巨集處理器 (二)

- 巨集處理器主要的四項功能爲：
  (1) 巨集定義之辨識
  (2) 巨集定義之儲存
  (3) 巨集呼叫之辨識
  (4) 巨集呼叫之展開與參數置換

- 巨集處理器之分類

  (1) 兩次處理巨集處理器 (Two-Pass Macro Processor)

  (2) 一次處理巨集處理器 (One-Pass Macro Processor)

  (3) 巨集組合程式 (Macro Assembler)

兩次處理巨集處理器

- 功能：於第一次處理時，處理所有的巨集定義。在第二次處理時，則展開所有的巨集呼叫。
- 限制：巨集定義內不可有巨集定義或巨集呼叫。

1. 第一次處理 ：巨集定義的辨認並儲存巨集定義。

- 使用到的資料庫 ：

  (a) 輸入 ：始程式。

  (b) 處理過程中使用到的資料結構：

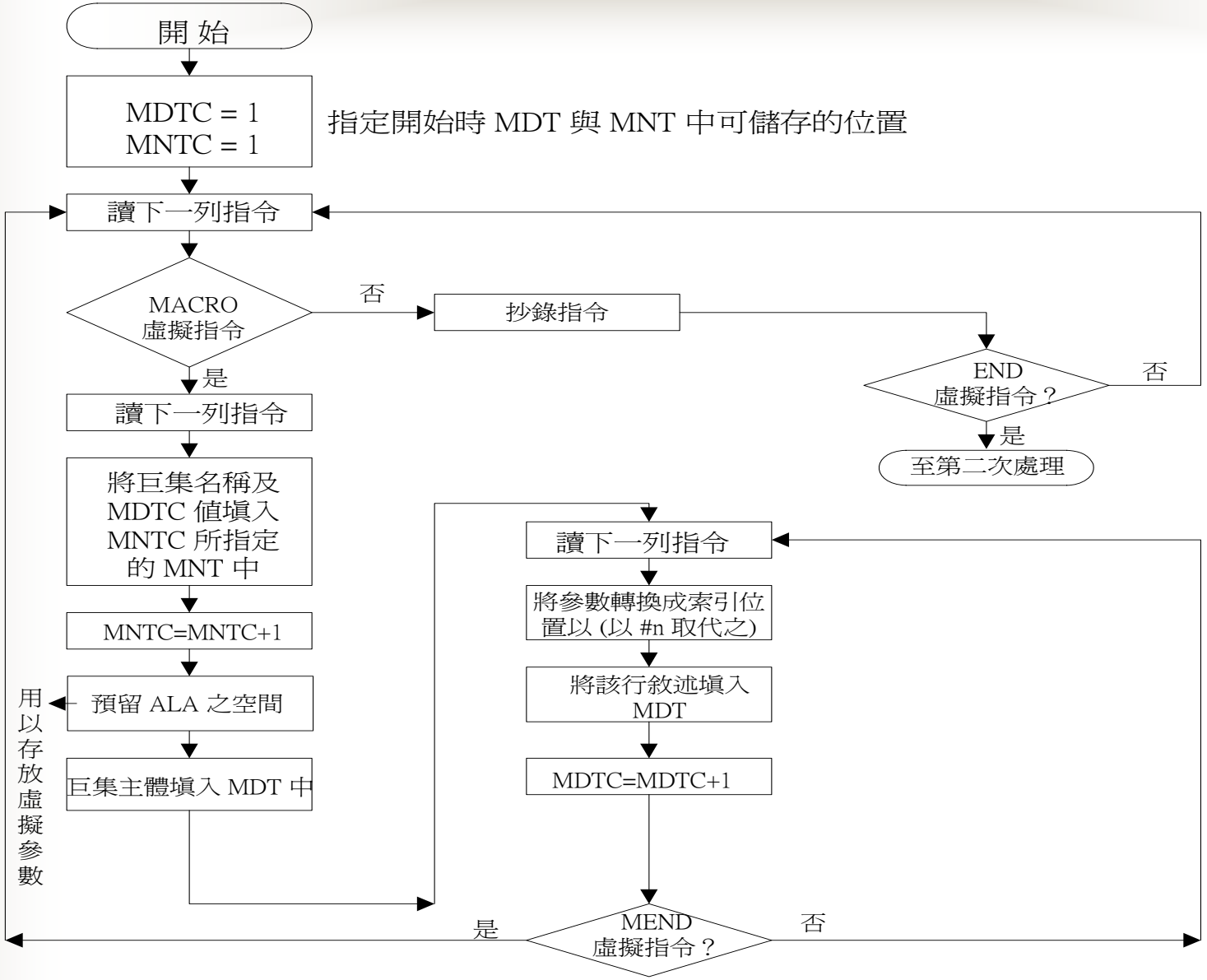
  - (i) 巨集定義表計數器 (MDTC)：用以指向下一個將會被使用的巨集定義表 (MDT)。
  - (ii) 巨集名稱表計數器 (MNTC)：用以指向下一個將會被使用的巨集名稱表 (MNT)。
  - (iii) 參數陣列 (ALA)：用以儲存巨集指令引用之虛擬參數，以便於巨集展開時能順利而正確的做參數取代。其虛擬參數一般會以其在參數陣列中之索引位置代替之。

