22H or
  MOV EAX,12345678H
- Pengalamatan Langsung : MOV CX,LIST

Mode Pengalamatan Data (Lanjt.)

FIGURE 3–1 The MOV instruction showing the source, destination, and direction of data flow



Mode Pengalamatan Data (Lanjt.)

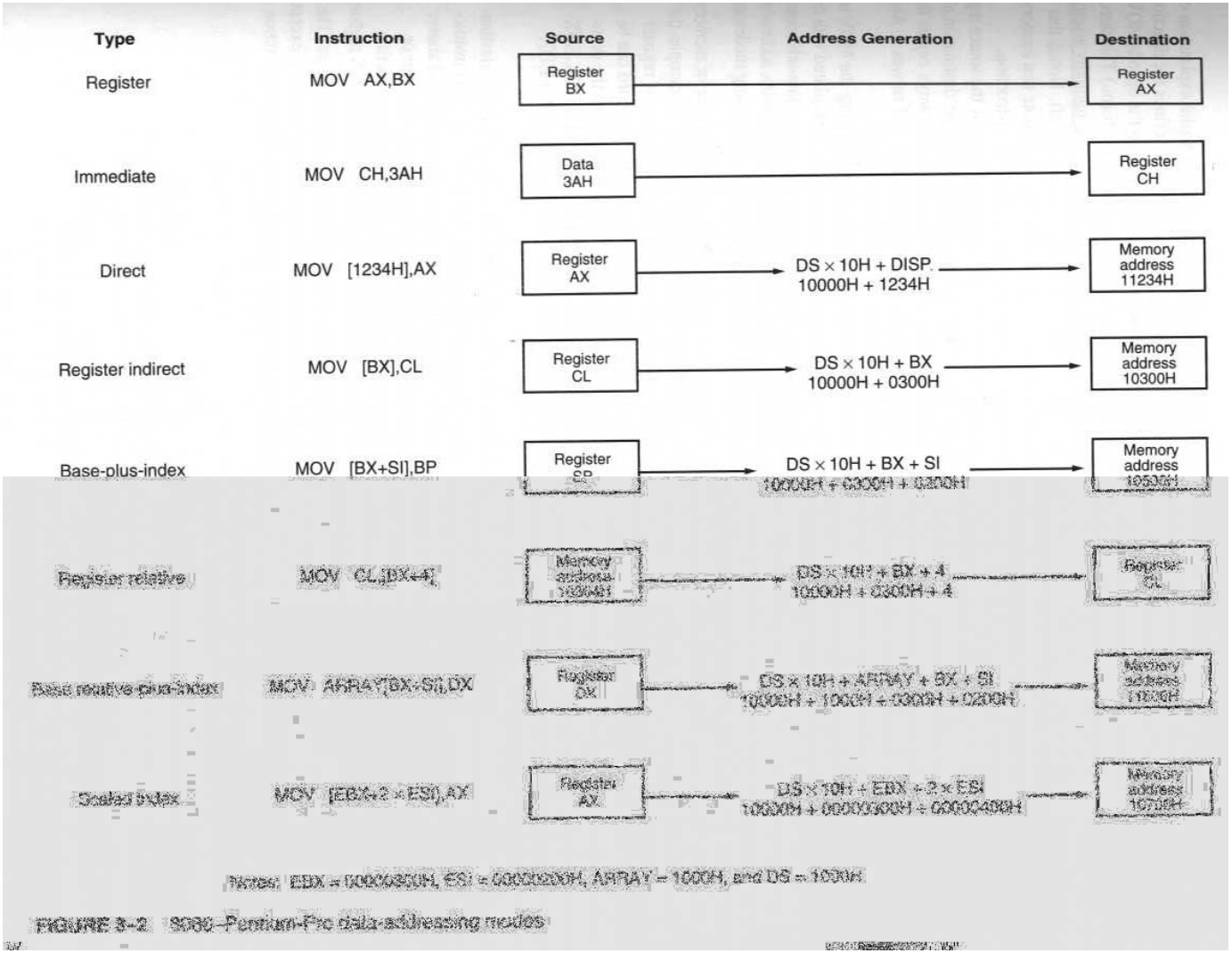


FIGURE 3–2 8086–Pentium-Pro data-addressing modes

Mode Pengalamatan Data (Lanjt.)

- Pengalamatan Base-plus-index :

  MOV [BX+DI], CL or MOV [EAX+EBX],CL

- Pengalamatan Register relative :

  MOV AX,[BX+4] or MOV AX,ARRAY[BX]

- Pengalamatan Base relative-plus-index :

  MOV AX,ARRAY[BX+DI] or
  MOV AX,[BX+DI+4]

- Pengalamatan Scaled-index :

  MOV EDX,[EAX+4*EBX]

Pengalamatan Register

- Pengalamatan register merupakan bentuk pengalamatan data yang paling dikenal, maka akan lebih mudah untuk memakainya.
- Mikroprosesor terdiri dari 8-bit, 16-bit, 32-bit register
  - Jangan pernah menggabungkan jenis register 8 & 16 bit, 8 & 32 bit, 16 & 32 bit karena hal ini tidak diijinkan oleh mikroprosesor dan hasilnya akan mengeluarkan pesan kesalahan pada saat dilakukan perakitan.

Pengalamatan Register (Lanjutan)

- **Tabel 3.1** mengilustrasikan sebagian dari banyak versi yang berbeda dari instruksi pemindahan register.
- **Gambar 3.3** mengilustrasikan fungsi instruksi MOV BC,CX
- **Contoh 3.1** memperlihatkan urutan instruksi rakitan yang menyalin data berbeda-beda antara register 8, 16 dan 32 bit.

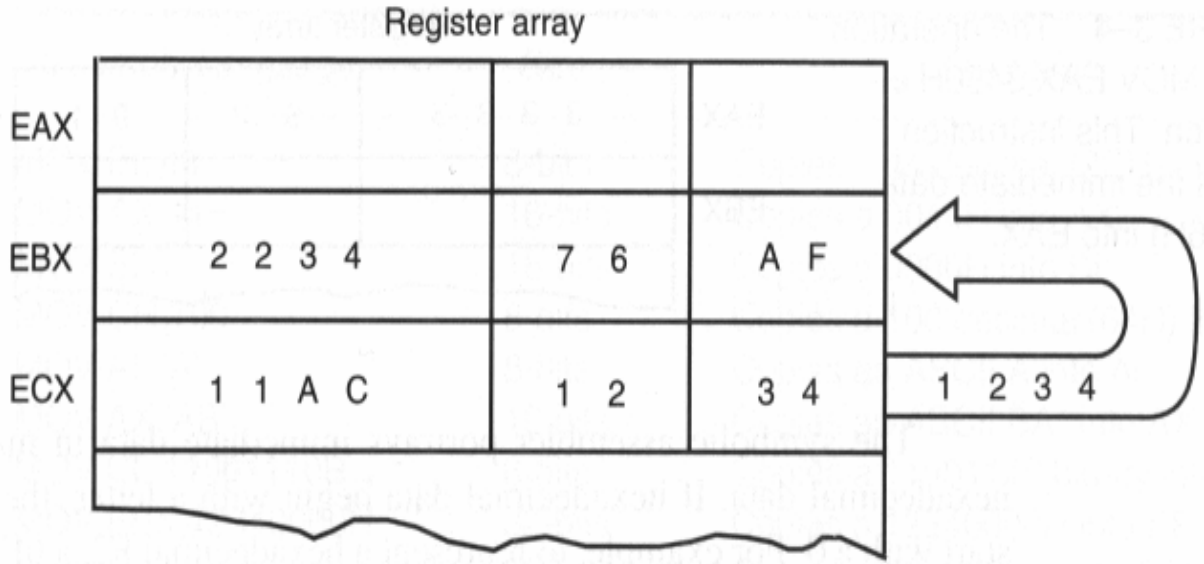
Pengalamatan Register (Lanjutan)

TABLE 3–1 Examples of the register-addressed instructions

Assembly Language	*Size*	*Operation*
MOV AL,BL	8-bits	Copies BL into AL
MOV CH,CL	8-bits	Copies CL into CH
MOV AX,CX	16-bits	Copies CX into AX
MOV SP,BP	16-bits	Copies BP into SP
MOV DS,AX	16-bits	Copies AX into DS
MOV SI,DI	16-bits	Copies DI into SI
MOV BX,ES	16-bits	Copies ES into BX
MOV ECX,EBX	32-bits	Copies EBX into ECX
MOV ESP,EDX	32-bits	Copies EDX into ESP
MOV ES,DS	—	Not allowed (segment-to-segment)
MOV BL,DX	—	Not allowed (mixed sizes)
MOV CS,AX	—	Not allowed (the code segment register may not be the destination register)

Pengalamatan Register (Lanjutan)

FIGURE 3–3 The effect of executing the MOV BX, CX instruction at the point just before the BX register changes. Note that only the rightmost 16-bits of register EBX change.



