第2章 組譯器

機械語言

- 電腦的運作原理，就是CPU不斷向記憶體取得指令來執行，而這些指令都是二 進制的，即只有０和１，這形式的東西就是「機械語言」(Machine Code) 了 。
- MS-DOS 有一個名叫debug的程式，可以用來編寫機語。機械語言與記憶體及ASCII碼息息相關.

利用機器語言輸出 “Hello”

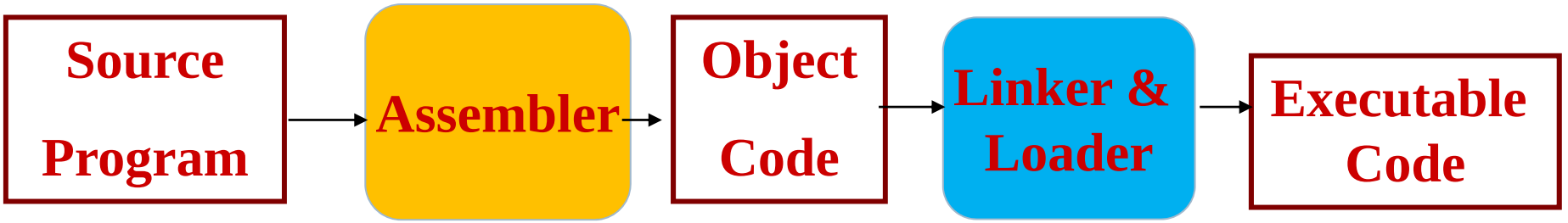
- C:\>debug
  -
- 執行後出現了一個減號，然後輸 e200 ：

- -E 200<按Enter鍵> (從偏移位址200開始。輸入 “Hello,World”)
  48<按空格>65<按空格> (輸入48h (H)和65h (e))
  6C<按空格>6C<按空格> (輸入6Ch (l)和6Ch (l))
  6F<按空格>2C<按空格> (輸入6Fh (o)和2Ch (,))
  57<按空格>6F<按空格> (輸入57h (W)和6Fh (o))
  72<按空格>6C<按空格> (輸入72h (r)和6Ch (l))
  64<按空格>24<按空格> (輸入64h (d)和24h ($))
  <按Enter鍵> (“Hello,World” 已經輸入完畢)
  -D 200<按Enter鍵> (顯示你剛剛輸入的內容：

組合語言

- 組合語言是把「機器指令」(或稱「機器碼」)表示成「文字指令」所構成的程式語言，使用組合語言所撰寫的程式需經Assembler(組譯器)將文字指令編譯成機器碼，所以組合語言可說是機械語言的進化．
-
- 執行時的順序:
- 組合語言 -----→ 組譯器 ------→ 機械碼
- MOV AH,09 (Assembler) 00010010

組譯器 Assemblers



利用組語來輸出 “Hello”

- C:\>debug
- -a100
- -A 100 <按Enter鍵> (用彙編語言寫一個新程序在IP-100h處開始)
  MOV AH,09 <按Enter鍵> (選項DOS的09號功能使用，顯示字串串)
  MOV DX,0200 <按Enter鍵> (把輸出位址(200h)，放進暫存器)
  INT 21 <按Enter鍵> (執行DOS功能使用，顯示"Hello,World")
  INT 20 <按Enter鍵> (結束程序回到DOS狀態)
  <按Enter鍵> (結束彙編語言輸入，回到DEBUG輸入狀態)
  -G <按Enter鍵>
  Hello,World
  Program terminated normally

- 組合語言所代表的是機器碼的文字指令，所以可以一對一地轉譯成機器碼，使其程式較為精簡，因此具備執行效能佳的優點，不過組合語言較難閱讀，也不好寫，因此隨著硬體及軟體技術的提升，也不斷有各種高階語言的誕生。

