另一種程式寫法 (補充教材)

- 課本的程式是採 立即定址 方式輸出字母
- 另一種寫法: 可採直接定址方式
  - 指令由 11100000 ($E0_H$) => 11100001 ($E1_H$)
  - 在程式碼後面插入資料 (參考 code/pep-p201.odc)

位址			
000	E1	00	10
003	E1	00	11
006	E1	00	12
009	E1	00	13
00C	E1	00	14
00F	00	48	65
012	6C	6C	6F

輸出 010_H 地址的內容所代表的字元

010_H 地址的內容為 48_H

作業練習-2

- 請參考前頁，以直接定址方式利用 Pep/7 列印出 "Hello, 你的英文名字"

Hello 程式變形 (補充教材)

- Hello 後增加一個由使用者互動輸入的字元 (參考 code/pep-p202.odc)

位址			
000	E0	00	48
003	E0	00	65
006	E0	00	6C
009	E0	00	6C
00C	E0	00	6F
00F	D9	00	14
012	E0	00	00
015	00		

CHARI，輸入的字元儲存到 014_H 記憶體位置

這裡是 014_H

7.5 組合語言

- 組合語言 (assembly language)
  - 指定一些助憶的(mnemonic)字母編碼 (簡稱助憶碼)來代表每一個機器語言指令。
  - 程式設計師使用這些助憶碼編寫程式而不需再使用二進制位元
  - 組譯器 (assembler) 這個程式就負責讀入每一個助憶符號形式的指令，然後轉譯成相對的機器語言形式。

圖 7.5 組譯過程

Pep/7組合語言

助憶符號	運算元，模式指標	指令的意義
STOP		停止執行
LOADA LOADA	h#008B,i h#008B,d	將 008B 載入暫存器 A 將 8B 位址的內容載入暫存器 A
STOREA	h#008B,d	將暫存器 A 的內容存到 8B 位址
ADDA ADDA	h#008B,i h#008B,d	將 008B 數值加入暫存器 A 將 8B 位址的內容值加入暫存器 A
SUBA SUBA	h#008B,i h#008B,d	由暫存器 A 減去 008B 數值 由暫存器 A 減去 8B 位址的內容值
CHARI	h#008B,d	讀取一個字元並儲存到位元組 8B 處
CHARO	c#/B/,i h#008B,d	寫入一個字元 B 寫入一個儲存在位元組 8B 處的字元
DECI DECO DECO	h#008B,d h#008B,i h#008B,d	讀取一個十進制數並儲存到 8B 位址 寫入一個十進制數 139 (十六進制 8B) 寫入儲存在 8B 位址處的十進制數

縮寫：CHAR 為 CHARACTER(字母)，DEC 為 DECIMAL (十進制)，h 為 hexidecimal，c 為 character，i 為 immediate，d 在 “,” 後為 direct， d 在 “,” 前是 decimal

虛擬運算碼

定義：組譯器巨集指揮動作（assembler directive），也就是請組譯器完成的準備動作，以便利組合語言程式的執行，在 Pep7 有 4 個虛擬運算碼（其中 .WORD 有 2 種運算元表示方式），都是以 ‘.’ 開始：

虛擬運算碼	運算元	意義
.ASCII	/…/	儲存位於兩右斜線之間 (//) 的字元到記憶體中
.BLOCK	d#3	產生 3 個位元組的儲存空間並設定每個位元組的值為零
.WORD	d#5	產生一個字語並存入十進制值 5
.WORD	h#0005	產生一個字語並存入十六進制值 5
.END		表示組合語言程式的結束

BLOCK：區塊，占用記憶體位元組數目由運算元的值決定
WORD：字語：占用記憶體位元組數目固定為 2 個位元組，適用於需要給訂初始值時

Hello程式(7-4節)的組合語言版本 (1)

- 寫出 “Hello”，在組合語言中必須一次輸出一個字母

```
CHARO c#/H/,i  ;Output 'H'
CHARO c#/e/,i  ;Output 'e'
CHARO c#/l/,i  ;Output 'l'
CHARO c#/l/,i  ;Output 'l'
CHARO c#/o/,i  ;Output 'o'
STOP
.END
```

(立即定址模式)

(參考 code/pep-p206-1.odc 及 code/pep-p206-2.odc)