Pengalamatan Register (Lanjutan)

EXAMPLE 3–1

```
0000 8B C3          MOV AX,BX       ;copy contents of BX into AX
0002 8A CE          MOV CL,DH       ;copy the contents of DH into CL
0004 8A CD          MOV CL,CH       ;copy the contents of CH into CL
0006 66|8B C3       MOV EAX,EBX     ;copy the contents of EBX into EAX
0009 66|8B D8       MOV EBX,EAX     ;copy EAX into EBX, ECX, and EDX
000C 66|8B C8       MOV ECX,EAX
000F 66|8B D0       MOV EDX,EAX
0012 8C C8          MOV AX,CS       ;copy CS into DS
0014 8E D8          MOV DS,AX
0016 8E C8          MOV CS,AX       ;assembles, but will cause problems
```

Pengalamatan Segera

- Istilah segera menyatakan bahwa data segera mengikuti kode operasi heksadesimal dalam memori
  - Data segera merupakan data konstan
  - Instruksi segera MOV memindahkan salinan data segera ke dalam sebuah register atau sebuah lokasi memori.

Pengalamatan Segera (Lanjutan)

- **Gambar 3.4** menggambarkan operasi instruksi MOV EAX,3456H.
- **Contoh 3.2** memperlihatkan berbagai instruksi segera dalam suatu program pendek yang menempatkan 0000H ke dalam register 16-bit AX, BX dan CX

Pengalamatan Segera (Lanjutan)

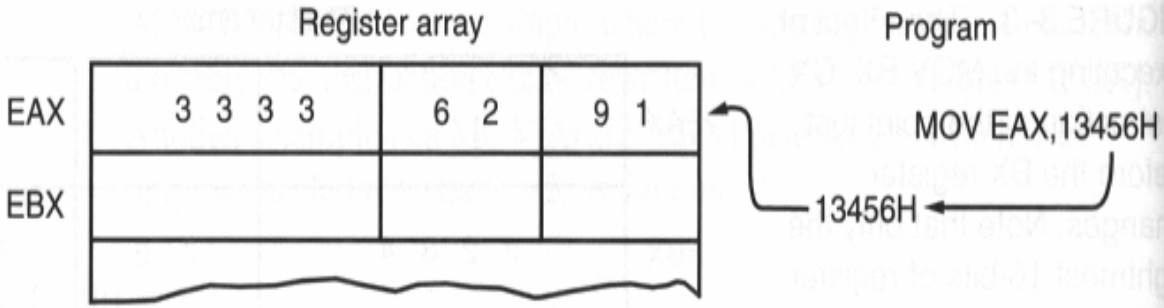


FIGURE 3–4 The operation of the MOV EAX,3456H instruction. This instruction copies the immediate data (13456H) into EAX.

Pengalamatan Segera (Lanjutan)

EXAMPLE 3–2

```
                        .MODEL TINY          ;choose single segment model
0000                    .CODE                ;indicate start of code segment

                        .STARTUP             ;indicate start of program

0100  B8 0000           MOV    AX,0          ;place 0000H into AX
0103  BB 0000           MOV    BX,0000H      ;place 0000H into BX
0106  B9 0000           MOV    CX,0          ;place 0000H into CX

0109  8B F0             MOV    SI,AX         ;copy AX into SI
010B  8B F8             MOV    DI,AX         ;copy AX into DI
010D  8B E8             MOV    BP,AX         ;copy AX into BP

                        .EXIT                ;exit to DOS
                        END                  ;end of file
```

Pengalamatan Data Langsung

- Ada dua bentuk dasar pengalamatan data langsung :
  - Pengalamatan langsung yang menggunakan instruksi MOV antara lokasi memori dan AL, AX atau EAX.
  - Pengalamatan displacement yang digunakan pada hampir semua instruksi dalam kumpulan instruksi.

Pengalamatan Data Langsung(Lanjutan)

- **Pengalamatan data langsung** : MOV AL,DATA **(Gambar 3.5)**
  - **Tabel 3.3** mencatat tiga instruksi pengalamatan langsung.
  - Instruksi MOV mempunyai panjang 3 byte.
- **Pengalamatan displacement** : MOV CL,DATA
  - Hampir mirip dengan pengalamatan langsung kecuali bahwa instruksi itu mempunyai lebar 4 byte bukan 3 byte.

Pengalamatan Data Langsung(Lanjutan)

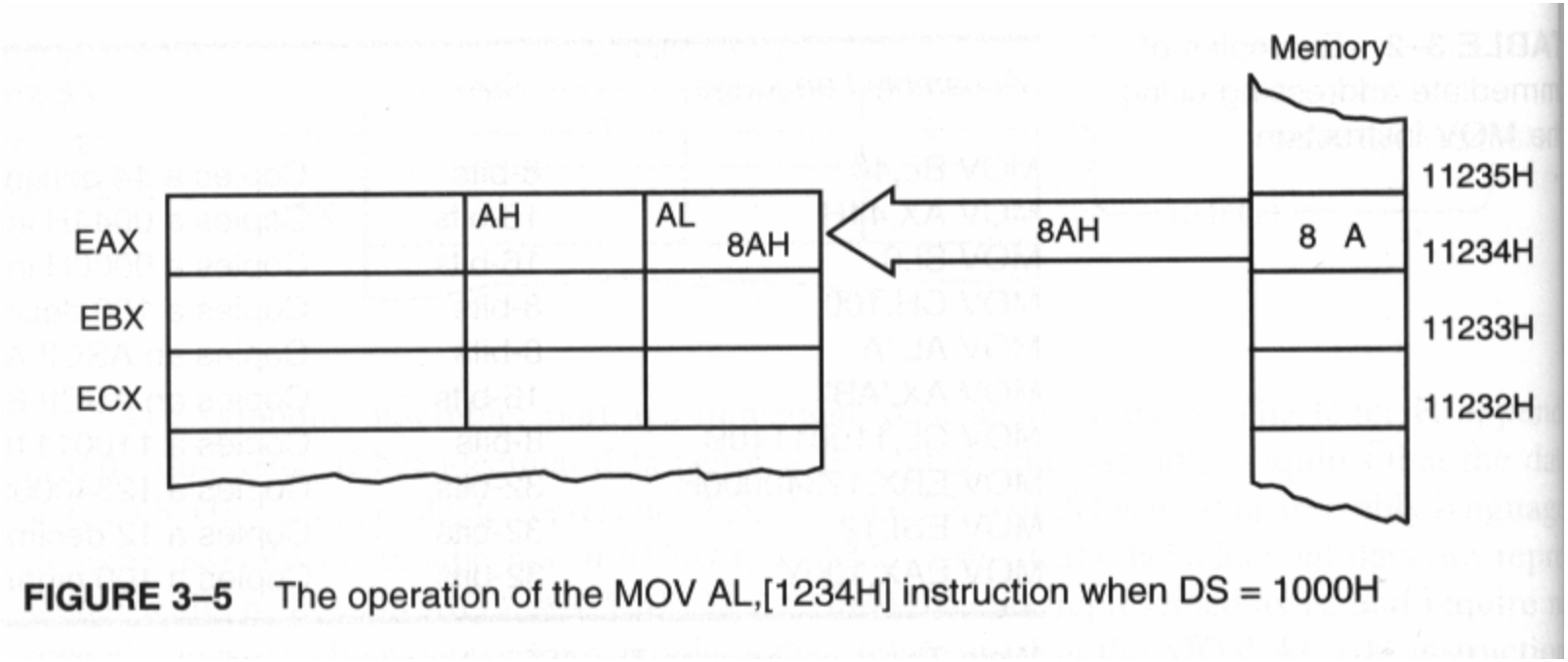


FIGURE 3–5 The operation of the MOV AL,[1234H] instruction when DS = 1000H

Pengalamatan Data Langsung(Lanjutan)

TABLE 3–3 Direct addressed instructions using EAX, AX and AL

Assembly Language	*Size*	*Operation*
MOV AL,NUMBER	8-bits	Copies the byte contents of data segment memory location NUMBER into AL
MOV AX,COW	16-bits	Copies the word contents of data segment memory location COW into AX
MOV EAX,WATER*	32-bits	Copies the doubleword contents of memory location WATER into EAX
MOV NEWS,AL	8-bits	Copies AL into data segment memory location NEWS
MOV THERE,AX	16-bits	Copies AX into data segment memory location THERE
MOV HOME,EAX*	32-bits	Copies EAX into data segment memory location HOME

**Note:* The 80386–Pentium Pro microprocessors will some times use more than three bytes of memory for the 32-bit move between EAX and memory.

