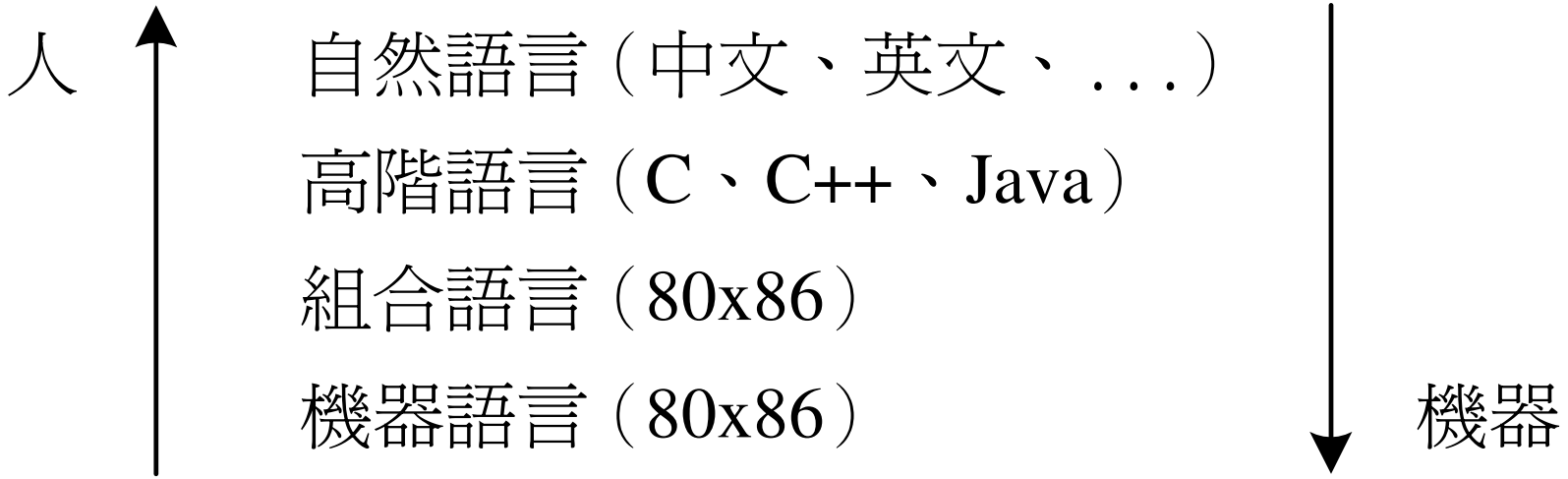
本章目標

- 了解電腦的基本功能與原理
- 了解組譯程式與組合語言程式
- 了解組合語言程式的建立與執行
- 了解基本的組譯程式假指令
- 了解組譯程式如何組譯組合語言程式

基本程式設計觀念



程式設計層次關係圖

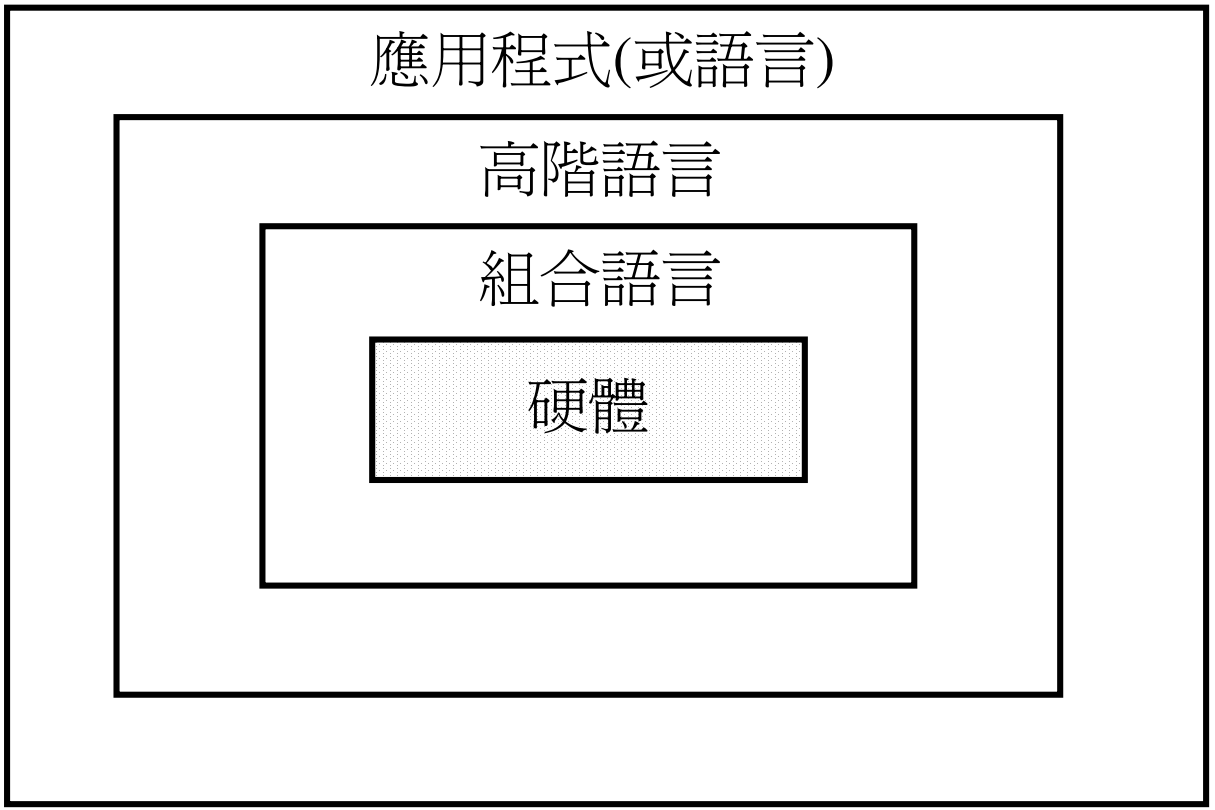
```
#include "stdio.h"
void main()
{
   int s,i;
   s = 0;
   for (i=1;i <= 10; i++)
       s = s + i;
   printf("%d\n",s);
}
```

(a) c 語言程式

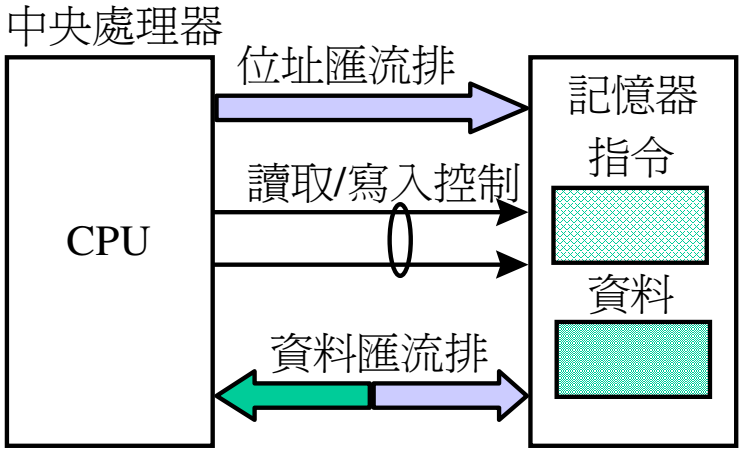
```
0000  55                MAIN:    PUSH  BP
0001  8B EC                      MOV   BP,SP
0003  83 EC 04                   SUB   SP,4
0006  C7 46 FE 0000              MOV   WORD PTR [BP-2],0
000B  C7 46 FC 0001              MOV   WORD PTR [BP-4],1
0010  8B 46 FC           $F11:   MOV   AX,[BP-4]
0013  01 46 FE                   ADD   [BP-2],AX
0016  FF 46 FC                   INC   WORD PTR [BP-4]
0019  83 7E FC 0A                CMP   WORD PTR [BP-4],10
001D  7E F1                      JLE   $F11
001F  FF 76 FE                   PUSH  WORD PTR [BP-2]
0022  9A ---- 0000 E             CALL  _PRINTF
0027  8B E5                      MOV   SP,BP
0029  5D                         POP   BP
002A  C3                         RET
```