(c) 輸出 ：

- (i) 不含有巨集定義之原始程式。
- (ii) 巨集定義表 (MDT)：
  其內用來儲存巨集指令之內容。
- (iii) 巨集名稱表 (MNT)：其內用來儲存巨集指令之名稱。

- 處理過程：

  讀入並檢查每一列指令，若該指令爲 MACRO，則將其後之巨集主體儲存於 MDTC 所指定之 MDT 位置內，並且將 MDTC 值加 1。另將巨集名稱儲存於 MNTC 所指定之 MNT 位置內，並且將 MNTC 值加 1。重覆上述之動作，直至讀 END 指令，便完成第一階段巨集定義的儲存工作。


開 始
MDTC = 1
MNTC = 1
指定開始時 MDT 與 MNT 中可儲存的位置
讀下一列指令
MACRO
虛擬指令
否
抄錄指令
END
虛擬指令？
否
是
至第二次處理
是
讀下一列指令
將巨集名稱及
MDTC 值填入
MNTC 所指定
的 MNT 中
MNTC=MNTC+1
預留 ALA 之空間
用以存放虛擬參數
巨集主體填入 MDT 中
讀下一列指令
將參數轉換成索引位置以 (以 #n 取代之)
將該行敘述填入
MDT
MDTC=MDTC+1
MEND
虛擬指令？
是
否

2. 第二次處理 (Pass 2)：辨識巨集呼叫，並做巨集展開。

- 使用到的資料庫：

  (a) 輸入：

  - (i) 不含有巨集定義之原始程式。
  - (ii) 巨集定義表 (MDT)
  - (iii) 巨集名稱表 (MNT)