Hello程式(7-4節)的組合語言版本 (2)

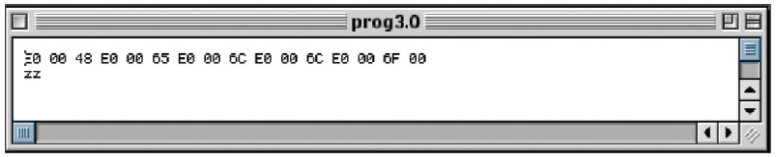
- (直接定址模式)

```
CHARO h#0010,d ;Output 'H'
CHARO h#0011,d ;Output 'e'
CHARO h#0012,d ;Output 'l'
CHARO h#0013,d ;Output 'l'
CHARO h#0014,d ;Output 'o'
STOP
.ASCII /Hello/ ;Store 'Hello' into proper places
.END
```

(參考 code/pep-p208.odc)

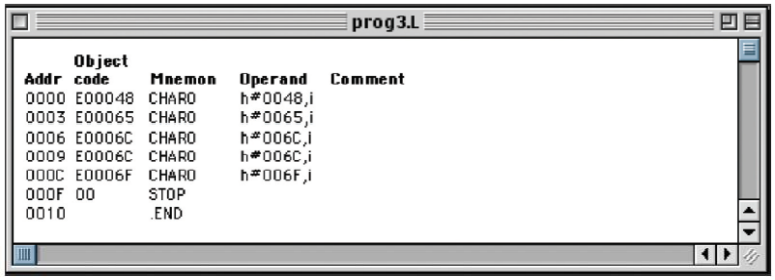
Hello程式(7-4節)的組合語言版本(3)

- 執行了Pep/7下拉選單的Assemble選項之後的結果顯示於下面的視窗畫面：



Hello程式(7-4節)的組合語言版本(4)

- 我們也可以點選Pep/7下拉選單下的Assembler Listing選項，以得到組譯器列表如下圖所示。



Hello 程式變形 的組合語言版(補充教材)

(參考 code/pep-p209.odc)

```
CHARO h#16,d ;Output 'H'
CHARO h#17,d ;Output 'e'
CHARO h#18,d ;Output 'l'
CHARO h#19,d ;Output 'l'
CHARO h#1A,d ;Output 'o'
CHARI h#14,d ;Input an initial
CHARO h#00,i ;Output initial
STOP
.ASCII /Hello/ ;Store 'Hello' into proper places
.END
```


一個加總程式範例 (1)

需求：請輸入 3 個整數，並計算其加總所得數字，

Reading and adding three numbers
Set sum to 0
Read num1
Add num1 to sum
Read num2
Add num2 to sum
Read num3
Add num3 to sum
Write sum

需要 4 個放整數的記憶體位置給 輸入值及加總結果 num1, num2, num3, sum

1. **將 sum 的記憶體內容先設為 0，載入到 ALU**
2. **輸入 num1，再利用 ADDA 將 ALU 內容(0)與 num1 加起來 (ALU 會存有 num1)**
3. **輸入 num2，再利用 ADDA 將 ALU 內容與 num2 加起來 (ALU 存有 num1 + num2結果)**
4. **輸入 num3，再利用 ADDA 將 ALU 內容與 num3 加起來 (ALU 存有三數加總結果)**
5. **將 ALU結果以 STOREA 存到 sum的記憶體位置**

一個加總程式範例 (2)

虛擬運算碼：

```
sum:   .WORD d#0    ;設定一個字語內容為零
num1:  .BLOCK d#2   ;設定兩個位元組區塊給 num1
num2:  .BLOCK d#2   ;設定兩個位元組區塊給 num2
num3:  .BLOCK d#2   ;設定兩個位元組區塊給 num3
```

標籤：可用來命名變數或標示程式位置，以變數名稱開頭後加上 ':'，如 **sum, num1, num2, num3**。**Main** 為所有程式的開始執行位置之標籤

一個加總程式範例 (3)