(b) 機器語言程式　　(c) 組合語言程式

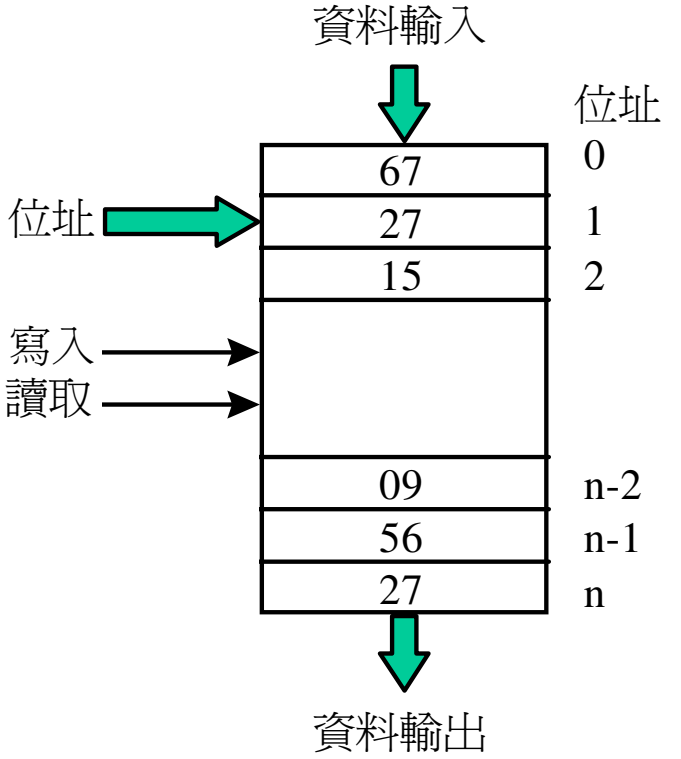
電腦的階層式結構



電腦與記憶器的邏輯結構



(a) 電腦的邏輯結構

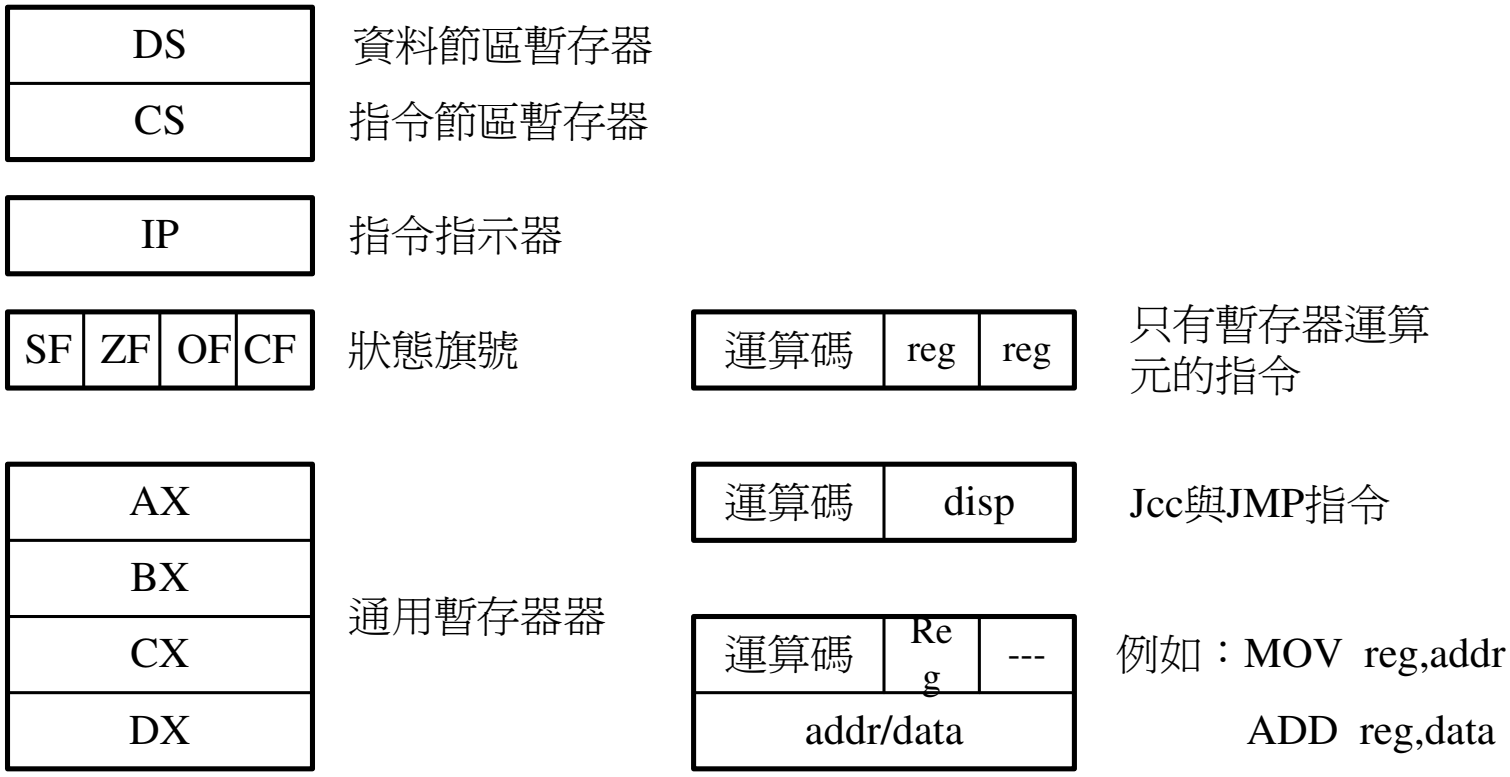


(b) 記憶器的邏輯結構

CPU的動作

```
CPU    模組
        IP  ：= 0 ；
        重複執行下列動作
        自記憶器位址爲 IP 的位置中摘取指令；
        執行該指令；
        IP  ：= IP + 1
END    CPU    模組
```

簡化的80x86微處理器規劃模式與指令格式



(a) 規劃模式　　(b) 指令格式

80x86最常用的資料轉移指令

指令	RTL 描述	說明
MOV reg,data	reg ← data	轉移立即資料(data)到暫存器 reg 內
MOV dreg,sreg	dreg ← sreg	轉移暫存器 sreg 的內容到暫存器 dreg
MOV segreg,reg	segreg ← reg	轉移暫存器 reg 的內容到暫存器 segreg
MOV reg,addr	reg ← Mem[DS×16+addr]	轉移記憶器(addr)的內容到暫存器 reg
MOV addr,reg	Mem[DS×16+addr]←reg	儲存暫存器 reg 的內容到記憶器(addr)中

80x86最常用的算術運算指令

指令	RTL 描述	說明
ADD reg,data	reg ← reg + data	立即資料(data)與 reg 相加後存回 reg 內
ADD dreg,sreg	dreg ← dreg + sreg	暫存器 dreg 與 sreg 相加後存回 dreg 內
ADC reg,data	reg ← reg + data+ C	立即資料(data)與 reg 及進位相加後存回 reg 內
ADC dreg,sreg	dreg ← dreg + sreg + C	暫存器 dreg 與 sreg 及進位相加後存回 dreg 內
SUB reg,data	reg ← reg – data	暫存器 reg 減去立即資料(data)後存回 reg 內
SUB dreg,sreg	dreg ← dreg – sreg	暫存器 dreg 減去 sreg 後存回 dreg 內
SBB reg,data	reg ← reg – data – C	暫存器 reg 減去立即資料(data)及進位後存回 reg
SBB dreg,sreg	dreg ← dreg – sreg – C	暫存器 dreg 減去 sreg 及進位後存回 dreg

80x86最常用的邏輯運算指令

指令	RTL 描述	說明
AND reg,data	reg ← reg ∧ data	暫存器 reg 與立即資料(data) AND 後存回 reg 內
AND dreg,sreg	dreg ← dreg ∧ sreg	暫存器 dreg 與 sreg AND 後存回 dreg 內
OR reg,data	reg ← dreg ∨ data	暫存器 reg 與立即資料(data) OR 後存回 reg 內
OR dreg,sreg	dreg ← dreg ∨ sreg	暫存器 dreg 與 sreg OR 後存回 dreg 內
XOR reg,data	reg ← reg ⊕ data	暫存器 reg 與立即資料(data) XOR 後存回 reg 內
XOR dreg,sreg	dreg ← dreg ⊕ sreg	暫存器 dreg 與 sreg XOR 後存回 dreg 內
NOT reg	reg ← $\overline{\text{reg}}$	暫存器 reg 內容取 1 補數後存回 reg 內

80x86最常用的分歧與跳躍指令

指令	RTL 描述	說明
JC disp	C ： IP ← IP + disp(2 補數)	當進位旗號爲 1 時分歧到標的位址
JNC disp	$\overline{C}$ ： IP ← IP + disp(2 補數)	當進位旗號爲 0 時分歧到標的位址
JZ disp	Z ： IP ← IP + disp(2 補數)	當零旗號爲 1 時分歧到標的位址
JNZ disp	$\overline{Z}$ ： IP ← IP + disp(2 補數)	當零旗號爲 0 時分歧到標的位址
JMP disp	IP ← IP + disp(2 補數)	無條件性分歧到標的位址