Pengalamatan Data Tidak Langsung

- Pengalamatan register tidak langsung memungkinkan data dialamatkan pada lokasi memori melalui offset yang ditunjukkan oleh setiap regiater : BP, BX, DI, and SI
  - MOV AX,[BX] → **Gambar 3.6**
- Data segmen digunakan secara default dengan pengalamatan register tidak langsung atau mode pengalamatan lainnya yang menggunakan BX, DI atau SI, untuk mengalamatkan memori.
  - Jika register BP mengalamatkan memori, maka segmen stack digunakan

Pengalamatan Data Tidak Langsung (Lanjutan)

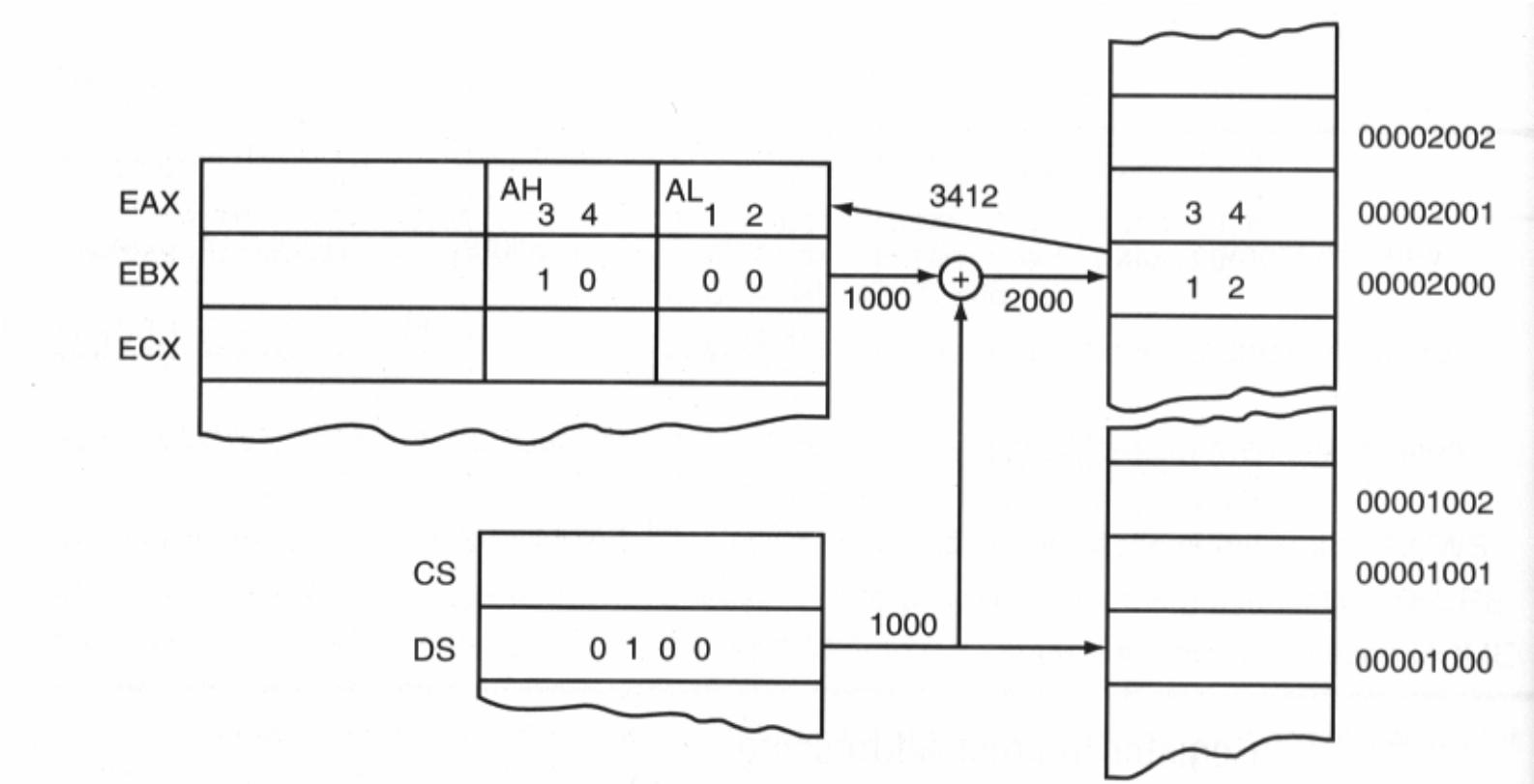


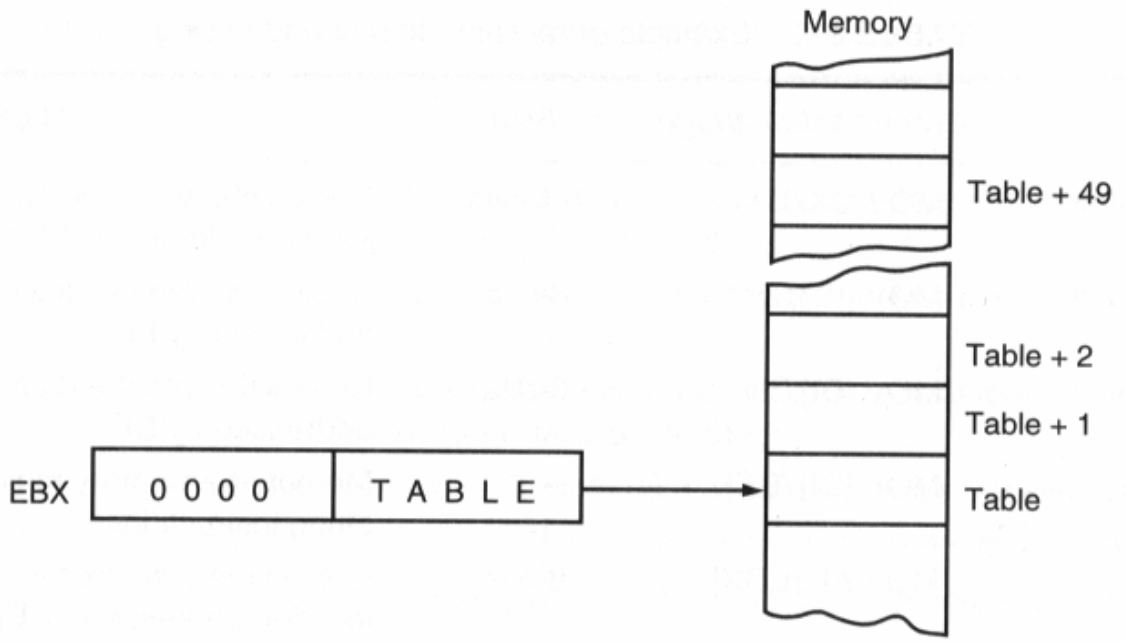
FIGURE 3–6 The operation of the MOV AX,[BX] instruction when BX = 1000H and DS = 0100H. Note that this instruction is shown after the contents of memory are transferred to AX.

Pengalamatan Data Tidak Langsung (Lanjutan)

- Dalam beberapa kasus, pengalamatan tidak langsung memerlukan ukuran data khusus yang ditetapkan dengan direktif assembler khusus BYTE PTR, WORD PTR or DWORD PTR
  - Direktif ini menunjukkan ukuran data memori yang dialamatkan oleh penunjuk memori (PTR)
- Pengalamatan tidak langsung sering digunakan untuk menunjukkan data tabular dalam sistem memori **(Gambar 3.7 & Contoh 3.6)**

Pengalamatan Data Tidak Langsung (Lanjutan)

FIGURE 3–7 An array (TABLE) containing 50 bytes that are indirectly addressed through register BX



Pengalamatan Data Tidak Langsung (Lanjutan)

EXAMPLE 3–6

```
                                .MODEL SMALL                ;select SMALL model
0000                            .DATA                       ;start of DATA segment

0000  0032 [         DATAS      DW     50 DUP (?)           ;setup array of 50 bytes
              0000
                   ]

0000                            .CODE                       ;start of CODE segment
                                .STARTUP                    ;start of program

0017  B8 0000                   MOV       AX,0
001A  8E C0                     MOV       ES,AX             ;address segment 0000 with ES

001C  BB 0000 R                 MOV       BX,OFFSET DATAS   ;address DATAS array
001F  B9 0032                   MOV         CX,50           ;load counter with 50
0022                  AGAIN:
0022  26:A1 046C                MOV       AX,ES:[046CH]     ;get clock value
0026  89 07                     MOV       [BX],AX           ;save clock value in DATAS
0028  43                        INC       BX                ;increment BX to next element
0029  E2 F7                    LOOP      AGAIN             ;repeat 50 times

                                .EXIT                       ;exit to DOS
                                END                         ;end file
```

Pengalamatan Base-Plus-Index

- Mirip dengan pengalamatan tidak langsung
  - Dalam mikroprosesor 8086 - 80286, tipe pengalamatan ini menggunakan satu register basis (BP atau BX) dan register index (DI atau SI) untuk secara tidak langsung mengalamatkan memori.
  - Dalam 80386 dan versi diatasnya, tipe pengalamatan ini mengijinkan kombinasi setiap dua register 32-bit kecuali ESP.
    - MOV DL, [EAX+EBX]

Pengalamatan Base-Plus-Index (lanjutan)

- **Gambar 3.8** menggambarkan bagaimana data dialamatkan untuk instruksi MOV DX, [BX+DI] pada saat mikroprosesor beroperasi dalam mode real
- Penggunaan utama dari mode pengalamatan ini adalah untuk mengalamatkan elemen di dalam suatu array memori.
  - **Gambar 3.9** memperlihatkan penggunaan BX dan DI untuk mengakses sebuah elemen dalam array data.