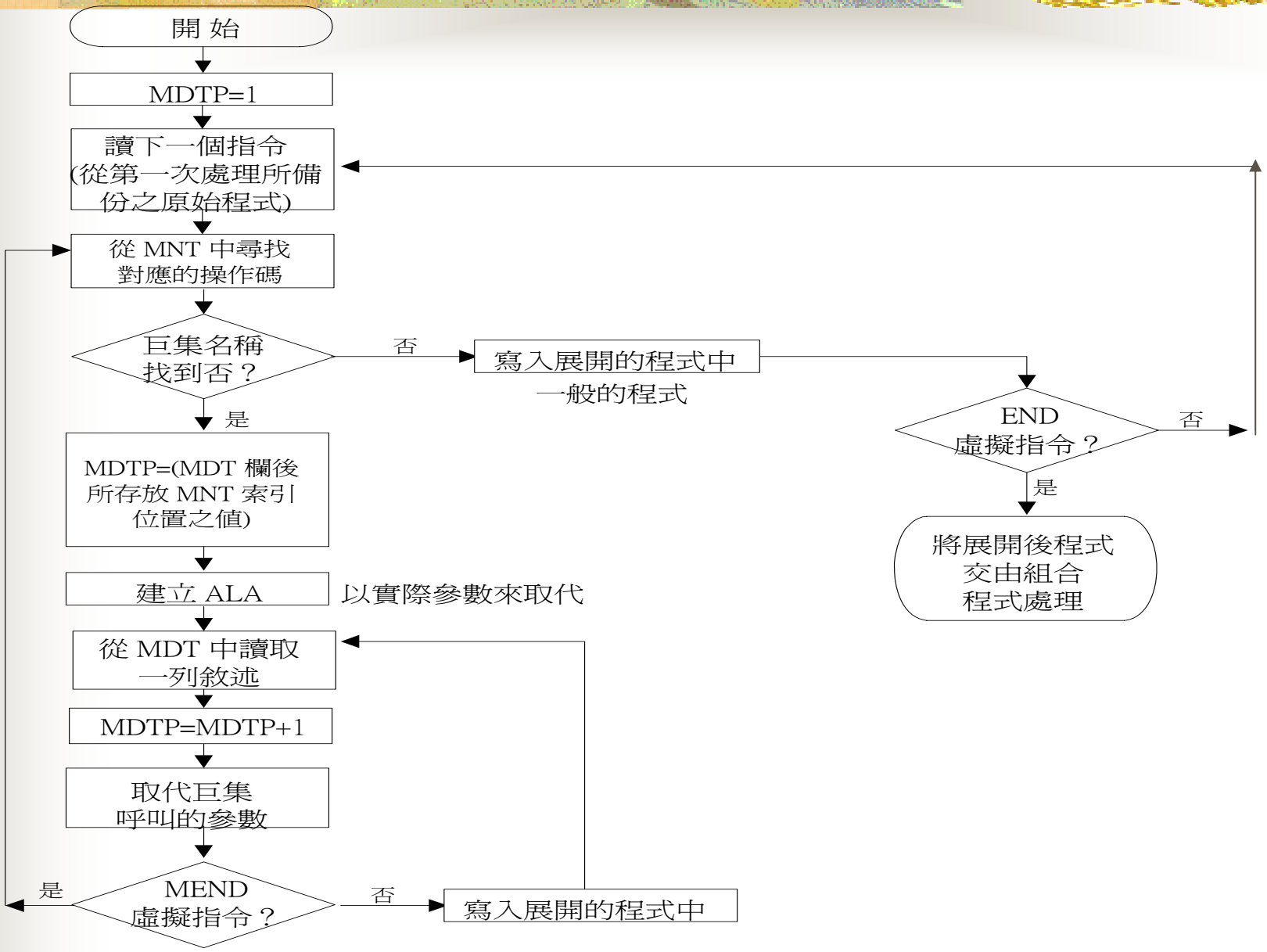
(b) 處理過程中使用到的資料結構：

- (i) 巨集定義表指標 (MDTP)－用以指向 MDT 中下一個要展開之巨集定義所在的位置。
- (ii) 參數陣列 (ALA)－將巨集指令後的實際參數依序填入參數陣列中，再把 MDT 中所有的虛擬參數以參數陣列中的實際參數取代之。

(c) 輸出：完全不含有巨集指令的組合語言程式。

- 處理過程：

  檢視程式中每一列操作碼是否已存在於 MNT 中，若該操作碼已儲存於 MNT 中，則設定巨集定義指標 (MDTP) 之值爲 MNT 中對應項之值；即是指向該巨集定義的開端處。接著將巨集呼叫後之實際參數依次填入參數陣列 (ALA) 中應對的位置中，再從 MDT 中讀取每一列敘述，並把所有之虛擬參數以 ALA 中之實際參數取代，直到遇到 MEND 才停止。繼續檢視下一列操作碼，直到處理到 END 時，則表示展開完畢。