CPU的動作

CPU　模組

IP　：= 0　；

重複執行下列動作

自記憶器位址爲 IP 的位置中摘取指令；

執行指令解碼；

若該指令執行時需要資料，則自記憶器中讀取運算元；

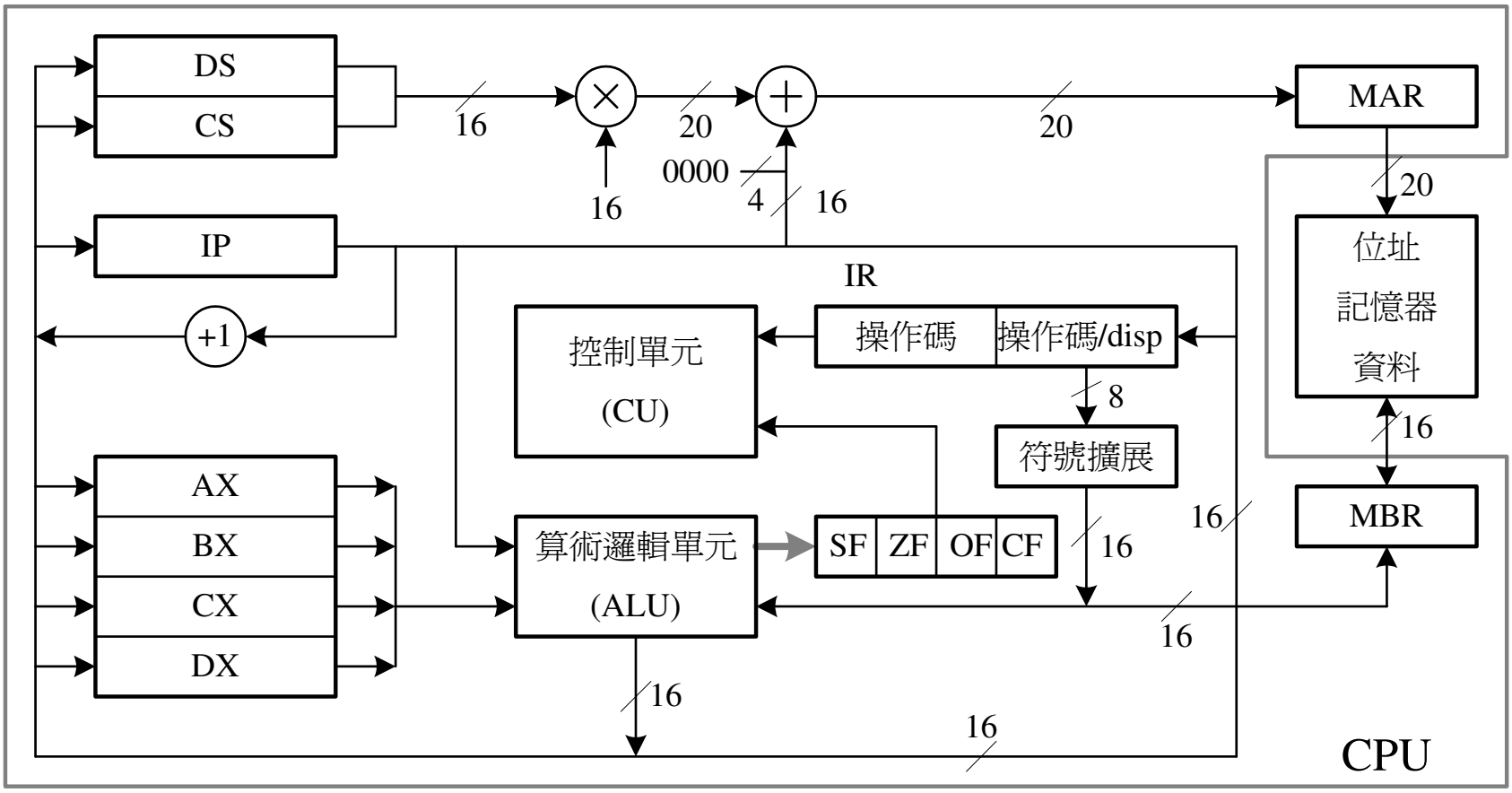
執行指令的動作；

若該指令需要儲存結果，則儲存結果於記憶器中；

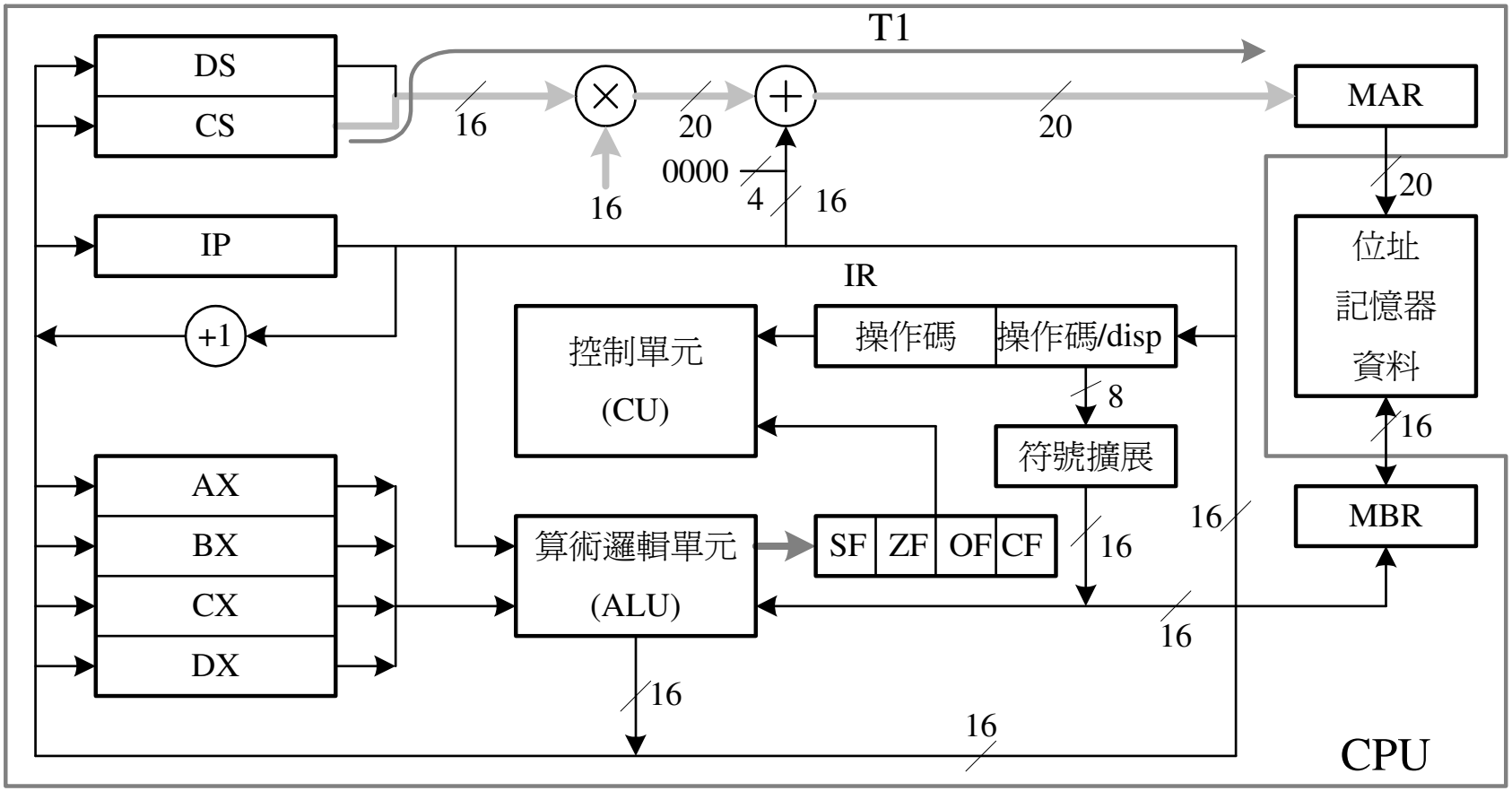
IP　：= IP + 1

END　CPU　模組

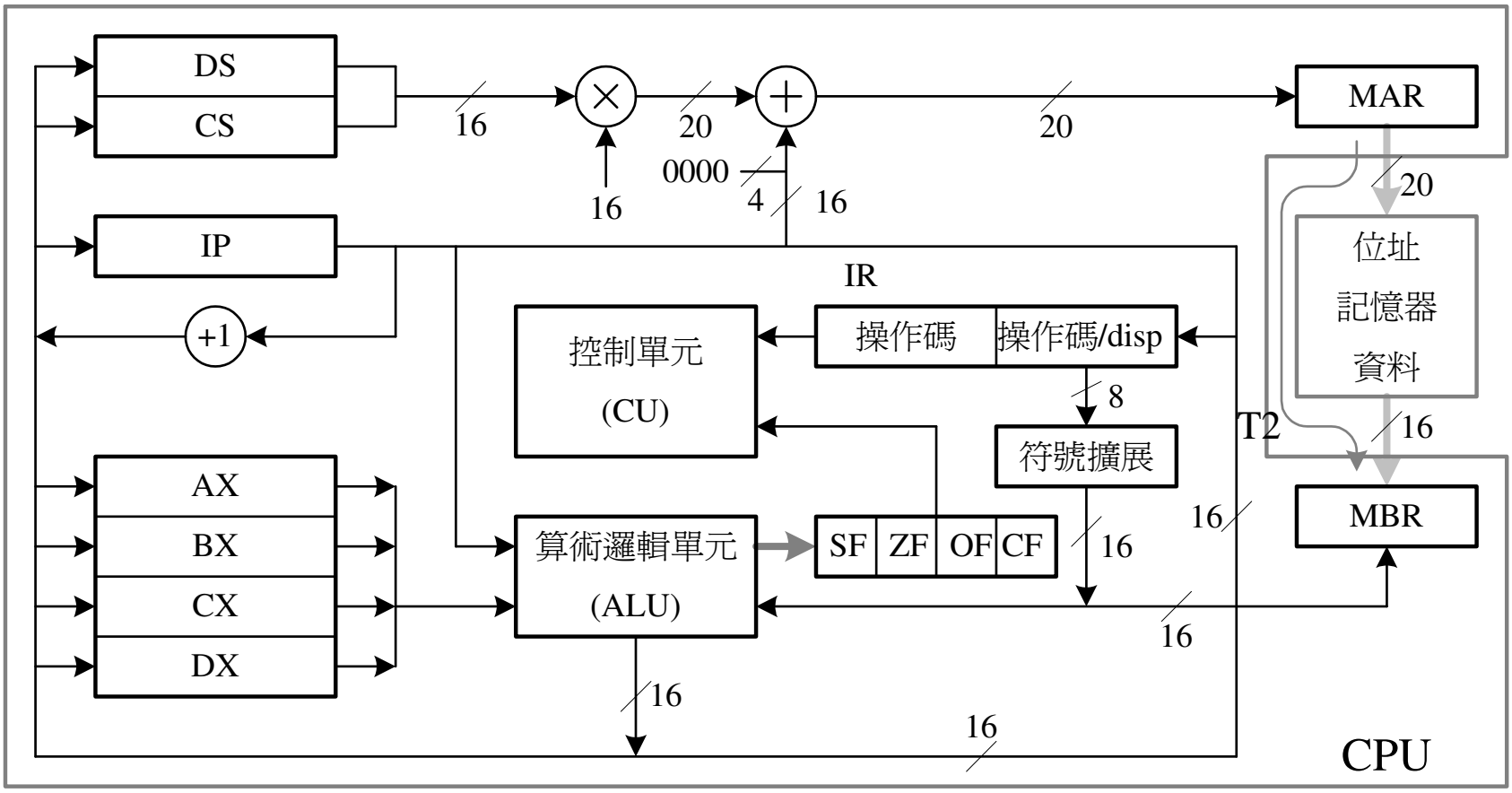
一個簡化的80x86 CPU RTL模型



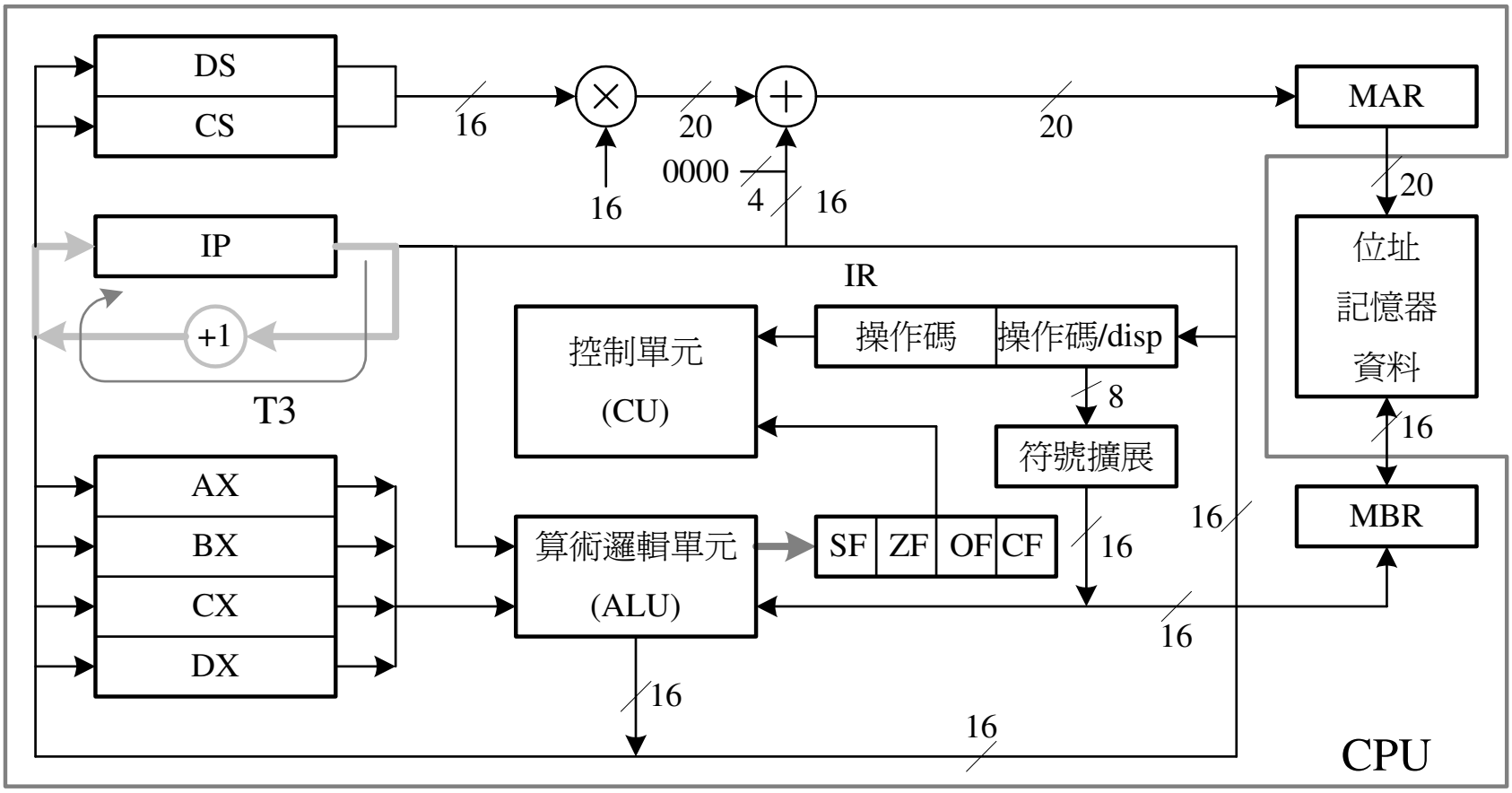
指令讀取的第一個步驟


T1
DS
CS
16
20
20
MAR
0000
4
16
16
20
IP
位址
記憶器
資料
IR
+1
控制單元
(CU)
操作碼
操作碼/disp
8
符號擴展
16
AX
BX
CX
DX