Pengalamatan Base-Plus-Index (lanjutan)

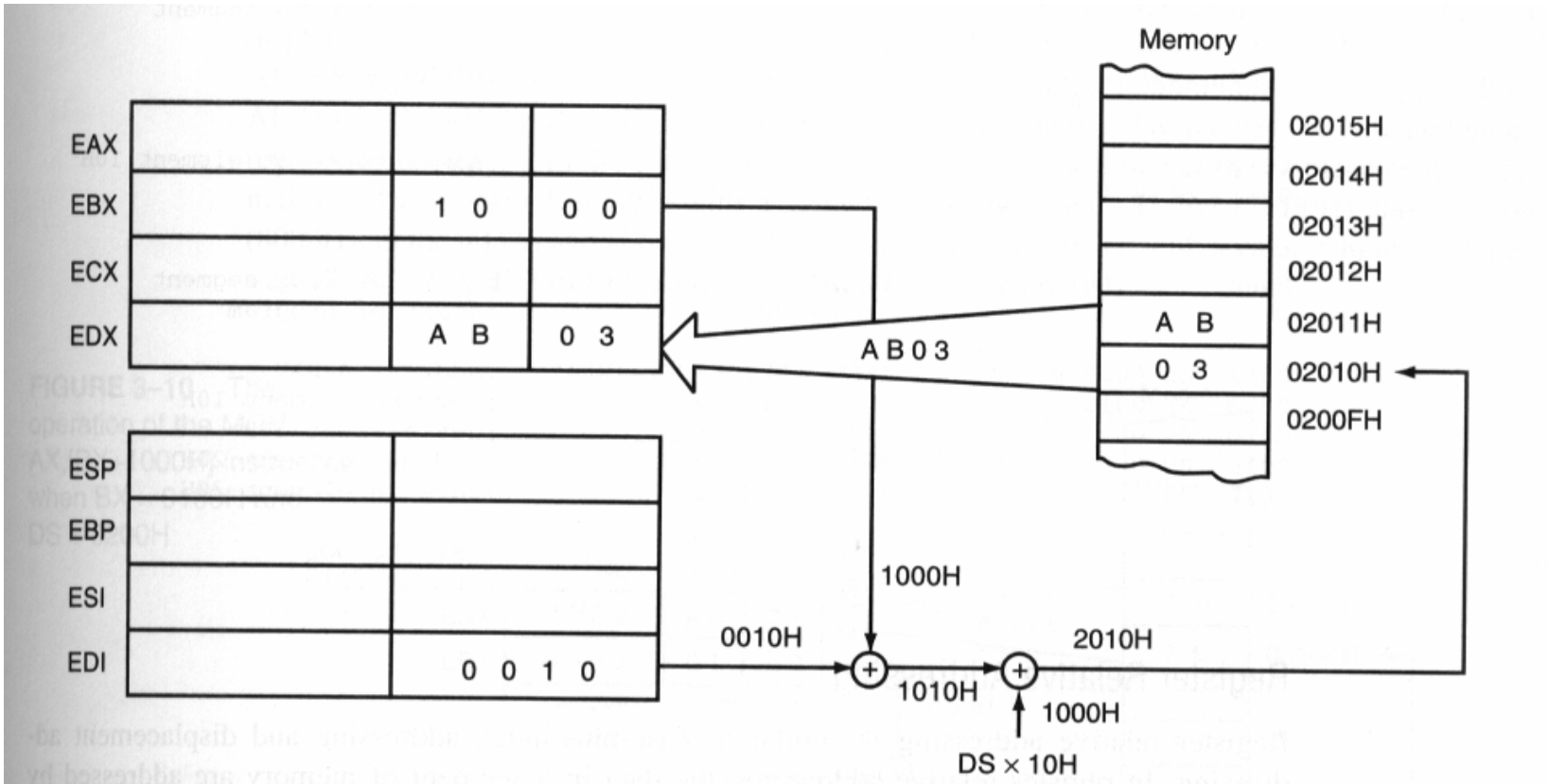


FIGURE 3–8 An example showing how the base-plus-index addressing mode functions for the MOV DX,[BX+DI] instruction. Notice that memory address 02010H is accessed because DS = 0100H, BX = 100H, and DI = 0010H.

Pengalamatan Base-Plus-Index (lanjutan)

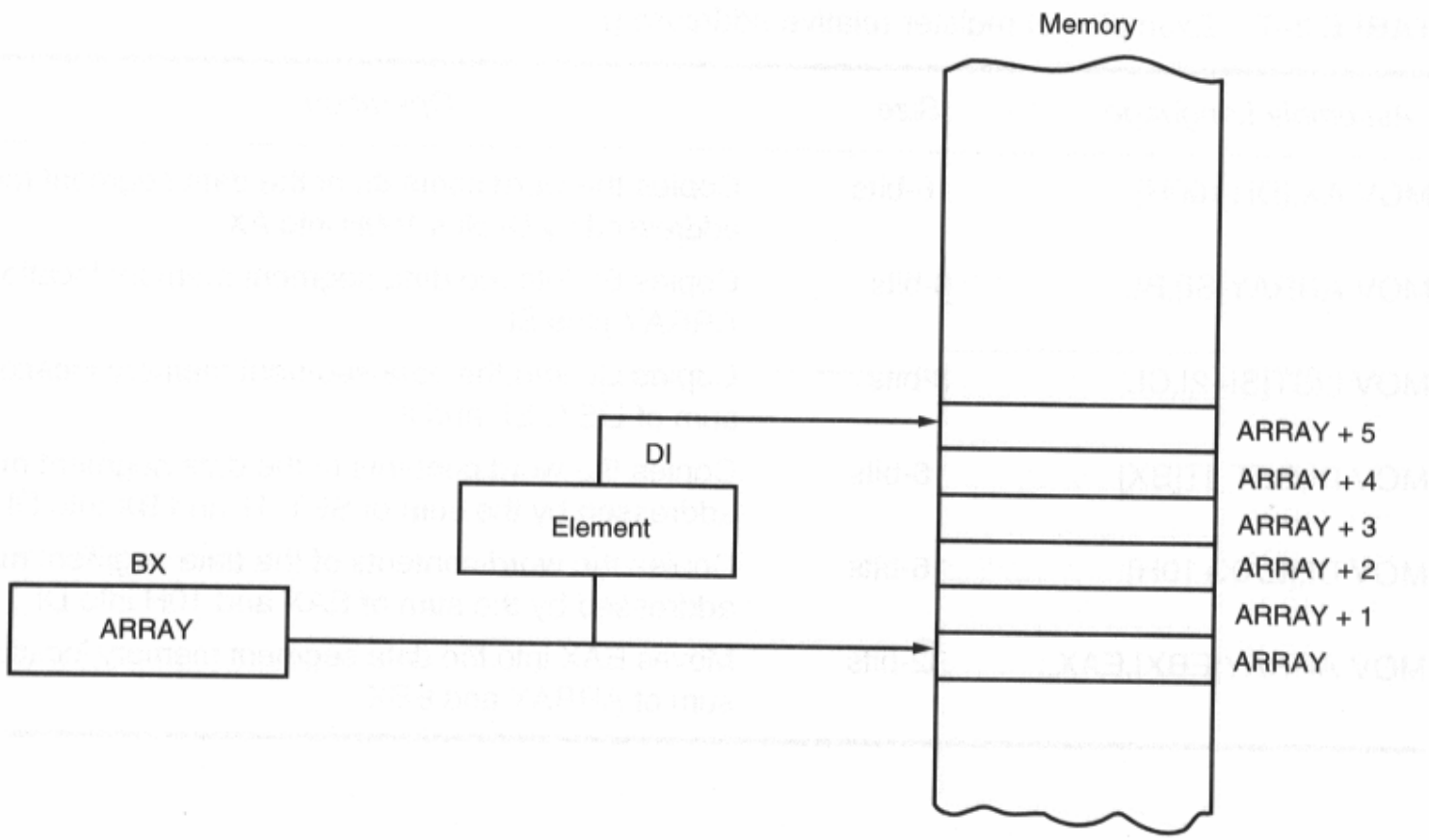


FIGURE 3–9 An example of the base-plus-index addressing mode. Here an element (DI) of an ARRAY (BX) is addressed.