開 始
MDTP=1
讀下一個指令
(從第一次處理所備
份之原始程式)
從 MNT 中尋找
對應的操作碼
巨集名稱
找到否？
否
寫入展開的程式中
一般的程式
END
虛擬指令？
否
是
將展開後程式
交由組合
程式處理
是
MDTP=(MDT 欄後
所存放 MNT 索引
位置之值)
建立 ALA
以實際參數來取代
從 MDT 中讀取
一列敘述
MDTP=MDTP+1
取代巨集
呼叫的參數
是
MEND
虛擬指令？
否
寫入展開的程式中

- 範例:

```
            M
            MARCO
            LOADING     &TAB,&VAR1,&VAR2,&VAR3
&TAB        L           1,&VAR1
            L           2,&VAR2
            L           3,&VAR3
            MEND
            M
            LOADING     LOOP1,ALPHA,BETA,GAMA
            M
            LOADING     LOOP2,BETA,GAMA,ALPHA
            M
ALPHA       DC          F'1'
BETA        DC          F'2'
GAME        DC          F'3'
            M
```

經二次處理巨集處理器處理，第一次處理與第二次處理所建立的表格與資料結構的過程如下：

- (1) 第一次處理：

  假設處理至此程式片斷時，先前已經處理過 6 個巨集定義，且此 6 個巨集定義之巨集主體共佔用了巨集定義表 (MDT) 34 個欄位，故目前之 MNTC 值爲 7，而 MDTC 值爲 35。

  - (a) 當遇到巨集定 LOADING，則將之置入 MNTC 口1。

巨 集 名 稱 表 (MNT)

索引位置	名稱	巨集定義表之索引位置
7	LOADING	35
M	M	M

- (b) 將虛擬參數依序儲存入 ALA 中。

參 數 陣 列 (ALA)

索引位置	參數
1	&TAB
2	&VAR1
3	&VAR2
4	&VAR3
M	M

- (c) 將巨集定義內之虛擬參數以相對應之索引位置取代之，並依目前之值將巨集主體逐條填入 MDT 中，每填入一條指令便將 MDTC 之值加 1。

巨 集 定 義 表 (MDT)

索引位置	指 令	
35	#1	L 1,#2
36		L 2,#3
37		L 3,#4
M		M

- (d) 產生不含有巨集定義之原始程式

```
          :
              LOADING LOOP1,ALPHA,BETA,GAMA
          :
              LOADING LOOP2,BETA,GAMA,ALPHA
          :
ALPHA     DC   F'1'
BETA      DC   F'2'
GAMA      DC   F'3'
```

- (2) 第二次處理：
  - (a) 於不含有巨集定義之原始程式中遇到巨集指令 LOADING LOOP1,ALPHA,BETA,GAMA，則將其後之實際參數依次取代 ALA 中之虛擬參數。

參數陣列 (ALA)

索引位置	參數
1	LOOP1
2	ALPHA
3	BETA
4	GAMA
..	..

- (b) 至 MNT 中搜尋後得知巨集指令 LOADING 之巨集主體是從 MDT 的第 35 欄位開始存放，將 MDT 中之指令逐一讀出，並將其虛擬參數以 ALA 中相對應索引位置之實際參數取代之。巨集展開如下：
- LOOP1 L 1 , ALPHA
- L 2 , BETA
- L 3 , GAMA