算術邏輯單元
(ALU)
SF
ZF
OF
CF
16
16
MBR
16
16
16
CPU

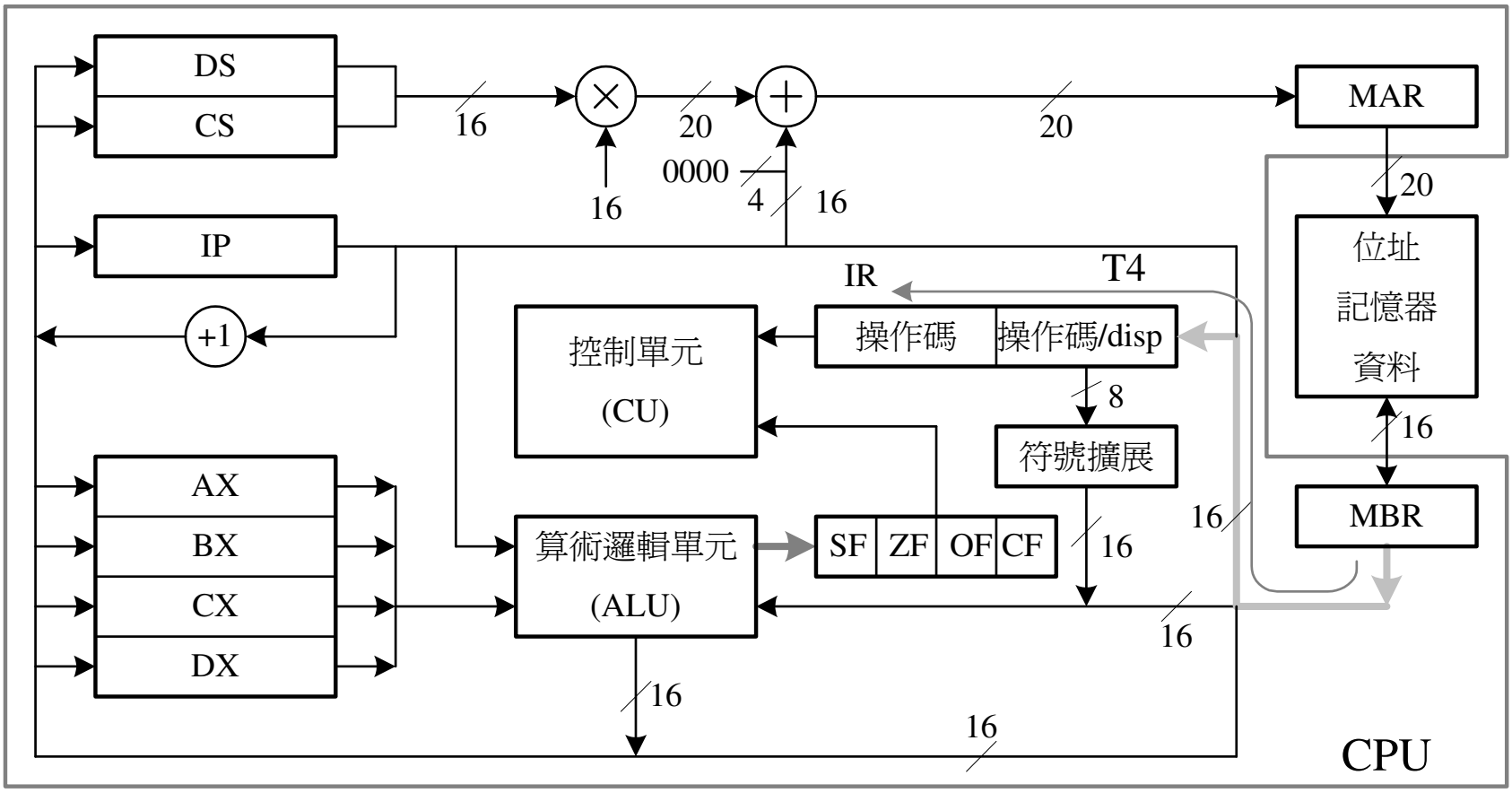
指令讀取的第二個步驟


DS
CS
16
20
20
MAR
0000
4
16
16
20
IP
IR
位址
記憶器
資料
+1
控制單元
(CU)
操作碼
操作碼/disp
8
T2
16
符號擴展
AX
BX
CX
DX
算術邏輯單元
(ALU)
SF
ZF
OF
CF
16
16
MBR
16
16
16
CPU

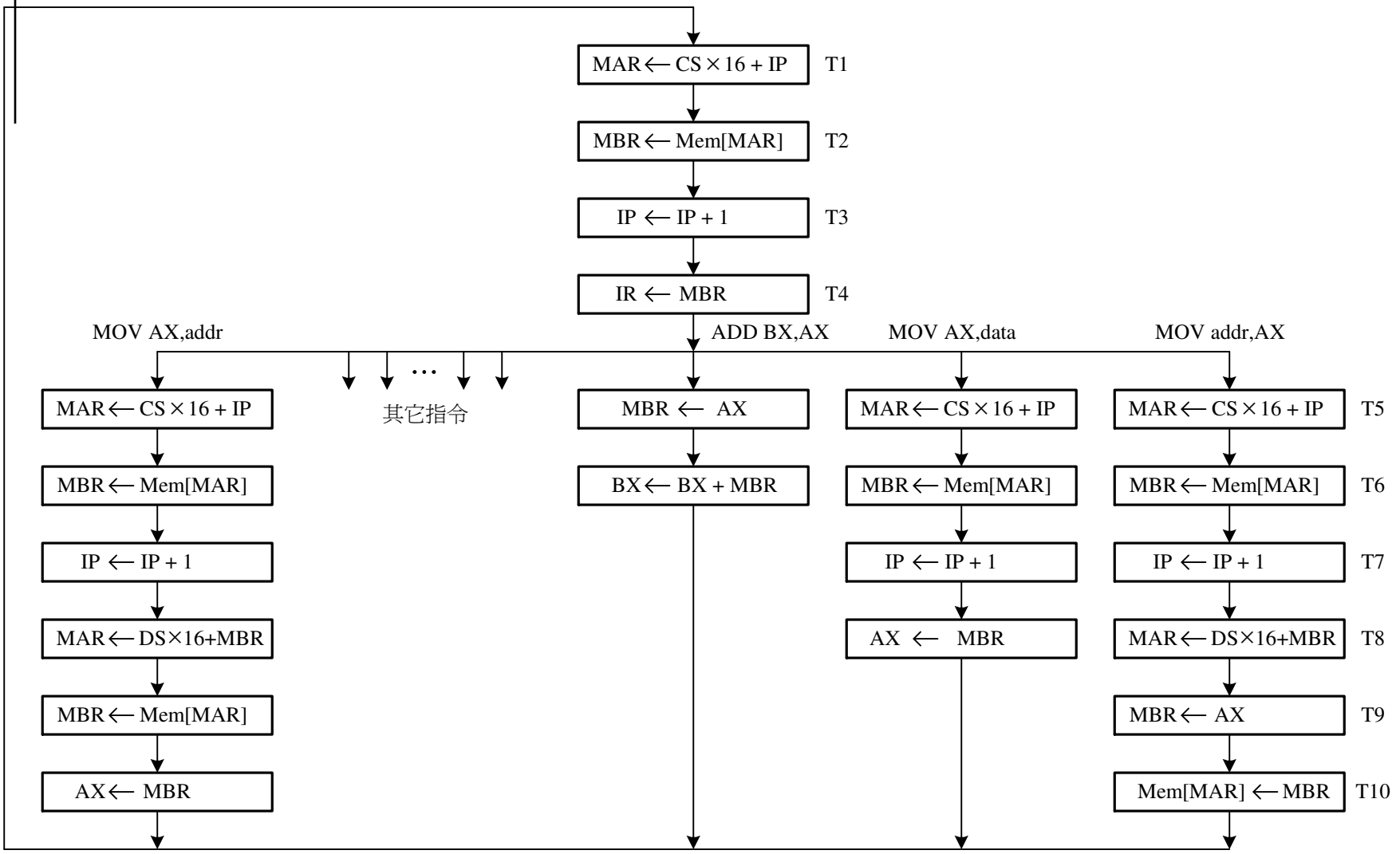
指令讀取的第三個步驟


DS
CS
16
20
20
MAR
0000
4
16
16
20
IP
IR
位址
記憶器
資料
+1
T3
控制單元
(CU)
操作碼
操作碼/disp
8
符號擴展
16
16
MBR
AX
BX
CX
DX
算術邏輯單元
(ALU)
SF
ZF
OF
CF
16
16
16
CPU