Pengalamatan Base-Plus-Index (lanjutan)

TABLE 3–6 Examples of base-plus-index addressing

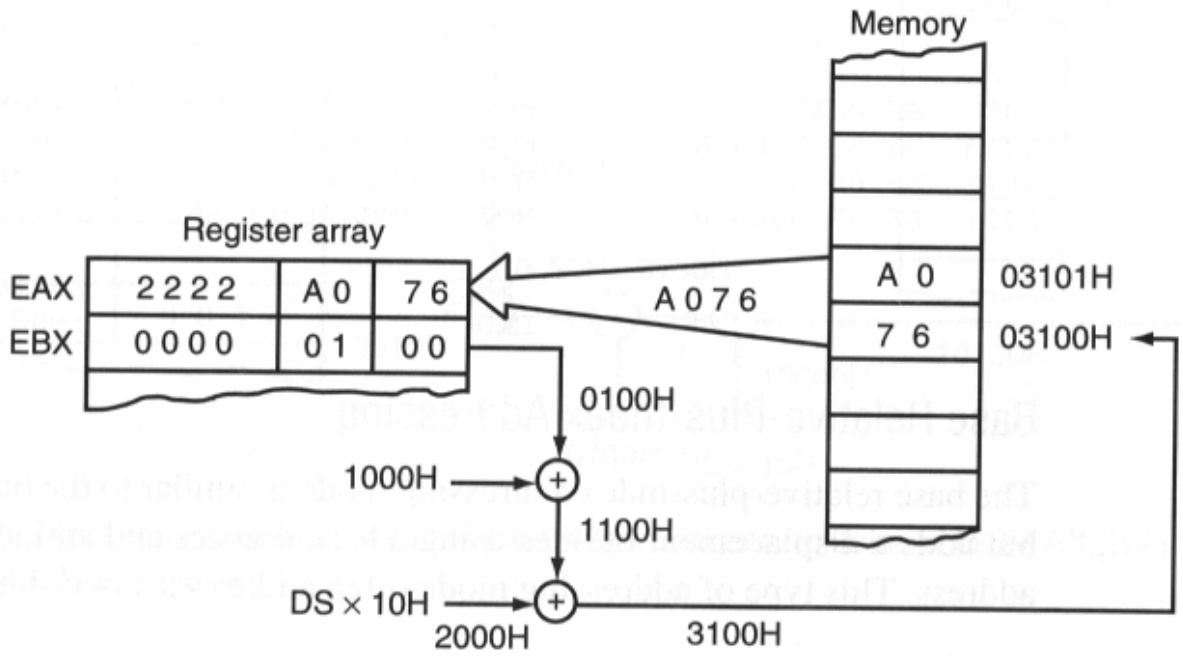
Assembly Language	*Size*	*Operation*
MOV CX,[BX+DI]	16-bits	Copies the word contents of the data segment memory location address by BX plus DI into CX
MOV CH,[BP+SI]	8-bits	Copies the byte contents of the stack segment memory location addressed by BP plus SI into CH
MOV [BX+SI],SP	16-bits	Copies SP into the data segment memory location addresses by BX plus SI
MOV [BP+DI],AH	8-bits	Copies AH into the stack segment memory location addressed by BP plus DI
MOV CL,[EDX+EDI]	8-bits	Copies the byte contents of the data segment memory location addressed by EDX plus EDI into CL
MOV [EAX+EBX],ECX	32-bits	Copies ECX into the data segment memory location addressed by EAX plus EBX

Pengalamatan Register Relatif

- Dalam pengalamatan register relatif data dalam segmen memori dialamatkan dengan menambahkan displacement pada isi register basis atau register index (BP, BX, DI, atau SI)
  - **Gambar 3.10** menggambarkan operasi instruksi MOV AX,[BX+ 1000H]

Pengalamatan Register Relatif (lanjutan)

FIGURE 3–10 The operation of the MOV AX,[BX+1000H] instruction, when BX = 0100H and DS = 0200H



Pengalamatan Register Relatif (lanjutan)

- Displacement dapat berupa bilangan yang ditambah dengan isi register dalam [ ], seperti instruksi MOV AL,[DI+2], atau dapat berupa displacement dikurangi dari isi register seperti dalam instruksi MOV AL,[SI-1].
  - **Gambar 3.11** menggambarkan pengalamatan register relatif digunakan untuk mengalamatkan elemen array.

Pengalamatan Register Relatif (lanjutan)

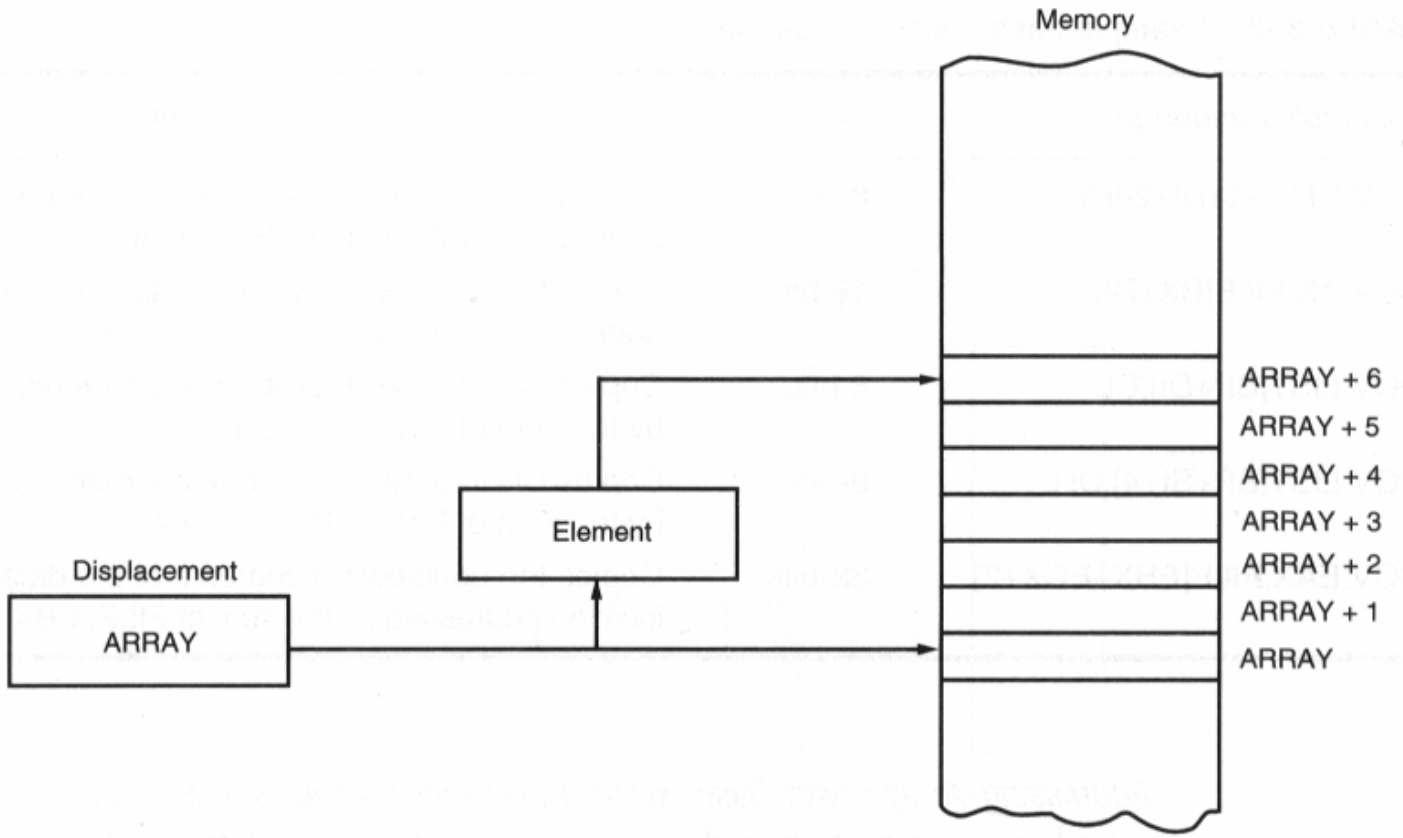


FIGURE 3–11 Register relative addressing used to address an element of ARRAY. The displacement addresses the start of ARRAY, and DI accesses an element.