(二) 一次處理巨集處理器

- (1) 功能及限制：

  (a) 允許巨集定義內含有巨集定義，但是內層的巨集定義必須在外層的巨集定義被呼叫後，內層的巨集定義才能被呼叫。不可在主程式中直接呼叫內層的巨集定義。

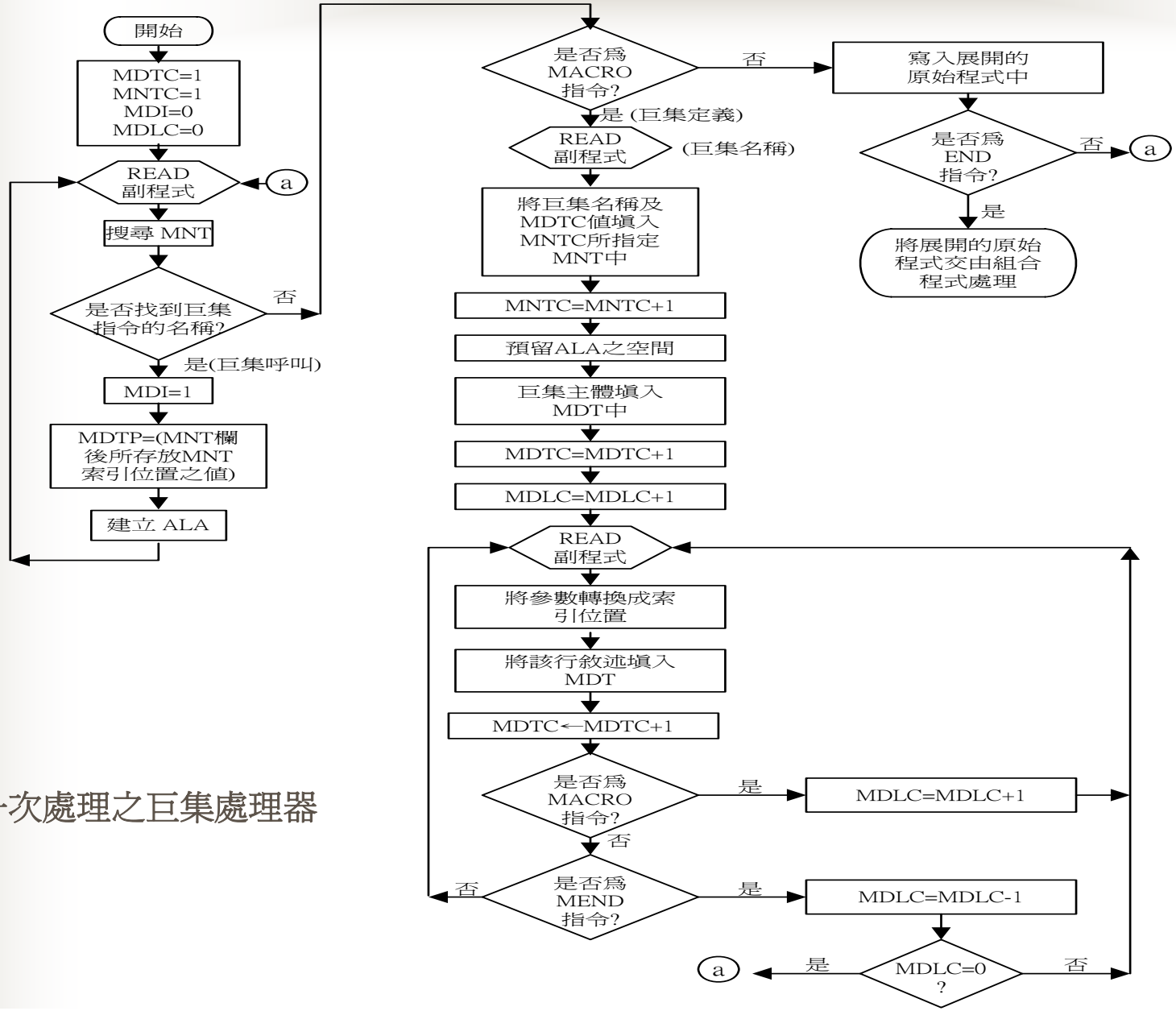
  (b) 巨集定義必須在巨集呼叫之前已定義妥善。

- (2) 使用到的資料庫：
  - (a) 輸入：原始程式。
  - (b) 處理過程中使用到的資料結構：
    - 巨集定義表 (MDT)。
    - 巨集名稱表 (MNT)。
    - 巨集定義表計數器 (MDTC)。
    - 巨集名稱表計數器 (MNTC)。
    - 參數陣列 (ALA)。
    - 巨集定義表指標 (MDTP)。
    - 巨集定義層數計數器 (MDLC)：用以確定整個巨集定義 (包括內層的巨集定義) 已被完整的儲存。當遇到 MACRO 指令時，MDLC 的值加 1；遇到 MEND 時，MDLC的值減1。
    - 巨集定義輸入指示器 (MDI)：用以指示目前是否正處於處理巨集展開。若是正處於展開巨集呼叫之情況，則 MDI 之值爲 1；否則 MDI 之值爲 0。