無條件跳躍

預備動作

開始執行標籤

```
        BR Main         ;跳到位址 Main
sum:    .WORD d#0       ;設定一個字語並放入零指定給 sum
num1:   .BLOCK d#2      ;設定兩個位元組區塊給 num1
num2:   .BLOCK d#2      ;設定兩個位元組區塊給 num2
num3:   .BLOCK d#2      ;設定兩個位元組區塊給 num3
Main:   LOADA sum,d     ;將 sum 內容載入到累加器
        DECI num1,d     ;讀入並儲存一個十進制數到 num1
        ADDA num1,d     ;將 num1 內容加到累加器
        DECI num2,d     ;讀入並儲存一個十進制數到 num2
        ADDA num2,d     ;將 num2 的內容加到累加器
        DECI num3,d     ;讀入並儲存一個十進制數到 num3
        ADDA num3,d     ;將 num3 的內容加到累加器
        STOREA sum,d    ;將累加器的內容儲存到 sum
        DECO sum,d      ;輸出 sum 的內容
        STOP            ;停止程序
        .END            ;程式結束
```


一個加總程式範例 (4)

Pep7 - [pep-p212.odc.L]

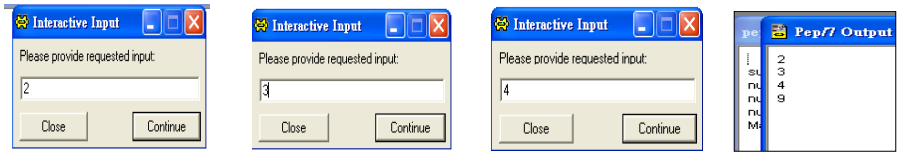
File Edit Attributes Text Pep7 Window Help

Addr	Object code	Symbol	Mnemon	Operand	Comment
0000	70000B		BR	Main	;跳到 Main
0003	0000	sum:	.WORD	d#0	;設定一 WORD 大小的變數 sum，其內容 爲 0
0005	0000	num1:	.BLOCK	d#2	;設定一2bytes大小的區塊給變數 num1
0007	0000	num2:	.BLOCK	d#2	;設定一2bytes大小的區塊給變數 num2
0009	0000	num3:	.BLOCK	d#2	;設定一2bytes大小的區塊給變數 num3
000B	090003	Main:	LOADA	sum,d	;將 sum 的內容載入到累加器
000E	E90005		DECI	num1,d	;讀入並儲存一十進制數到 num1
0011	190005		ADDA	num1,d	;將 num1 內容加到累加器
0014	E90007		DECI	num2,d	;讀入並儲存一十進制數到 num2
0017	190007		ADDA	num2,d	;將 num2 內容加到累加器
001A	E90009		DECI	num3,d	;讀入並儲存一十進制數到 num3
001D	190009		ADDA	num3,d	;將 num3 內容加到累加器
0020	110003		STOREA	sum,d	;將累加器的內容存到 sum
0023	F10003		DECO	sum,d	;輸出 sum 的內容
0026	00		STOP		
0027			.END		

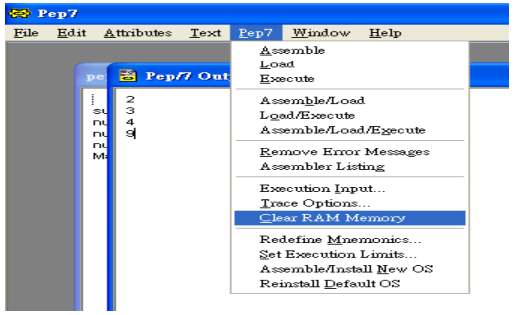
Symbol	Value	Symbol	Value
Main	000B	num1	0005
num2	0007	num3	0009
sum	0003		

 (參考 code/pep-p212.odc)

執行結果



繼續執行同一程式會一直累加，若不想累加之前執行的結果，須以如下方式清空記憶體 內容，再重新 **Assemble/Load/Execute**



作業練習-3

- 請參考課本的加總程式範例，改寫成只需要 2 個放整數的記憶體位置給輸入值 (num)及加總結果 (sum)
  - 提示：因為本例題的各個輸入值加到加總後就沒有用了，因此，可以讓原程式的 num1, num2, num3 共用一個記憶體位置 num，也就是將原來的 “DECI num1, d”、“DECI num2, d”、“DECI num3, d” 都改為 “DECI num,d” ，記得**虛擬運算碼部分也要將 3 個虛擬運算碼改成 1 個**