Pengalamatan Base Relative-Plus-Index

- Tipe mode pengalamatan ini sering digunakan untuk menangani array dua dimensi dari data memori.
  - Mode pengalamatan ini paling jarang digunakan
  - **Gambar 3.12** menunjukkan bagaimana data ditunjukkan jika instruksi yang dieksekusi oleh mikroprosesor adalah MOV AX, [BX+SI+100H]
- Pengalamatan array dengan base relative-plus-index
  - Misalkan bahwa suatu file dari banyak record ada dalam memori dan tiap record itu berisi banyak elemen.
- **Contoh 3.9** dan **Gambar 3.13**

Pengalamatan Base Relative-Plus-Index (Lanjutan)

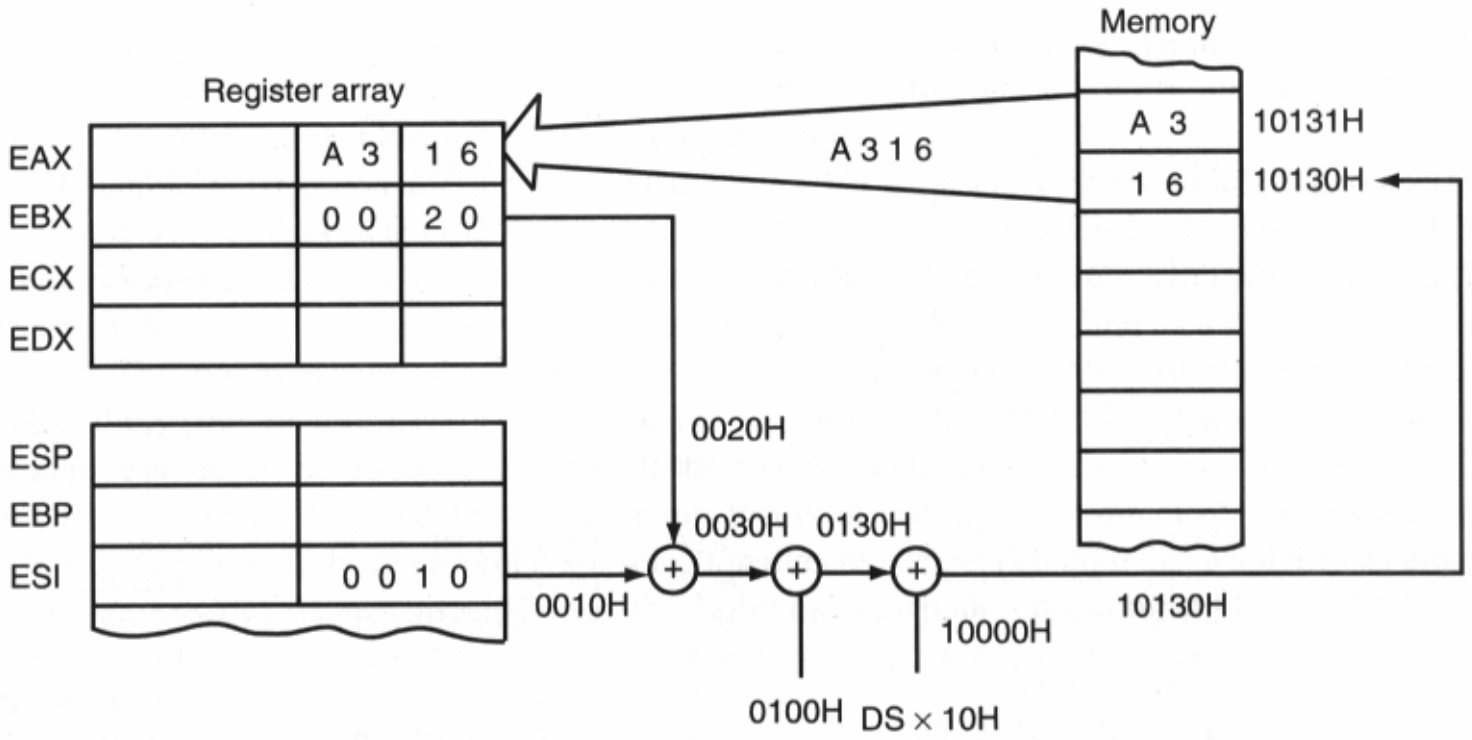


FIGURE 3–12 An example of base relative-plus-index addressing using a MOV AX,[BX+SI+100H] instruction. Note: DS = 1000H.

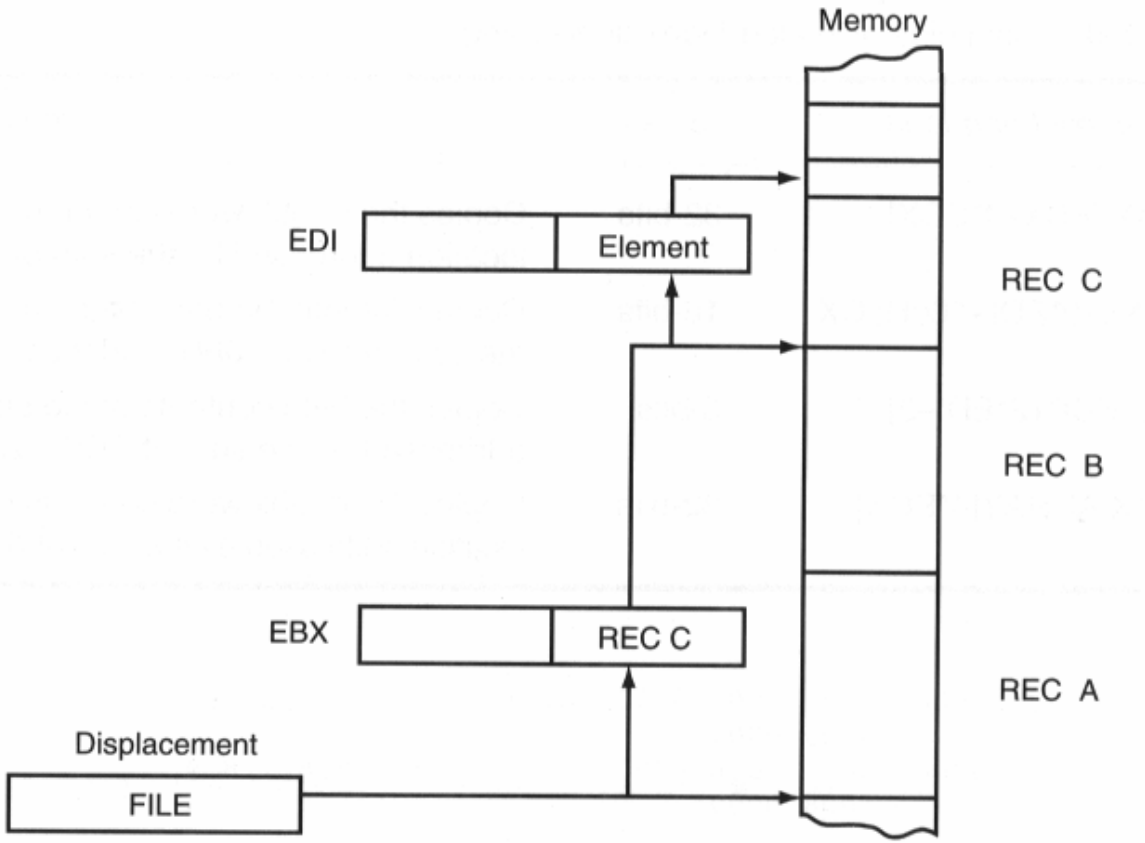
Pengalamatan Base Relative-Plus-Index (Lanjutan)

EXAMPLE 3–9

```
                        .MODEL SMALL                    ;SMALL model
0000                    .DATA                           ;start of DATA segment

0000  = 0000            FILE  EQU    THIS BYTE          ;assign FILE to this byte

0000  000A [            RECA  DB     10 DUP (?)         ;reserve 10 bytes for RECA
              00
                 ]
000A  000A [            RECB  DB     10 DUP (?)         ;reserve 10 bytes for RECB
              00
                 ]
0014  000A [            RECC  DB     10 DUP (?)         ;reserve 10 bytes for RECC
              00

001E  000A [            RECD  DB     10 DUP (?)         ;reserve 10 bytes for RECD
              00
                 ]

0000                          .CODE                     ;start of CODE segment
                              .STARTUP                  ;start of program

0017  BB 0000 R               MOV    BX,OFFSET RECA     ;address RECA
001A  BF 0000                 MOV    DI,0               ;address element 0
001D  8A 81 0000 R            MOV    AL,FILE[BX+DI]     ;get data
0021  BB 0014 R               MOV    BX,OFFSET RECC     ;address RECC
0024  BF 0002                 MOV    DI,2               ;address element 2
0027  88 81 0000 R            MOV    FILE[BX+DI],AL     ;save data

                              .EXIT                     ;exit to DOS
                              END                       ;end of file
```

Pengalamatan Base Relative-Plus-Index (Lanjutan)

FIGURE 3–13 Base relative-plus-index addressing used to access a FILE that contains multiple records (REC)