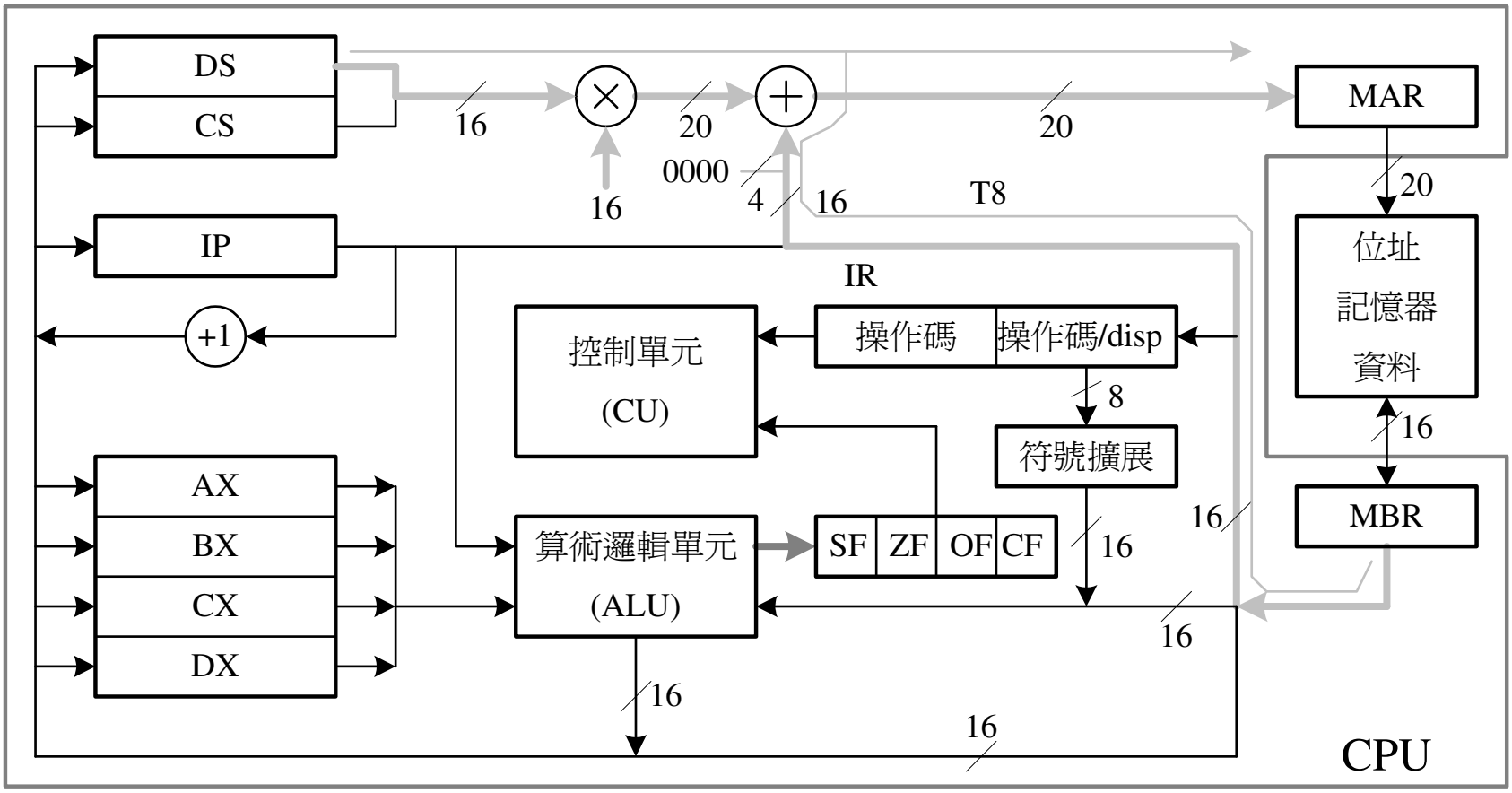
指令讀取的第四個步驟


DS
CS
16
20
20
MAR
0000
4
16
16
20
IP
位址
記憶器
資料
IR
T4
+1
控制單元
(CU)
操作碼
操作碼/disp
8
符號擴展
16
AX
BX
CX
DX
算術邏輯單元
(ALU)
SF
ZF
OF
CF
16
16
MBR
16
16
16
CPU

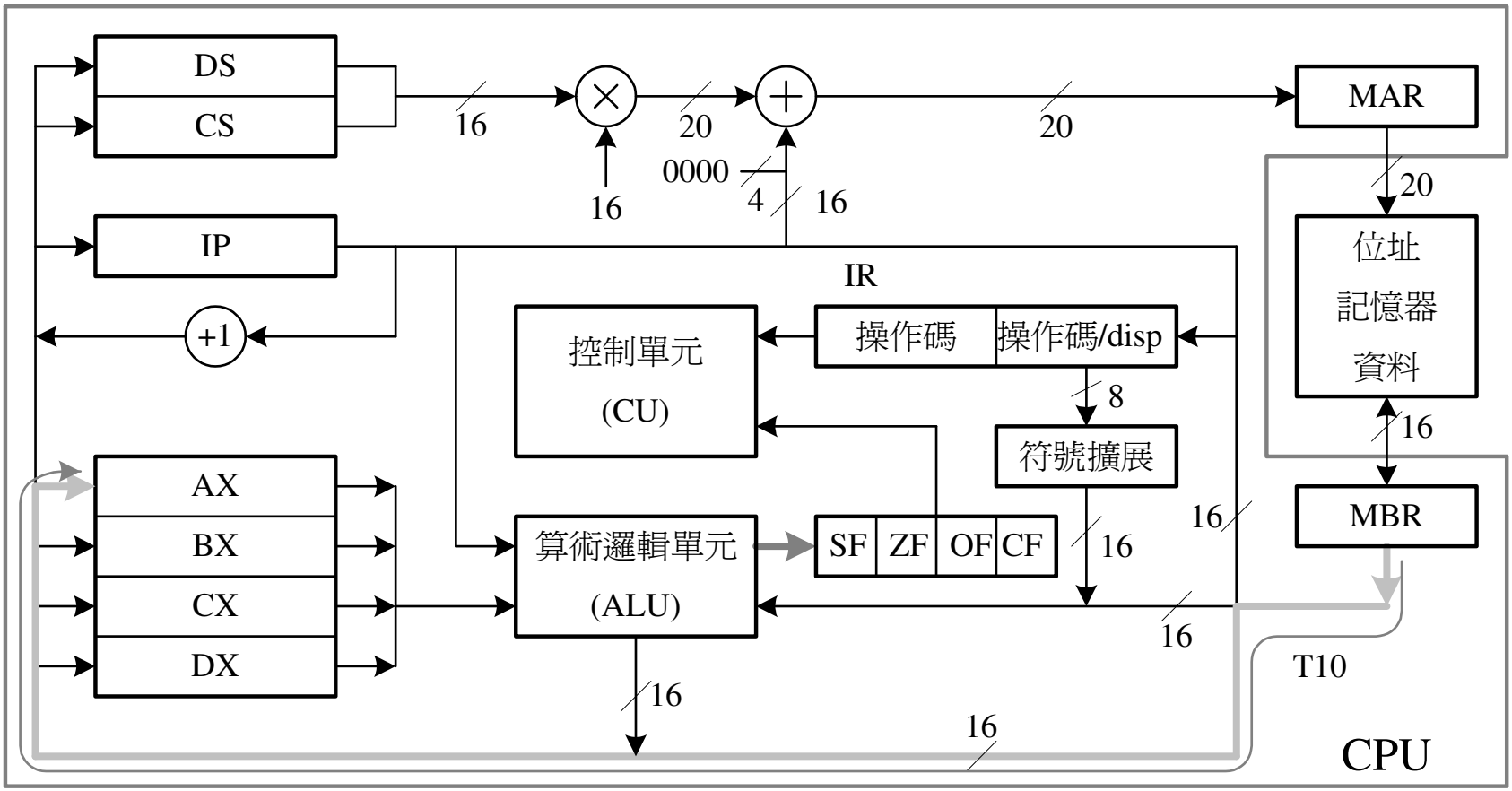
指令的讀取與執行動作時序圖



指令MOV AX,addr執行的第四個步驟



指令MOV AX,addr執行的最後一個步驟



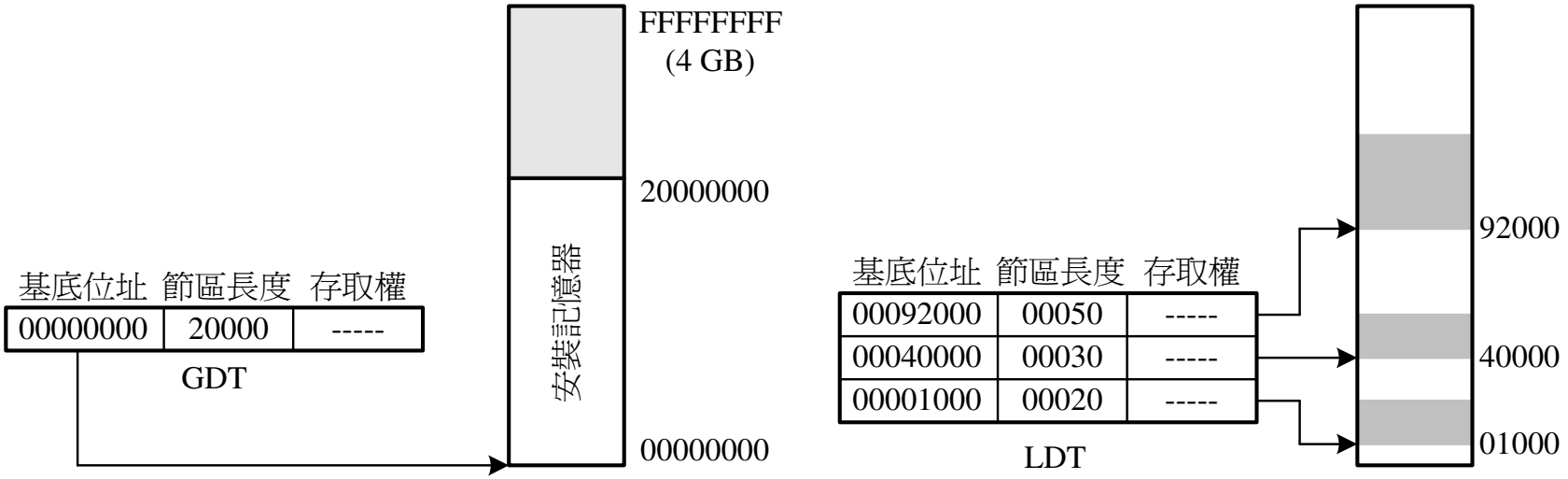
典型的組合語言程式列表

```
                        ;program 2.2-1 (file_name test.asm)
                              page  45,70
0000                      DATA      SEGMENT PUBLIC 'DATA'
0000 0047                   OPR1      DW    0047H
0002 0023                   OPR2      DW    0023H
0004                      DATA      ENDS
                        ;Exchange two words in memory
                        ;using absolute addressing mode
0000                      CODE      SEGMENT PUBLIC 'CODE'
                        ASSUME    CS:CODE,DS:DATA
0000                      SWAP      PROC  NEAR
0000  B8 ---- R                     MOV   AX,DATA ;load DS
0003  8E D8                         MOV   DS,AX
0005  A1 0000 R                       MOV   AX,OPR1 ;get opr1
0008  8B 1E 0002 R                    MOV   BX,OPR2 ;get opr2
000C  A3 0002 R                       MOV   OPR2,AX ;save opr1
000F  89 1E 0000 R                    MOV   OPR1,BX ;save opr2
0013  C3                            RET
0014                      SWAP      ENDP
0014                      CODE      ENDS
                              END   SWAP
```