- **課本習題第 16-20 題（手寫）**

請於 12 月 5 日前將 3 題作業練習的程式碼繳交至教學支援平台

在組合語言中做決策(作判斷)的方法

Branch 及 Compare 的縮寫，EQ 為 Equal，LT 為 Less Than

助憶符號	運算元，模式指標	指令的意義
BRLT	沒有用到	如果 N 為 1 (A 暫存器小於 0)，設定 PC 為運算元
BREQ	沒有用到	如果 Z 為 1 (A 暫存器等於 0)，設定 PC 為運算元
COMPA	H#008B, i	比較累加器的內容與 008B 位址的內容，並設定狀態位元
COMPA	H#008B, d	比較累加器的內容與運算元的內容，並設定狀態位元

本例為 $008B_H$

範例：

```
LOADA num1,d
BRLT lessThan ；如果 num1 小於 0 就分歧跳到 lessThan
```


在組合語言中做決策(作判斷)的方法

- COMPA limit, d：比較 A 暫存器內容與 limit 變數的值
  - 當 A 暫存器內容 < limit 的值，會設定狀態位元 N 的值為 1
  - 當 A 暫存器內容 = limit 的值，會設定狀態位元 Z 的值為 1
- BRLT Case2: 當狀態位元 N (表 Negative，負) 的值為 1，則將程式計數器暫存器 (Program Counter ，PC) 設為運算元 Case2 標籤的記憶體位址
  - 註：當 N 為 1 表示 A暫存器的值小於 0
- BREQ Case1:當狀態位元 Z (表 Zero，零) 的值為 1，則將程式計數器暫存器 (Program Counter ，PC) 設為運算元 Case1標籤的記憶體位址
  - 註：當 Z 為 1 表示 A暫存器的值等於 0

在組合語言中做決策(作判斷)的方法

- 假設需求改為：當總和sum為正數時就印出sum，如果sum為負值則印出一個錯誤訊息。

```
...
Add num3 to sum
If sum is negative
    Write "Error"
else
    Write sum
```

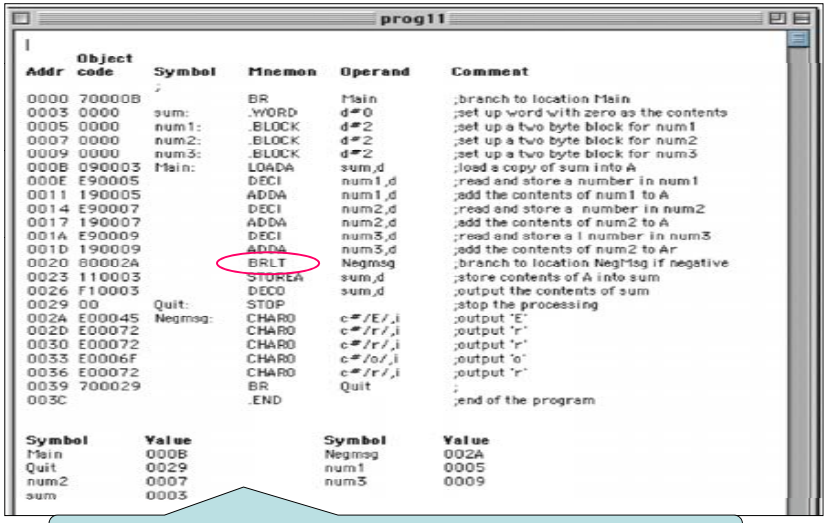
if 敘述，其基本型式(注意縮排)為：
if 條件
**　條件成立時的操作**
else
**　條件不成立時的操作**

- 上面這個 *If* 敘述展開成為組合語言的演算法的形式如下：

```
if status bit N is 1
    Go to NegMsg
Wirte sum
Quit: STOP
NegMsg: Write the message and go to Quit
```