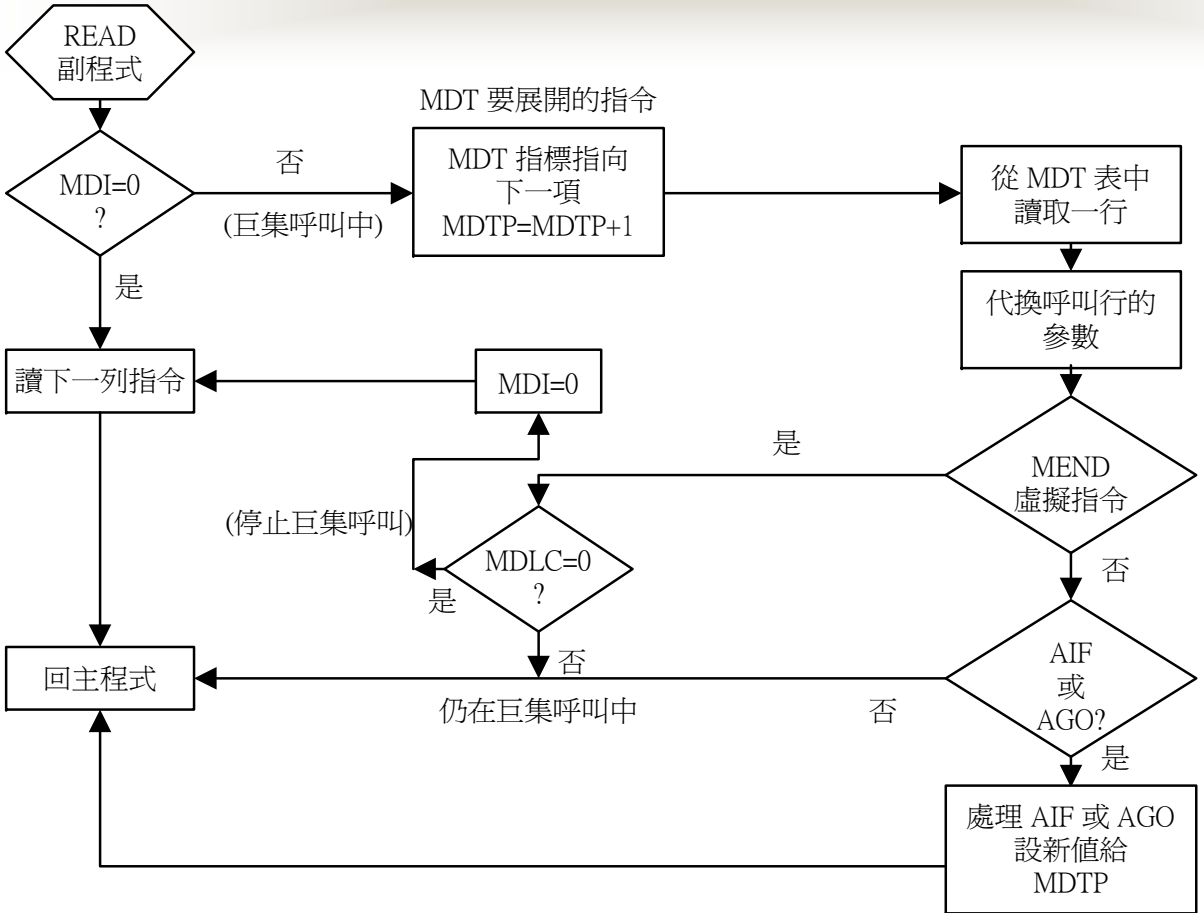
(c) 輸出：經巨集展開後之原始程式。

- (3) 處理過程：

  一次處理之巨集處理器爲了能夠順利的處理巢狀式巨集 (即巨集定義內含有巨集呼叫)，需額外使用兩個變數：巨集定義層數計數器 (MDLC) 與巨集定義輸入指示器 (MDI)。當在做巨集展開時，MDI 之值爲 1；否則，MDI 之值爲 0。當 MDI 之值爲 1 時且又遇到 MACRO 指令時，則 MDLC 之值加 1；遇到 MEND 指令時，則 MDLC 之值減 1。如此，才可確保整個巨集定義 (包括內層的巨集定義) 已被完整的儲存。