最常用的部分假指令

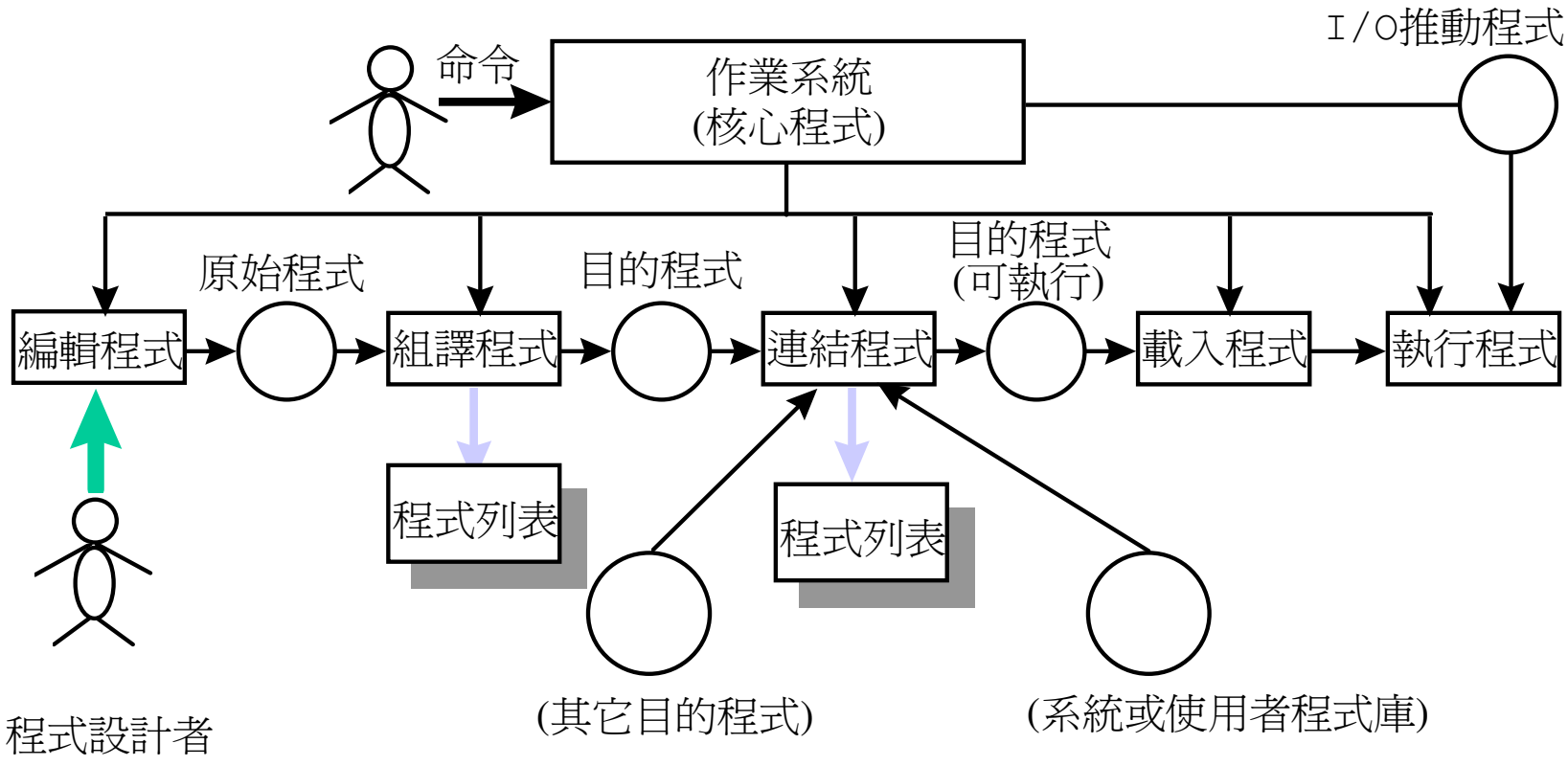
假指令	意義	例子
DB <exp>[,<exp>,...] DB <string>[,<string>,...]	}定義位元組資料	DB 07,0EFH DB 'AB','BOOK'
DW <exp>[,<exp>,...]	定義語句(2 位元組)資料	DW 07,0E23FH
DD <exp>[,<exp>,...]	定義雙語句(4 位元組)資料	DD 0E23F0011H
DB number DUP(?)	保留位元組儲存空間	DB 5 DUP(0)
DW number DUP(?)	保留語句儲存空間	DW 50 DUP(0)
DD number DUP(?)	保留雙語句儲存空間	DD 25 DUP(0)
DQ number DUP(?)	保留四語句儲存空間	DQ 25 DUP(0)
ORG <exp>	定義機器碼起始位址	ORG 0100H
<name> EQU <exp>	指定 name 的值爲 exp	THREE EQU 3
END [<label>]	表程式到此結束，並且設定程式的啓始位址爲 label (若有)	END END START

保護模式的兩種記憶器映成模式



(a) 平坦模式　　(b) 多節區模式

產生與執行一個組合語言程式的過程



DOS模式組合語言程式的建立與測試

步驟1

```
C:>masm test test test
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993.  All rights reserved.
Invoking: ML.EXE /I. /Zm /c /Fl test.asm
Assembling: test.asm
```

DOS模式組合語言程式的建立與測試

步驟2

```
C:>link test test   或
C:>link
Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Object Modules [.obj]:test     (輸入欲連結的目的程式檔名)
Run File [test.exe]:test       (輸入欲產生的目的程式檔名)
List File [nul.map]:           (輸入enter)
Libraries [.lib]:              (輸入enter)
Definitions File [nul.def]:    (輸入enter)
LINK : warning L4021: no stack segment
```

DOS模式組合語言程式的建立與測試

步驟3

C:>debug test.exe

```
-u
11B0:0000 B8AF11      MOV     AX,11AF
11B0:0003 8ED8        MOV     DS,AX
11B0:0005 A10000      MOV     AX,[0000]
11B0:0008 8B1E0200    MOV     BX,[0002]
11B0:000C A30200      MOV     [0002],AX
11B0:000F 891E0000    MOV     [0000],BX
11B0:0013 C3          RET
11B0:0014 33DB        XOR     BX,BX
11B0:0016 8CC8        MOV     AX,CS
11B0:0018 83E007      AND     AX,+07
11B0:001B 8E060A00    MOV     ES,[000A]
11B0:001F 0F          DB 0F
```

DOS模式組合語言程式的建立與測試

```
-d 11AF:0
11AF:0000  47 00 23 00 00 00 00 00-00 00 00 00 00 00 00 00   G.#.............
11AF:0010  B8 AF 11 8E D8 A1 00 00-8B 1E 02 00 A3 02 00 89   ................
11AF:0020  1E 00 00 C3 33 DB 8C C8-83 E0 07 8E 06 0A 00 0F   ....3...........
11AF:0030  03 0E 0A 00 41 D1 E9 80-3E 84 25 00 74 37 26 39   ....A...>.%.t7&9
11AF:0040  17 75 28 53 52 50 E8 75-FC 72 18 F7 C3 F0 FF 75   .u(SRP.u.r.....u
11AF:0050  0D 0B D3 74 09 5A 52 B8-01 01 CD 31 73 05 58 50   ...t.ZR....1s.XP
11AF:0060  E8 55 FF 58 5A 5B 26 C7-07 00 00 83 C0 08 83 C3   .U.XZ[&.........
11AF:0070  02 E2 CB EB 12 26 39 17-75 05 50 E8 3A FF 58 83   .....&9.u.P.:.X.
```

DOS模式組合語言程式的建立與測試

```
-e 11af:0
11AF:0000  47.12


-d 11af:0
11AF:0000  12 00 23 00 00 00 00 00-00 00 00 00 00 00 00 00   ..#.............
11AF:0010  B8 AF 11 8E D8 A1 00 00-8B 1E 02 00 A3 02 00 89   ................
11AF:0020  1E 00 00 C3 33 DB 8C C8-83 E0 07 8E 06 0A 00 0F   ....3...........
11AF:0030  03 0E 0A 00 41 D1 E9 80-3E 84 25 00 74 37 26 39   ....A...>.%.t7&9
11AF:0040  17 75 28 53 52 50 E8 75-FC 72 18 F7 C3 F0 FF 75   .u(SRP.u.r.....u
11AF:0050  0D 0B D3 74 09 5A 52 B8-01 01 CD 31 73 05 58 50   ...t.ZR....1s.XP
11AF:0060  E8 55 FF 58 5A 5B 26 C7-07 00 00 83 C0 08 83 C3   .U.XZ[&.........
11AF:0070  02 E2 CB EB 12 26 39 17-75 05 50 E8 3A FF 58 83   .....&9.u.P.:.X.
```