Pengalamatan Indeks-Berskala

- Mode pengalamatan ini adalah khusus untuk mikroprosesor 80386 – Pentium Pro.
  - Menggunakan dua register 32-bit (register basis dan register indeks) untuk mengakses memori.
  - Register kedua (indeks) dikalikan dengan faktor skala (1X, 2X, 4X, or 8X)
  - MOV AX,[EDI+2*ECX]
  - Lihat **contoh 3.10** dan **Tabel 3.9**

Pengalamatan Indeks-Berskala (Lanjutan)

EXAMPLE 3–10

```
                                .MODEL SMALL                ;select SMALL model
                                .386                        ;use the 80386
0000                            .DATA                       ;start of DATA segment

0000  0000 0001 0002  LIST      DW   0,1,2,3,4             ;define array list
      0003 0004
000A  0005 0006 0007            DW   5,6,7,8,9
      0008 0009

0000                            .CODE                       ;start of CODE segment
                                .STARTUP                    ;start of program
0010  66| BB 00000000 R         MOV  EBX,OFFSET LIST        ;address array LIST

0016  66| B9 00000002           MOV  ECX,2                  ;get element 2
001C  67& 8B 04 4B              MOV  AX,[EBX+2*ECX]

0020  66| B9 00000004           MOV  ECX,4                  ;store in element 4
0026  67& 89 04 4B              MOV  [EBX+2*ECX],AX

002A  66| B9 00000007           MOV  ECX,7                  ;store in element 7
0030  67& 89 04 4B              MOV  [EBX+2*ECX],AX

                                .EXIT                       ;exit to DOS
                                END                         ;end of file
```

Pengalamatan Indeks-Berskala (Lanjutan)

TABLE 3–9 Examples of scaled-index addressing

Assembly Language	*Size*	*Operation*
MOV EAX,[EBX+4*ECX]	32-bits	Copies the doubleword contents of the data segment memory location addressed by the sum of 4 times ECX plus EBX into EAX
MOV [EAX+2*EDI+100H],CX	16-bits	Copies CX into the data segment memory location addressed by the sum of EAX, 100H, and 2 times EDI
MOV AL,[EBP+2*EDI–2]	8-bits	Copies the byte contents of the stack segment memory location addressed by the sum of EBP, –2, and 2 times EDI into AL
MOV EAX,ARRAY[4*ECX]	32-bits	Copies the doubleword contents of the data segment memory location addressed by the sum of ARRAY plus 4 times ECX into EAX

Struktur Data

- Struktur data digunakan untuk menetapkan bagaimana informasi disimpan dalam array memori dan akan sangat berguna untuk aplikasi yang menggunakan array.
  - Awal dari struktur diidentifikasikan dengan direktif bahasa rakitan STRUC dan diakhiri dengan pernyataan ENDS
  - Lihat **contoh 3.11**
- Pada saat data dialamatkan dalam struktur, gunakan nama struktur dan nama field untuk memilih field yang ada dalam struktur **(Contoh 3.12)**

Struktur Data (Lanjutan)

EXAMPLE 3–11

```
                                ;Define INFO data structure
0057                            INFO    STRUC
0000   0020   [                 NAMES   DB     32 DUP (?)          ;32 bytes for name
                 00
                    ]
0020   0020   [                 STREET  DB     32 DUP (?)          ;32 bytes for street
                 00
                    ]
0040   0010   [                 CITY    DB     16 DUP (?)          ;16 bytes for city
                 00
                    ]
0050   0002   [                 STATE   DB      2 DUP (?)          ;2 bytes for state
                 00
                    ]
0052   0005   [                 ZIP     DB      5 DUP (?)          ;5 bytes for zip-code
                 00
                    ]
                                INFO    ENDS
0000  42 6F 62 20 53 6D        NAME1   INFO    <'Bob Smith','123 Main Street','Wanda','OH','44444'>
      69 74 68
      0017 [
              00
                 ]
      31 32 33 20 4D
      61 69 6E 20 53 74
      72 65 65 74
      0011 [
              00
                 ]
      57 61 6E 64 61
      000B [
              00
                 ]
      4F 48 34 34 34
      34 34

0057 53 74 65 76 65 20          NAME2   INFO <'Steve Doe','222 Mouse Lane','Miller','PA','18100'>
      44 6F 65
      0017 [
              00
                 ]
      32 32 32 20 4D
      6F 75 73 65 20 4C
      61 6E 65
      0012 [
              00
                 ]
      4D 69 6C 6C 65
      72
      000A [
              00
                 ]
      50 41 31 38 31
      30 30

00AE 42 65 6E 20 44 6F          NAME3   INFO <'Ben Dover','303 Main Street','Orender','CA','90000'>
      76 65 72
      0017 [
              00
                 ]
      33 30 33 20 4D
      61 69 6E 20 53 74
      72 65 65 74
      0011 [
              00
                 ]
      4F 72 65 6E 64
      65 72
      0009 [
              00
                 ]
      43 41 39 30 30
      30 30
```

Struktur Data (Lanjutan)

EXAMPLE 3–12

```
                        ;Clear names in array NAME1

0000  B9 0020                     MOV   CX,32
0003  B0 00                       MOV   AL,0
0005  BE 0000 R                   MOV   SI,OFFSET NAME1.NAMES
0008  F3/AA                       REP   STOSB

                        ;Clear street in array NAME2

000A  B9 0020                     MOV   CX,32
000D  B0 00                       MOV   AL,0
0010  BE 0077 R                   MOV   SI,OFFSET NAME2.STREET
0013  F3/AA                       REP   STOSB

                      ;Clear zip-code in array NAME3

0015  B9 0005                     MOV   CX,5
0018  B0 00                       MOV   AL,0
001A  BE 0100 R                   MOV   SI,OFFSET NAME3.ZIP
001D  F3/AA                       REP   STOSB
```

Mode Pengalamatan Memori Program

- Mode pengalamatan memori program (JMP dan CALL) terdiri dari tiga bentuk yang berbeda : direct, relative, and indirect
- Pengalamatan Memori Program Langsung
  - Instruksi-instruksi untuk pengalamatan memori program langsung menyimpan alamat dengan opcode.
  - Lihat **Gambar 3.14**
  - Loncatan langsung sering disebut loncatan jauh karena dapat meloncat ke setiap lokasi memori untuk instruksi berikutnya.

Mode Pengalamatan Memori Program (lanjutan)

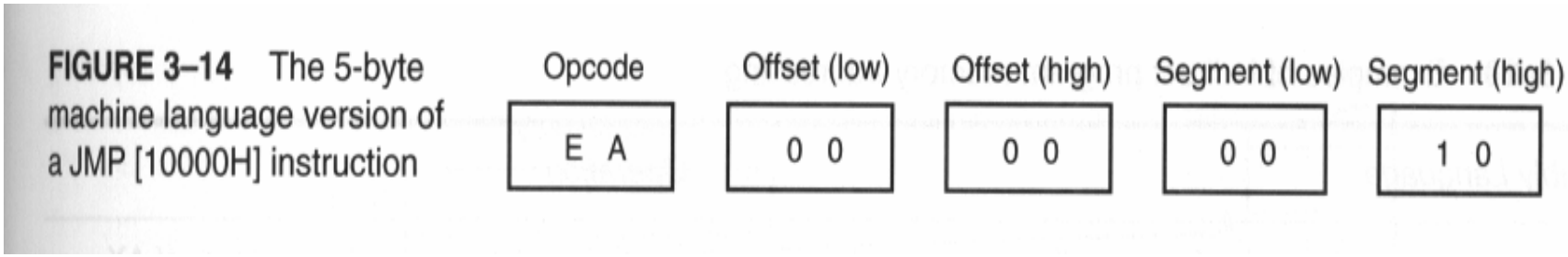


FIGURE 3–14 The 5-byte machine language version of a JMP [10000H] instruction

Mode Pengalamatan Memori Program (lanjutan)

- Pengalamatan Memori Program Relatif
  - Istilah relatif sebenarnya berarti “relatif terhadap pointer instruction (IP)”
  - Lihat **Gambar 3.15**
  - Instruksi JMP adalah instruksi 1-byte, dengan displacement 1-byte atau 2-byte yang ditambahkan ke penunjuk instruksi.
  - Instruksi JMP dan CALL relatif berisi baik displacement 8-bit dan 16-bit.