if 敘述的變形，其基本型式為：
if 條件
**　條件成立時的操作**
不管條件成不成立，都進行的操作

在組合語言中做決策的方法範例程式



prog11

```
Addr  Object code  Symbol   Mnemon   Operand    Comment
0000  700008      ;        BR       Main       ;branch to location Main
0003  0000        sum:     .WORD    d#0        ;set up word with zero as the contents
0005  0000        num1:    .BLOCK   d#2        ;set up a two byte block for num1
0007  0000        num2:    .BLOCK   d#2        ;set up a two byte block for num2
0009  0000        num3:    .BLOCK   d#2        ;set up a two byte block for num3
000B  090003      Main:    LOADA    sum,d      ;load a copy of sum into A
000E  E90005               DECI     num1,d     ;read and store a number in num1
0011  190005               ADDA     num1,d     ;add the contents of num1 to A
0014  E90007               DECI     num2,d     ;read and store a  number in num2
0017  190007               ADDA     num2,d     ;add the contents of num2 to A
001A  E90009               DECI     num3,d     ;read and store a I number in num3
001D  190009               ADDA     num3,d     ;add the contents of num2 to Ar
0020  80002A               BRLT     Negmsg     ;branch to location NegMsg if negative
0023  110003               STOREA   sum,d      ;store contents of A into sum
0026  F10003               DECO     sum,d      ;output the contents of sum
0029  00          Quit:    STOP                ;stop the processing
002A  E00045      Negmsg:  CHAR0    c#'E',i    ;output 'E'
002D  E00072               CHAR0    c#'r',i    ;output 'r'
0030  E00072               CHAR0    c#'r',i    ;output 'r'
0033  E0006F               CHAR0    c#'o',i    ;output 'o'
0036  E00072               CHAR0    c#'r',i    ;output 'r'
0039  700029               BR       Quit       ;
003C                       .END                ;end of the program

Symbol  Value   Symbol  Value
Main    000B    Negmsg  002A
Quit    0029    num1    0005
num2    0007    num3    0009
sum     0003
```

因為 硬體自動設定 N 及 Z，所以可以直接BRLT

一個具有迴圈的程式

迴圈：程式可以反覆執行同一段程式碼若干次

做法：讓 BR (或 BREQ, BRLT, COMPA) 等可以跳回至前面的標籤

假設需求為：請先輸入要加總的整數的個數(每次執行時才決定，不是事先決定的)，再請使用者一個一個輸入這些整數，加總後列印其總和

```
Read limit
Set sum to 0
While (limit is not zero)
    Read number
    Set sum to sum + number
    Set limit to limit - 1
```

while 敘述 的類型 (要注意縮排)：

```
while condition
    statement1
    statement2
    statement3
```


一個具有迴圈的程式

- 第一步是預備動作，先宣告我們會使用到的變數：limit, number及sum。

```
        BR Main
sum:    .WORD d#0    ; 宣告 sum 是一個字語初值為 0
number: .BLOCK d#2   ; 宣告 number 是兩個位元組的區段
limit:  .BLOCK d#1   ; 宣告 limit 是一個位元組的區段
```

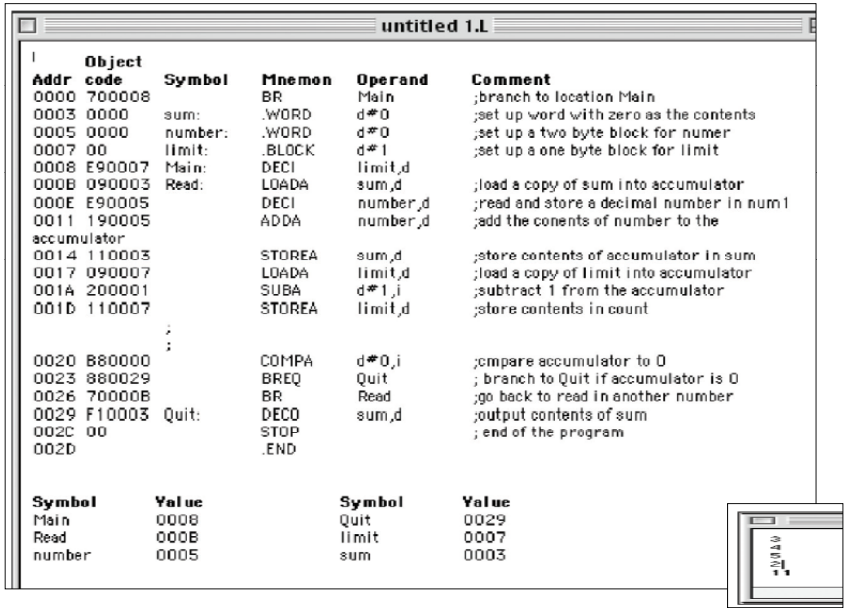
每次進行迴圈時，內部的做法：

```
Set the accumulator to limit
Subtract one from the accumulator
Compare accumulator to zero
if status bit Z is 1
    go to Quit
Else
    go to Read
```