開始
MDTC=1
MNTC=1
MDI=0
MDLC=0
READ
副程式
a
搜尋 MNT
是否找到巨集
指令的名稱?
否
是(巨集呼叫)
MDI=1
MDTP=(MNT欄
後所存放MNT
索引位置之值)
建立 ALA
是否爲
MACRO
指令?
否
寫入展開的
原始程式中
是否爲
END
指令?
否
a
是
將展開的原始
程式交由組合
程式處理
是 (巨集定義)
READ
副程式
(巨集名稱)
將巨集名稱及
MDTC值填入
MNTC所指定
MNT中
MNTC=MNTC+1
預留ALA之空間
巨集主體填入
MDT中
MDTC=MDTC+1
MDLC=MDLC+1
READ
副程式
將參數轉換成索
引位置
將該行敘述填入
MDT
MDTC←MDTC+1
是否爲
MACRO
指令?
是
MDLC=MDLC+1
否
是否爲
MEND
指令?
否
是
MDLC=MDLC-1
a
是
MDLC=0
?
否

一次處理之巨集處理器



巨集展開中 **READ** 副程式的流程

- 範例：

```
MACRO
MAIN            &ROUTINE
MACRO
&ROUTINE        &X
CNOP            0,4
BAL             1,*+4
DC              A(&X)
L               15,V(&ROUTINE)
BALR            14,15
MEND
MEND
  M
MAIN            VALUE
VALUE           DATA
  M
```

就上面程式片斷經一次處理巨集處理器其所建立的表格與資料結構過程如下：

- 假設此程式片斷中僅具有 MAIN 此一巨集定義，故 MNTC 及 MDTC 目前之值皆爲 1。

- a. 當遇到巨集名稱 MAIN，則將之置入 MNTC 所指示之 MNT 欄位內，並將 MNTC 值加 1；即目前之 MNTC 之值變爲 2。

巨集名稱表 (MNT)

索引位置	名稱	巨集定義表之索引位置
1	MAIN	1
11	11	11

- b. 將虛擬參數依序儲存入 ALA 中。

參數陣列 (ALA)

索引位置	參數
1	&ROUTINE
M	M

- c. 將巨集定義 MAIN 內之虛擬參數以相對應的索引位置取代之；即將 &ROUTINE 以 #1 取代。並依 MDTC 目前之值將巨集主體逐條填入 MDT 中，每填入一條指令便將 MDTC 之值加 1。巨集定義完全填入後，MDTC 之值將變成 10。

巨 集 定 義 表 (MDT)

索引位置	指令
1	MACRO
2	#1 &X
3	CONP 0,4
4	BAL 1,*+4
5	DC A(&X)
6	L 15,=V(#1)
	BALR 14,15
8	MEND
9	MEND

- d. 當遇到巨集呼叫指令 “MAIN VALUE” 時，則設定 MDI 之值爲 1，並將 ALA 中之虛擬參數 &ROUTINE 以實際參數 VALUE 取代之。其 ALA 之內容如下：

參 數 陣 列 (ALA)

索引位置	參數
1	VALUE
M	M

- e. 由巨集呼叫指令 “MAIN VALUE” 中的 VALUE 亦是巨集定義，故須將之儲存於 MNTC 所指定之 MNT 欄位內，並將 MNTC 之值加 1。此時 MNTC 之值變爲 3。

巨 集 名 稱 表 (MNT)