Mode Pengalamatan Memori Program (lanjutan)

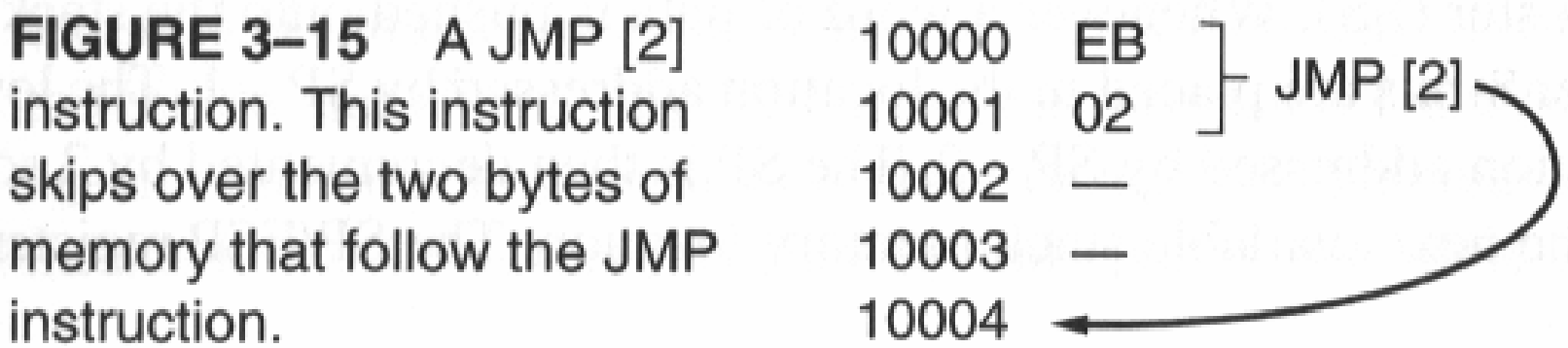


FIGURE 3–15 A JMP [2] instruction. This instruction skips over the two bytes of memory that follow the JMP instruction.

Mode Pengalamatan Memori Program (lanjutan)

- Pengalamatan Memori Program Tidak Langsung
  - **Tabel 3.10** mencatat beberapa instruksi jump tidak langsung dalam program yang bisa diterima yang menggunakan semua register 16-bit, semua register relatif, dan semua register relatif dengan displacement.
  - Jika register 16-bit menyimpan alamat instruksi JMP, maka loncatnya dekat.

Mode Pengalamatan Memori Program (lanjutan)

TABLE 3–10 Examples of indirect program memory addressing

Assembly Language	*Operation*
JMP AX	Jumps to the current code segment location addressed by the contents of AX
JMP CX	Jumps to the current code segment location addressed by the contents of CX
JMP NEAR PTR [BX]	Jumps to the current code segment location addressed by the contents of the data segment memory location addressed by BX
JMP NEAR PTR[DI+2]	Jumps to the current code segment location addressed by the contents of the data segment memory location addressed by DI plus 2
JMP TABLE[BX]	Jumps to the current code segment location addressed by the contents of the data segment memory location addressed by TABLE plus BX
JMP ECX	Jumps to the current code segment location addressed by the contents of ECX

Mode Pengalamatan Memori Program (lanjutan)

- Jika register relatif menyimpan alamat, loncatnya juga dapat dipertimbangkan sebagai loncatan tidak langsung.
- **Gambar 3.16** menunjukkan suatu tabel loncat yang disimpan, mulai pada lokasi memori TABLE.

Mode Pengalamatan Memori Program (lanjutan)

FIGURE 3–16 A jump table that stores addresses of various programs. The exact address chosen from the TABLE is determined by an index stored with the jump instruction.

```
TABLE   DW   LOC0
        DW   LOC1
        DW   LOC2
        DW   LOC3
```

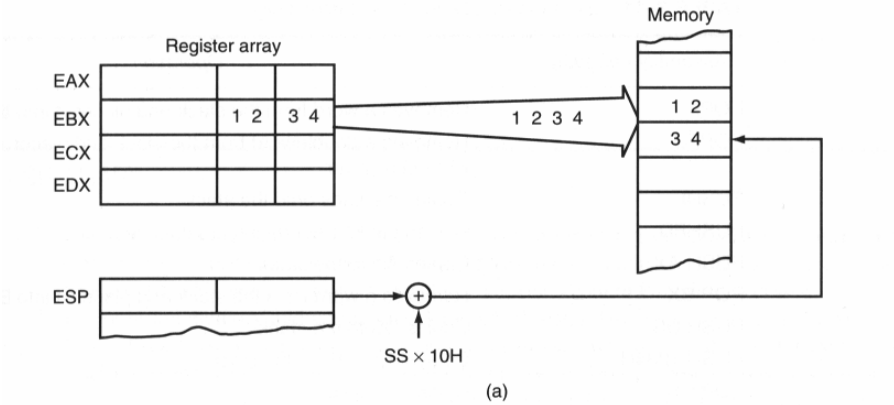
Mode Pengalamatan Memori Stack

- Stack menampung data sementara dan menyimpan alamat untuk kembali ke prosedur.
  - Memori stack adalah memori LIFO.
  - Data ditempatkan pada stack dengan instruksi PUSH.
  - Dihapus dengan instruksi POP.

Mode Pengalamatan Memori Stack (Lanjutan)

- Memori stack dipelihara oleh dua register : SP atau ESP, dan SS
  - Lihat **Gambar 3.17**
  - Instruksi PUSHA dan POPA merupakan instruksi memasukkan atau mengambil semua isi register, kecuali register segmen dalam stack **(lihat contoh 3.14)**

Mode Pengalamatan Memori Stack (Lanjutan)



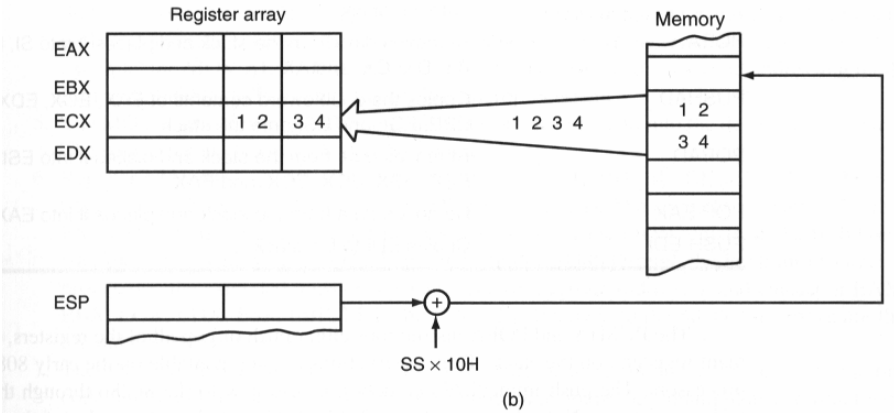


FIGURE 3–17 The PUSH and POP instructions. (a) PUSH BX places the contents of BX onto the stack, (b) POP CX removes data from the stack and places them into CX. Both instructions are shown after execution.

Mode Pengalamatan Memori Stack (Lanjutan)

EXAMPLE 3–14

```
                        .MODEL TINY             ;select TINY model
0000                    .CODE                   ;start CODE segment
                        .STARTUP                ;start of program
0100  B8 1000           MOV    AX,1000H         ;load test data
0103  BB 2000           MOV    BX,2000H
0106  B9 3000           MOV    CX,3000H

0109  50                PUSH   AX               ;1000H to stack
010A  53                PUSH   BX               ;2000H to stack
010B  51                PUSH   CX               ;3000H to stack

010C  58                POP    AX               ;3000H to AX
010D  59                POP    CX               ;2000H to CX


010E  5B                POP    BX               ;1000H to BX
                        .EXIT                   ;exit to DOS
                        END                     ;end of file
```

EXAMPLE 3–7

```
                                .MODEL SMALL                ;select SMALL model
0000                            .DATA                       ;start of DATA segment

0000  0010 [        ARRAY   DB      16 DUP (?)              ;setup ARRAY
              00
                    ]
0010  29                    DB      29H                     ;sample data at element 10H
0011  001E [                DB      30 DUP (?)
              00
                    ]
0000                            .CODE                       ;start of CODE segment
                                .STARTUP                    ;start of program

0017  BB 0000 R                 MOV     BX,OFFSET ARRAY     ;address ARRAY
001A  BF 0010                   MOV     DI,10H              ;address element 10H
001D  8A 01                     MOV     AL,[BX+DI]          ;get element 10H
001F  BF 0020                   MOV     DI,20H              ;address element 20H
0022  88 01                     MOV     [BX+DI],AL          ;save in element 20H

                                .EXIT                       ;exit to DOS
                                END                         ;end of file
```

EXAMPLE 3–8

```
                                .MODEL SMALL            ;select SMALL model
0000                            .DATA                   ;start of DATA segment

0000  0010 [        ARRAY   DB    16 DUP (?)        ;setup ARRAY
              00
                    ]
0010  29                    DB    29H               ;sample data at element 10H
0011  001E [                DB    30 DUP (?)
              00
                    ]
0000                            .CODE                   ;start of CODE segment
                                .STARTUP                ;start of program

0017  BF 0010                   MOV   DI,10H            ;address element 10H
001A  8A 85 0000 R              MOV   AL,ARRAY[DI]      ;get element 10H
001E  BF 0020                   MOV   DI,20H            ;address element 20H
0021  88 85 0000 R              MOV   ARRAY[DI],AL      ;save in element 20H

                                .EXIT                   ;exit to DOS
                                END                     ;end of file
```