一個具有迴圈的程式

```
Main:   DECI limit,d     ; 讀取並儲存一個十進制數到 limit
Read:   LOADA sum,d      ; 載入 sum 的值到累加器
        DECI number,d    ; 讀取並儲存一個十進制數到 number
        ADDA number,d    ; 將 number 的內容加到累加器中
        STOREA sum,d     ; 將累加器的內容儲存到 sum
        LOADA limit,d    ; 載入 limit 的值到累加器
        SUBA d#1,i       ; 將累加器的內容減 1
        STOREA limit,d   ; 將計數值儲存到 limit
        COMPA d#0,i      ; 比較累加器的內容是否為 0
        BREQ Quit        ; 如果累加器的內容為 0 就跳到 Quit
        BR Read          ; 如果不為 0 就跳到 Read
Quit    DECO sum,d       ; 輸出 sum 的內容到螢幕上
        STOP             ; 停止執行
        .END             ; 結束程式
```

一個具有迴圈的程式



untitled 1.L

```
1      Object
Addr   code     Symbol    Mnemon   Operand     Comment
0000   700008             BR       Main        ;branch to location Main
0003   0000     sum:      .WORD    d#0         ;set up word with zero as the contents
0005   0000     number:   .WORD    d#0         ;set up a two byte block for numer
0007   00       limit:    .BLOCK   d#1         ;set up a one byte block for limit
0008   E90007   Main:     DECI     limit,d
000B   090003   Read:     LOADA    sum,d       ;load a copy of sum into accumulator
000E   E90005             DECI     number,d    ;read and store a decimal number in num1
0011   190005             ADDA     number,d    ;add the conents of number to the
accumulator
0014   110003             STOREA   sum,d       ;store contents of accumulator in sum
0017   090007             LOADA    limit,d     ;load a copy of limit into accumulator
001A   200001             SUBA     d#1,i       ;subtract 1 from the accumulator
001D   110007             STOREA   limit,d     ;store contents in count
                 ;
                 ;
0020   B80000             COMPA    d#0,i       ;cmpare accumulator to 0
0023   880029             BREQ     Quit        ; branch to Quit if accumulator is 0
0026   70000B             BR       Read        ;go back to read in another number
0029   F10003   Quit:     DECO     sum,d       ;output contents of sum
002C   00                 STOP                 ; end of the program
002D                      .END

Symbol      Value       Symbol      Value
Main        0008        Quit        0029
Read        000B        limit       0007
number      0005        sum         0003
```

更多範例程式 (補充教材)

- 比大小(使用 if 的概念)
  - code/Pr0701.odc ：跟 100 比大小，大的話輸出 “high” ，否則輸出 “low”
  - code/Pr0701-2.odc ：增加 等於 100 時輸出 “eq” 的檢查
  - code/Pr0701-3.odc： 較簡潔的寫法
- 迴圈 (while)
  - code/Pr0702.odc ：警車追賊車

code/Pr0701.odc

```
;Program 0701 補充教材
        BR      Main
limit:  .EQUATE d#100       ;設定 limit 為十進位100，此為不會改變的常數
                            ;用 .EQUATE 不會佔用記憶體位置
num:    .BLOCK  d#2         ;
Main:   DECI    num,d       ;輸入十進位數字 num
If:     LOADA   num,d
        COMPA   limit,i     ;比較 num 及 limit
        BRLT    Case2       ;if (num < limit) 則跳到 Case2 位置
        CHARO   c#/h/,i     ;否則輸出 "high"，
                            ;當 num=100 有問題，請參考 Pr0701-2.odc
                            ;                       或 Pr0701-3.odc
        CHARO   c#/i/,i
        CHARO   c#/g/,i
        CHARO   c#/h/,i
        BR      EndIf       ;跳到 EndIf 標籤位置
Case2:  CHARO   c#/l/,i     ;輸出 "low"
        CHARO   c#/o/,i
        CHARO   c#/w/,i
EndIf:  STOP
        .END
```