索引位置	名稱	巨集定義表之索引位置
1	MAIN	1
2	VALUE	10
M	M	M

- f. 將巨集定義 VALUE 之虛擬參數依序填入，另一個 ALA 中，其內容如下：

參 數 陣 列 (ALA)

索引位置	參數
1	VALUE
2	&X

- g. 將巨集定義 VALUE 內之虛擬參數以相對應之索引位置取代之；即以 #2 取代 &X。並依 MDTC 目前之值將巨集主體逐條填入 MDT 中，每填入一條指令便將 MDTC 之值加 1。當巨集定義完全填入後，MDTC 之值變成 16，MDI 之值變爲 0。

巨 集 定 義 表 (MDT)

索引位置	指令
1	MACRO
:	:
8	MEND
9	MEND
10	CNOP 0,4
11	BAL 1,*+4
12	DC A(#2)
13	L 15,V(VALUE)
14	BALR 14,15
15	MEND

- h. 當遇到巨集呼叫指令 “VALUE DATA”，則將虛擬參數 &X 以實際參數 DATA 取代之。其內容如下：

參數陣列 (ALA)

索引位置	參數
1	VALUE
2	DATA
"	"

- i. 從 MNT 中得知巨集名稱 VALUE 的巨集主體於 MDT 中之儲存位置。將它的巨集定義從 MDT 中逐條讀出，並且將虛擬參數以 ALA 中相對應的實際參數取代。展開之程式片斷如下：

```
CNOP      0,4
BAL       1,*+4
DC A(DATA)
L         15,V(VALUE)
BALR      14,15
```

(三)巨集組合程式

- (1) 所謂巨集組合程式 (Macro Assembler) 乃是將巨集處理器合併於組合程式中。
- 前述之一次處理巨集處理器與二次處理巨集處理器皆可稱之爲前置處理器。因爲用它們來處理巨集定義與巨集展開，最後將展開後的原始程式供組合程式做進一步之處理。

- 若是將巨集處理器併入組合程式中，則可避免產生過多的中間檔案，減輕系統的負擔，並且可以把兩者間相似的功能整合在一起，如 MDT 可併入 MOT 或 POT中，只需用一個旗標來指示是否爲巨集名稱即可。巨集指令可視之爲組合語言內的虛擬指令，當處理到此指令時，再交由處理巨集的部分加以處理即可。

將巨集處理器併入組合程式中的第一次處理中有其優缺，其優缺點分述如下：

- ＜優點＞：

  a. 許多相似的功能可以整合在一起，不必再重覆執行。如讀入指令、檢查敘述的型態。

  b. 因不需要先產生中間檔案 (即展開之原始程式)，再由組合程式讀取該中間檔案做組譯的工作，故可以減少處理過程的負擔。

  c. 可把組合語言之特性加於巨集指令中，使程式設計更具有彈性。

- ＜缺點＞：

  a. 合併兩者使得組合程式變大，以致於無法一次將它完全置入主記憶體中。

  b. 兩者間的整合、協調非常不容易，以致設計起來會相當地複雜。

對於巨集組合程式處理巨集指令的方式，一般可分下列兩種：

- 方法一：每當讀入一列指令，則先去機器操作碼表 (MOT) 中搜尋，若沒有找到，則視之爲巨集指令，便再去巨集名稱表 (MNT) 中搜尋。
- 方法二：每當讀入一列指令，則先去巨集名稱表 (MNT) 中搜尋，若在巨集名稱表中沒找到時，才去機器操作碼表 (MOT) 中搜尋。

方法二建構巨集組合程式的優缺點。

- ＜優點＞：

  它允許程式師重新定義原先在組合語言中已存在的操作指令；即是對於某一操作指令，程式師可藉由巨集定義改變其原先之功能。即修改組合語言指令功能。

- ＜缺點＞：

  需耗費較多的搜尋時間。因爲程式中的巨集指令較一般的運算指令少，而每次讀入一列指令，皆先去巨集名稱表 (MNT) 中搜尋，當搜尋失敗時，再去機器操作碼表 (MOT) 中搜尋。對於一般的運算指令，無形中浪費許多時間在 MNT 中搜尋。