DOS模式組合語言程式的建立與測試

```
-g=0,13
AX=0012  BX=0023  CX=0024  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=11AF  ES=119F  SS=11AF  CS=11B0  IP=0013   NV UP EI PL NZ NA PO NC
11B0:0013 C3          RET
```

```
-d 11af:0
11AF:0000  23 00 12 00 00 00 00 00-00 00 00 00 00 00 00 00   #...............
11AF:0010  B8 AF 11 8E D8 A1 00 00-8B 1E 02 00 A3 02 00 89   ................
11AF:0020  1E 00 00 C3 33 DB 8C C8-83 E0 07 8E 06 0A 00 0F   ....3...........
11AF:0030  03 0E 0A 00 41 D1 E9 80-3E 84 25 00 74 37 26 39   ....A...>.%.t7&9
11AF:0040  17 75 28 53 52 50 E8 75-FC 72 18 F7 C3 F0 FF 75   .u(SRP.u.r.....u
11AF:0050  0D 0B D3 74 09 5A 52 B8-01 01 CD 31 73 05 58 50   ...t.ZR....1s.XP
11AF:0060  E8 55 FF 58 5A 5B 26 C7-07 00 00 83 C0 08 83 C3   .U.XZ[&.........
11AF:0070  02 E2 CB EB 12 26 39 17-75 05 50 E8 3A FF 58 83   .....&9.u.P.:.X.
```

程式的另外一種結構

```
;program 2.3-2 (file_name test1.asm)
        page  45,70
        .model small (宣告DATA與CODE爲分開的節區)
        .data       (取代DATA  SEGMENT與DATA  ENDS兩個假指令)
OPR1     DW    0047H
OPR2     DW    0023H
;Exchange two words in memory
;using absolute addressing mode
        .code    (取代CODE  SEGMENT與CODE  ENDS兩個假指令)
        .startup  (設定DS、SS、與SP的初值)
SWAP     PROC  NEAR
        MOV   AX,OPR1 ;get opr1
        MOV   BX,OPR2 ;get opr2
        MOV   OPR2,AX ;save opr1
        MOV   OPR1,BX ;save opr2
        RET
SWAP     ENDP
        END
```

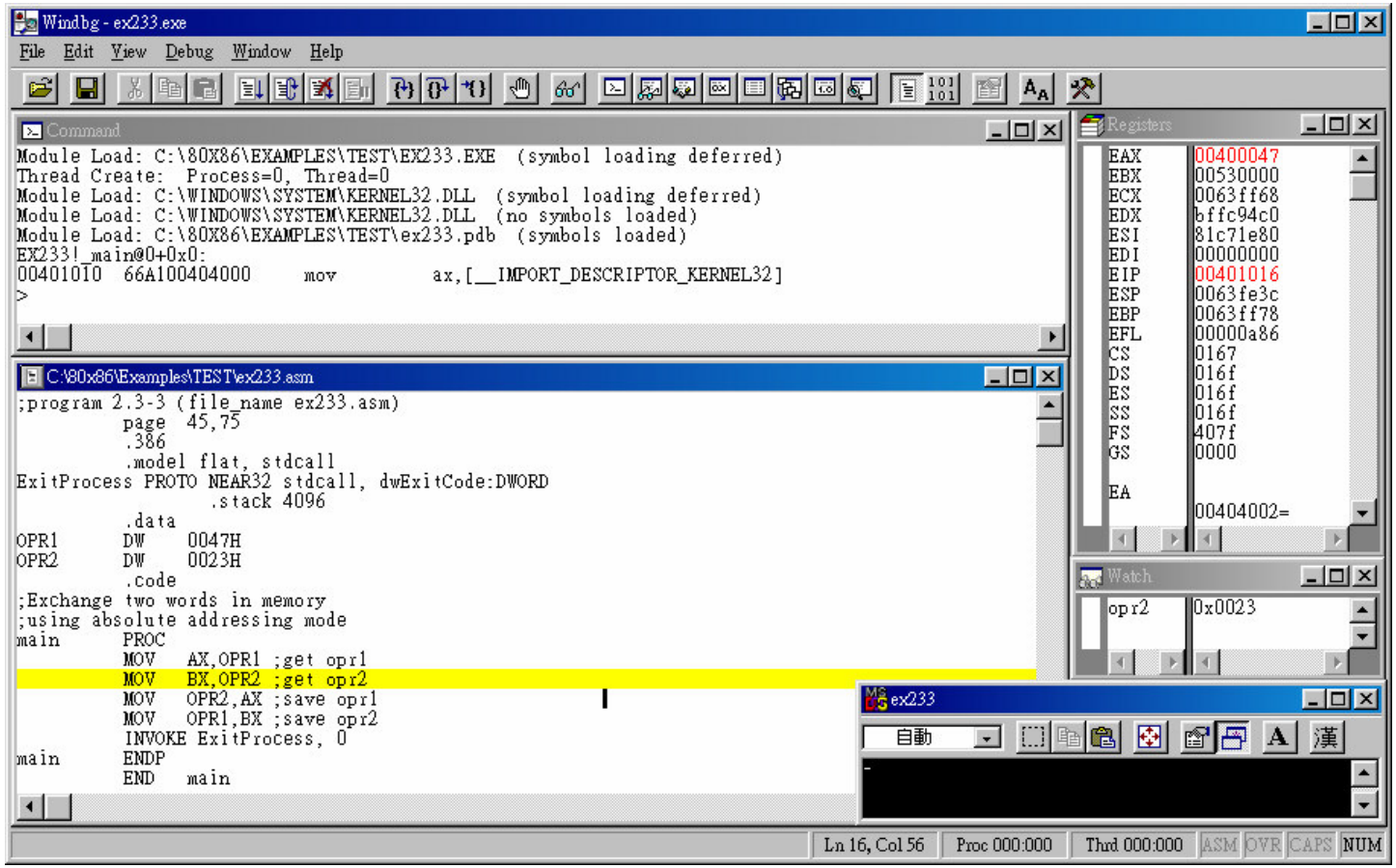
debug常用的命令

命令	功能
?	顯示 debug 的所有命令與格式
A[<啓始位址>]	自指定的啓始位址開始接受指令輸入並作組譯
C<記憶器區範圍><位址>	比較記憶器內容
D[<記憶器區範圍>]	顯示記憶器內容
E<啓始位址>[<資料序列>]	顯示記憶器內容並允許更改
F<記憶器區範圍><資料序列>	填入資料到記憶器內
G[＝<啓始位址>[<終止位址>…]]	執行
H<數值 1><數值 2>	計算數值 1 與數值 2 的和及差
I<輸入埠位址>	讀取輸入埠的資料值
L[啓始位址][磁碟機][起始扇形區][數目]	載入檔案於記憶器內
M<記憶器區範圍><目的位址>	記憶器區資料的移動
N[檔名][參數]	設定欲載入記憶器(或寫入磁碟)的檔名或程式執行時的參數
O<輸出埠位址><位元組值>	設定輸出埠的值
P［＝啓始位址］［執行的指令個數］	單步執行(見內文說明)
Q	結束 debug 程式
R[暫存器名稱]	顯示暫存器內容並允許更改
S<記憶器區範圍><資料序列>	在記憶器區範圍內尋找資料
T[= 啓始位址][執行的指令個數]	單步執行 (見內文說明)
U[記憶器區範圍或位址]	解組譯
W[啓始位址][磁碟機][起始扇形區][數目]	寫入記憶器中的資料到檔案內

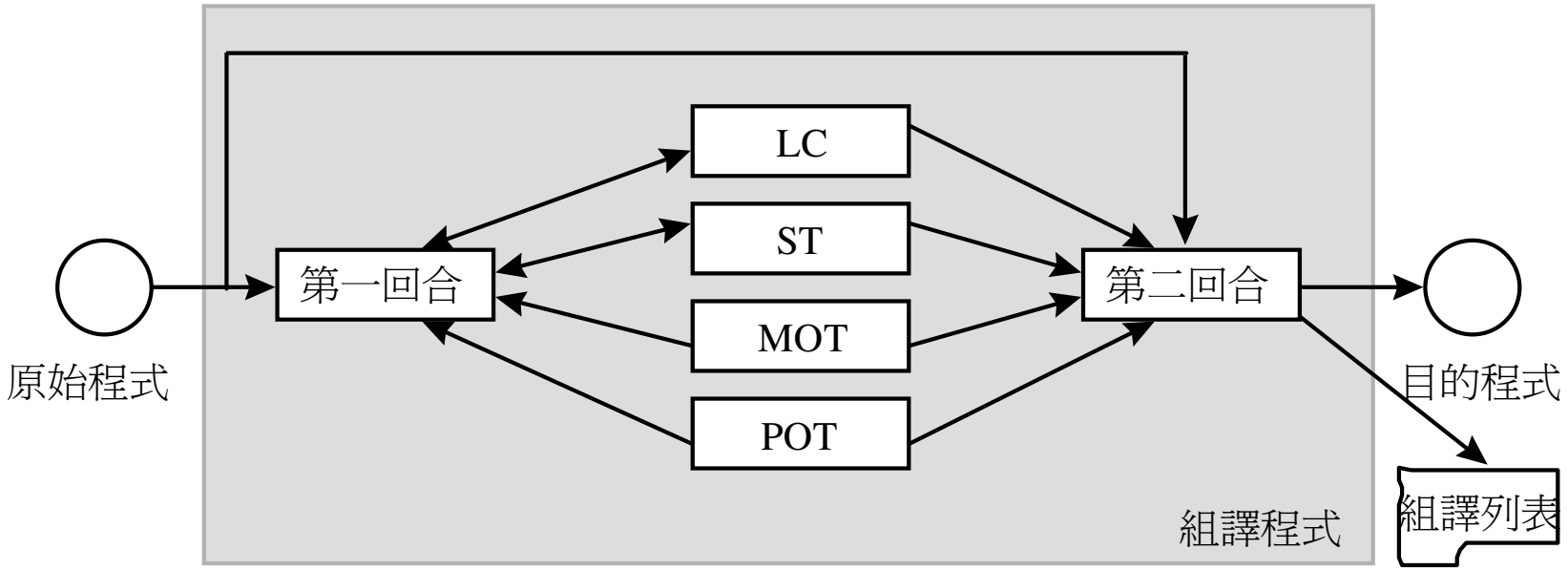
在平坦模式下的組合語言程式例

```
                        ;program 2.3-3 (file_name ex233.asm)
                                .386
                                .model flat, stdcall
                        ExitProcess PROTO NEAR32 stdcall, dwExitCode:DWORD
                                .stack 4096
00000000                            .data
00000000 0047                   OPR1    DW    0047H
00000002 0023                   OPR2    DW    0023H
00000000                            .code
                        ;Exchange two words in memory
                        ;using absolute addressing mode
00000000                        main    PROC
00000000  66| A1 00000000 R              MOV   AX,OPR1 ;get opr1
00000006  66| 8B 1D 00000002 R            MOV   BX,OPR2 ;get opr2
0000000D  66| A3 00000002 R              MOV   OPR2,AX ;save opr1
00000013  66| 89 1D 00000000 R            MOV   OPR1,BX ;save opr2
                                INVOKE ExitProcess, 0
00000021                        main    ENDP
                                END   main
```