因為 limit 為常數，此處才能用 i(立即模式),

code/Pr0701-2.odc

```
;Program 0701 補充教材
        BR      Main
limit:  .EQUATE d#100       ;設定 limit 為十進位  100
num:    .BLOCK  d#2
;
Main:   DECI    num,d       ;輸入十進位數字 num
If:     LOADA   num,d
        COMPA   limit,i     ;比較 num 及 limit
        BRLT    Case2       ;if ( num < limit ) 則跳到 Case2 位置
        LOADA   limit, i
        COMPA   num, d
        BRLT    Case1       ; if (limit < num) 則跳到 Case1 位置
        CHARO   c#/e/,i     ;輸出 "eq"
        CHARO   c#/q/,i
        BR      EndIf       ;跳到 EndIf 標籤位置
Case1:  CHARO   c#/h/,i     ;輸出 "high"
        CHARO   c#/i/,i
        CHARO   c#/g/,i
        CHARO   c#/h/,i
        BR      EndIf
Case2:  CHARO   c#/l/,i     ;輸出 "low"
        CHARO   c#/o/,i
        CHARO   c#/w/,i
EndIf:  STOP
        .END
```

code/Pr0701-3.odc

```
;Program 0701 補充教材
        BR      Main
limit:  .EQUATE d#100       ;設定 limit 爲十進位  100
num:    .BLOCK  d#2
;
Main:   DECI    num,d       ;輸入十進位數字 num
If:     LOADA   num,d
        COMPA   limit,i     ;比較 num 及 limit
        BRLT    Case2       ;if ( num < limit ) 則跳到 Case2 位置
        BREQ    Case1       ; if (limit == num) 則跳到 Case1 位置
        CHARO   c#/h/,i     ;輸出 "high"
        CHARO   c#/i/,i
        CHARO   c#/g/,i
        CHARO   c#/h/,i
        BR      EndIf       ;跳到 EndIf 標籤位置
Case1:  CHARO   c#/e/,i     ;輸出 "eq"
        CHARO   c#/q/,i
        BR      EndIf
Case2:  CHARO   c#/l/,i     ;輸出 "low"
        CHARO   c#/o/,i
        CHARO   c#/w/,i
EndIf:  STOP
        .END
```


code/Pr0702.odc

```
;Program 0702 補充教材
;警車追罪犯，一開始，警車車速爲 0，罪犯車速爲 45，
;警車需加速到比罪犯還快，但罪犯也會加速，
;警車每次加速 25 公里，罪犯每次加速 15 公里
;問題是最後警車的車速爲多少？
        BR      Main
cop:    .BLOCK  d#2         ; 警車車速
criminal: .BLOCK d#2        ; 罪犯車速
;
Main:   LOADA   d#0,i       ; A 暫存器內容設爲 0
        STOREA  cop,d       ; 警車車速設爲 0
        LOADA   d#45,i      ; A 暫存器內容設爲 45
        STOREA  criminal,d  ; 罪犯車速設爲 45
Do:     LOADA   cop,d       ; 以下三列讓警車車速加 25
        ADDA    d#25,i      ;
        STOREA  cop,d       ;
        LOADA   criminal,d  ; 以下3列讓罪犯車速加 15
        ADDA    d#15,i      ;
        STOREA  criminal,d  ;
While:  CHARO   c#/c/,i     ; 以下5列 輸出警車車速
        CHARO   c#/o/,i
        CHARO   c#/p/,i
        CHARO   c#/=/,i
        DECO    cop,d
        CHARO   c#/ /,i
        CHARO   c#/c/,i     ; 以下5列輸出罪犯車速
        CHARO   c#/r/,i
        CHARO   c#/i/,i
        CHARO   c#/=/,i
        DECO    criminal,d
        CHARO   c#/ /,i
        LOADA   cop,d       ; 將警車車速載入 A 暫存器
        COMPA   criminal,d  ; 比較 A 暫存器內容與罪犯車速
        BRLT    Do          ; 當 (A 暫存器內容 < 罪犯車速) 則跳到 Do 標籤處
        BREQ    Do          ; 當 (A 暫存器內容 == 罪犯車速) 則跳到 Do 標籤處
        DECO    cop,d       ; 輸出 警車車速
        STOP
        .END
```