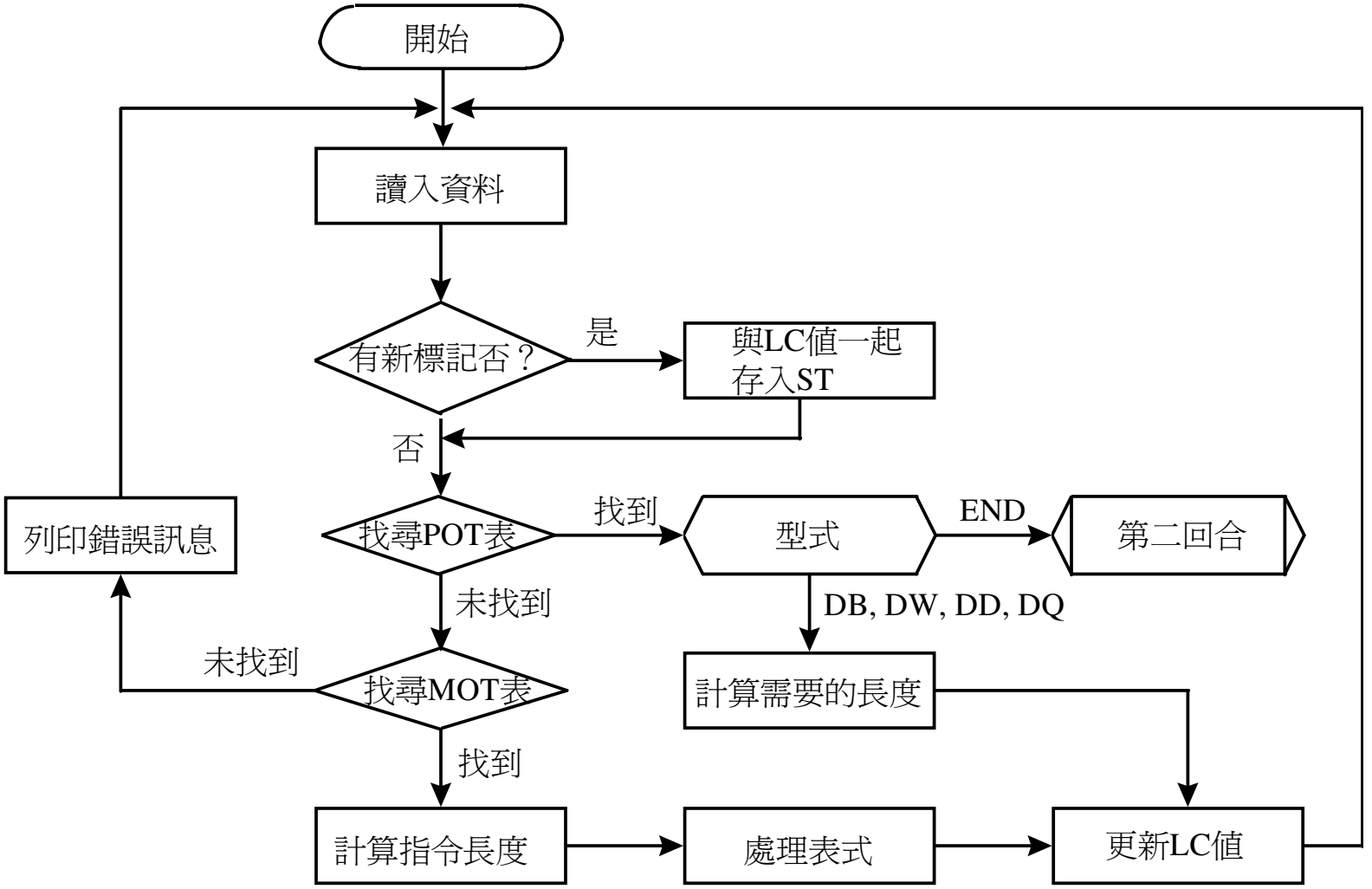
使用Windbg程式執行程式2.3-3時的相關視窗



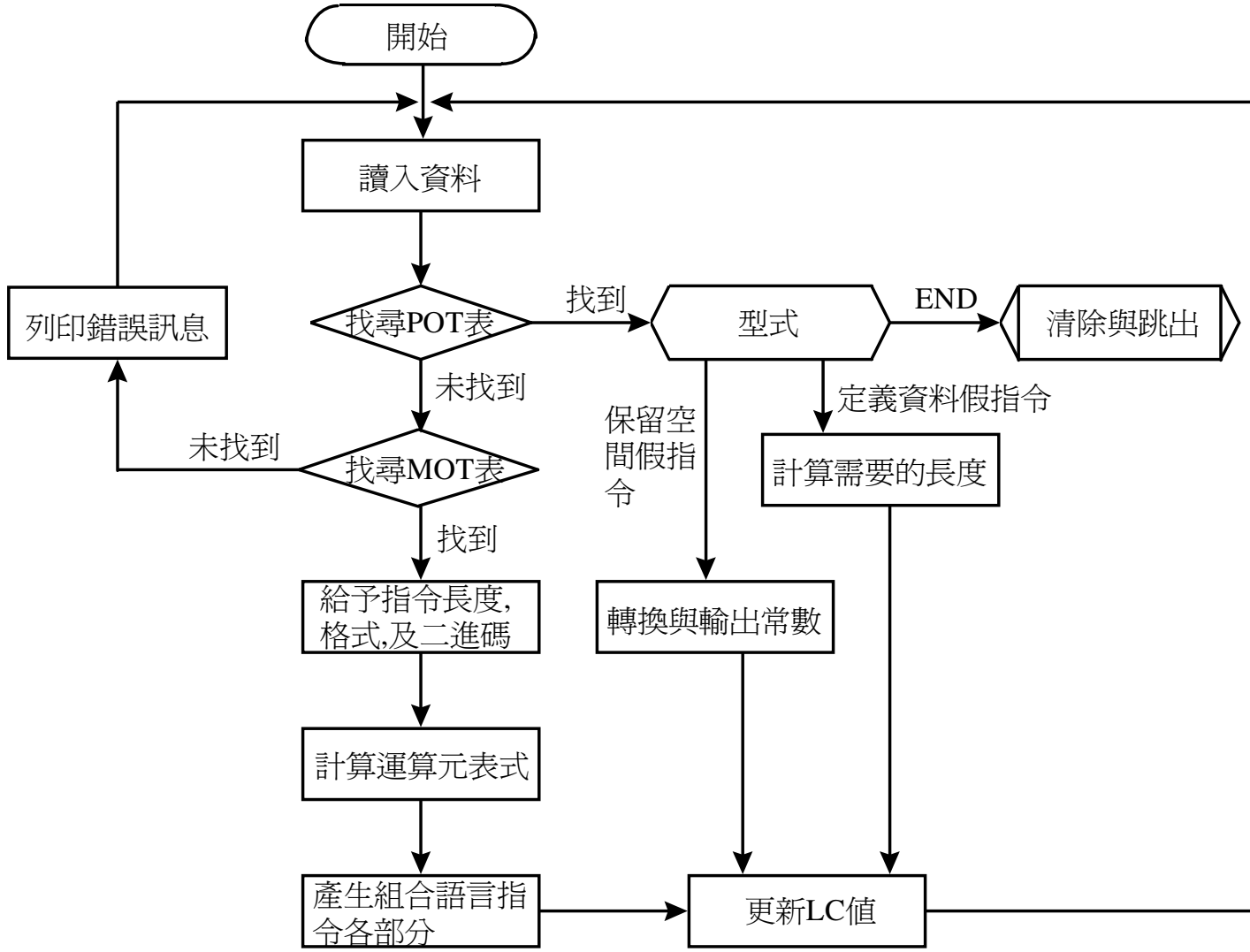
組譯程式與其相關的資料結構



組譯程式第一個回合概觀：建立ST表



組譯程式第二個回合概觀：計值與產生目的程式



組譯程序例

```
;ex5.1-4.asm
DATA      SEGMENT PUBLIC 'DATA'              LC   BYTE  CODE
BCOUNT    EQU   08H        ;bit bumber       = 0008
TDATA     DB    47H        ;test data        0000    1   47
COUNT     DB    00H        ;result           0001    1   00
EMASK     DB    01H,02H,04H,08H ;mask        0002    4   01 02 04 08
          DB    10H,20H,40H,80H               0006    4   10 20 40 80
DATA      ENDS                                000A
;count the number of 1-bit in a given byte
;using MASK and AND instruction.
```

組譯程序例

```
CODE    SEGMENT PUBLIC 'CODE'            0000
ASSUME  CS:CODE,DS:DATA
B1CNTS  PROC  NEAR                       0000
        MOV   AX,DATA    ;load DS        0000   3   B8 ---- R
        MOV   DS,AX                      0003   2   8E D8
        MOV   CX,BCOUNT  ;put count in CX  0005   3   B9 0008
        MOV   SI,00H     ;zero index     0008   3   BE 0000
        XOR   AH,AH      ;zero AH        000B   2   32 E4
BEGIN:  MOV   AL,TDATA   ;get test data   000D   3   A0 0000 R
        AND   AL,EMASK[SI];test bit value  0010   4   22 84 0002 R
        JZ    NEXT       ;if not zero    0014   2   74 02
        INC   AH         ;increase count  0016   2   FE C4
NEXT:   INC   SI         ;increase index  0018   1   46
        DEC   CX         ;repeat until    0019   2   49
        JNZ   BEGIN      ;CX = 0         001A   2   75 F1
        MOV   COUNT,AH   ;store result    001C   3   88 26 0001 R
        RET                              0020   1   C3
B1CNTS  ENDP                             0021
CODE    ENDS                             0021
        END   B1CNTS
```

組譯程序例

ST表(Symbol Table)

Name	Type	Value	Attr
BCOUNT	Number	0008h	
BEGIN	L Near	000D	CODE
COUNT	Byte	0001	DATA
EMASK	Byte	0002	DATA
NEXT	L Near	0018	CODE
TDATA	Byte	0000	DATA