作業： 12/12 交到教學支援平台

1. 請輸入 3 個整數，分別存在 num1, num2, num3 這 3 個變數，請計算出 num1 – num2 + num3 的值，輸出結果。如果 結果為正，輸出 “++”，如果結果為負，輸出 “--”，如果結果為 0，輸出 “0” (註：減法的部分，可用 SUBA 助憶符號)
2. 請輸入一個正整數值 n，計算 1 + 2 + … + n 的總和
   - 說明：可以參考 第 24 頁範例(一個具有迴圈的程式)，但本程式在迴圈中不用輸入值，直接加上 limit 的值即可
3. 課本習題第 41, 42 題，找出程式碼的錯誤，並且加以訂正 (第 41 題以文字檔繳交，第 42 題以程式碼繳交)

7.6 其他重要的關聯：抽象化與測試

- 機器語言有很少的資訊隱藏，僅隱藏以下位元資訊：以二的補數表示負整數
- 組合語言可用抽象化概念隱藏一些細節
  - 用變數來命名一個區塊(BLOCK)或一個字組(WORD)
  - 標示程式位置，這樣程式可直接跳到這個位置，這在設計迴圈及 IF-ELSE 程式碼時尤其方便
- 一個測試計畫 (test plan)基本上是一件文件，當中指定要執行多少次及使用什麼樣的資料來測試程式以達到程式的完整測試。每一組輸入資料值就叫作一個**測試樣本** (*test case*)。

抽象化與測試

- 測試過程中有幾種測試方法可以依循
  - 一種稱為涵蓋碼 (code-coverage) 的方法是設計測試樣本時要確保程式中所有的敘述都要被執行到。因為對於測試者而言，程式碼都是看得到的，所以這種方法也被稱作透明盒子測試 (clear-box testing)。
  - 另一種稱為涵蓋資料測試 (data-coverage testing) 的方法是設計測試樣本時要確保所有可能資料範圍都必須涵蓋到。由於這種方法只基於輸入資料本身而不管程式碼，所以也被稱作黑盒子測試 (black-box testing)。

測試計畫的實施

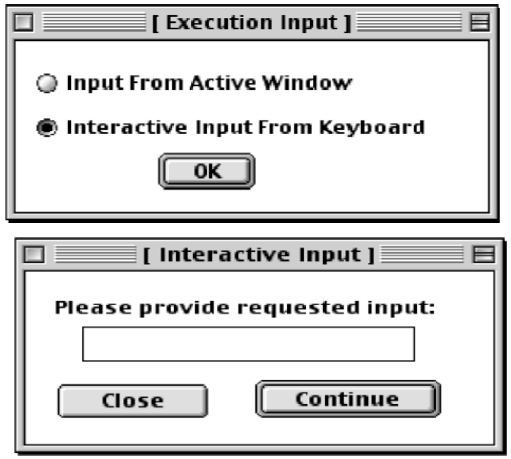
- 測試計畫的實施 (test-plan implementation) 涵蓋執行測試計畫中所列的每一個測試樣本，並記錄所有的結果。
- 我們要做加總三個數字的動作，所以設計輸入值時必須確定加總後的和不會超過 $\pm 2^{15} - 1$。

使用測試樣本的理由	輸入值	預期輸出值	實際看到的輸出
假設：輸入數值不大於 $2^{15} - 1$ 或不小於 -2^{15}			
輸入三個正值數字	4, 6, 1	11	
輸入三個負值數字	−4, −6, −1	−11	
輸入混合數字	4, 6, −1	9	
	4, −6, 1	−1	
	−4, 6, 1	3	
輸入最大數值	32767, −1, + 1	32767	

- 要實現這個測試計畫，程式必須執行六次，逐次測試每一種資料樣本。

測試計畫的實施

- Pep/7選單中選擇*Execution Input* (執行輸入)，我們將會見到以下的畫面：



測試計畫的實施

- 我們只要鍵入一個數字並點選*Continue*鈕。這個畫面會再出現兩次以便我們輸入另外兩個數值。如果我們先輸入第一個數值4，第二個數值6，及第三個數值1，則我們可得到下面的輸出畫面：