高階語言

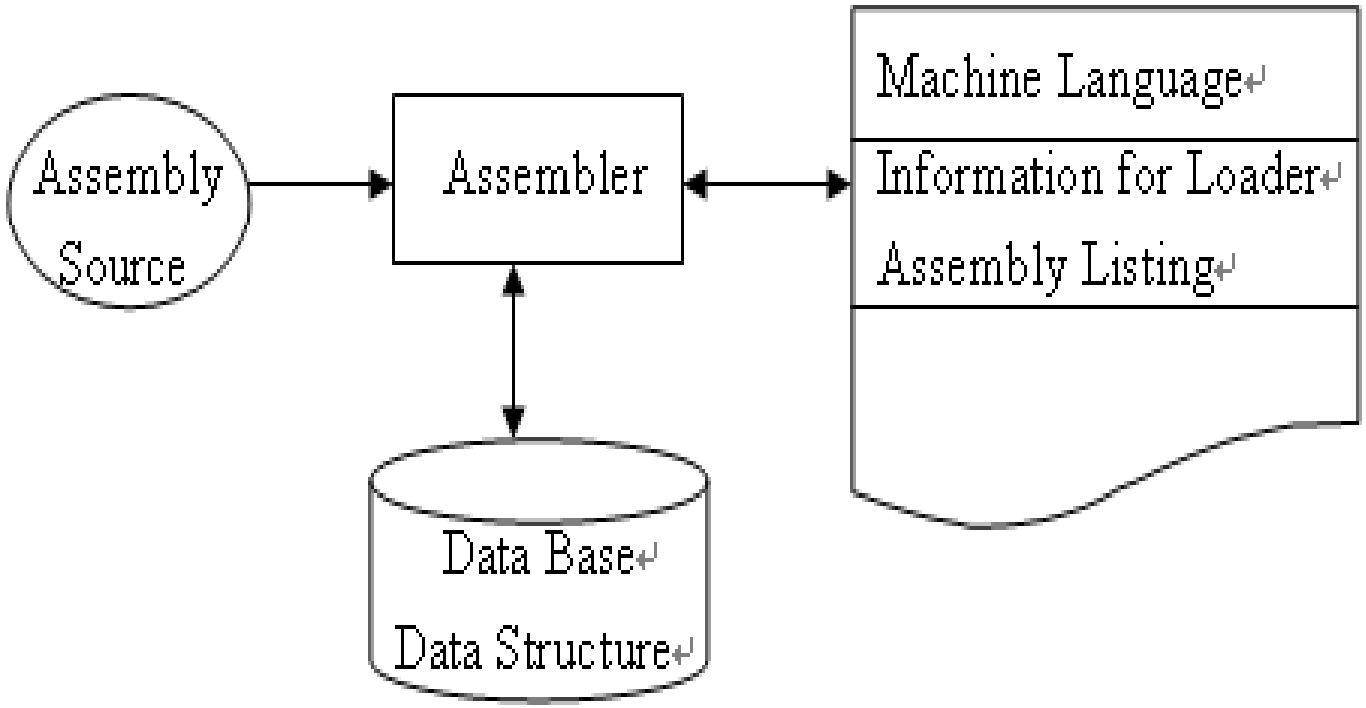
- 組合語言為低階語言，語言稱之為低階或高階在於其較接近機器碼或較接近人類語言，較接近機器碼的語言稱為低階語言，較接近人類語言的語言稱為高階語言。
- 高階語言以類似英文和數學的型態出現，將所能使用者可能用到的功能、函數等，全都內建在語言之中，使用者所需要的，高階語言中都有，由於有了編譯器(compiler)，使得程式語言有極大的進步，不必再使用艱澀難懂的機械語言，可以用類似人類語言的程式語言，來撰寫程式。

編 譯 器 (COMPILER)

- (1)編譯器就是一種能接受高階語言程式當作輸入,而產生等效的機械語言為輸出的軟體程式,例如 Turbo C, C++, Borland C++, Microsoft C, Visual C++ etc.
(2)編譯器基本上由語言分析(Lexical Analysis),語法分析(Syntax Analysis),編碼產生器(Code Generator),符號表(Symbol Table)所組成.

- 原始程式 --> 經由編譯器 --> 變成目的檔 --> 經由連結器 --> 變成執行檔
- **SOURCE.C --> COMPILER --> SOURCE.OBJ --> LINKER --> SOURCE.EXE**

組譯器之結構


Assembly
Source
Assembler
Machine Language
Information for Loader
Assembly Listing
Data Base
Data Structure

組譯器的基本功能

- 將助憶碼 (opcode, 組合語言指令) 轉換成機器碼
- 指定機器位址給符號標籤
- 組合語言 -> 目的碼 (object code)

組譯器- 簡介

- 組譯程式之類型
  - One-Pass Assembler(單回合組譯程式)
  - Two-Pass Assembler(雙回合組譯程式)
  - Multi-Pass Assembler(多回合組譯程式)
- 組譯程式之基本功能
  - 將助憶碼轉成對應之機器碼
  - 指定機器位址給所定義的符號標記(symbolic label)
  - 處理虛擬指令(假指令、組譯器指引命令)
  - 處理前向參考(forward reference)
  - 產生報表檔(listing)
  - 產生目的碼(object code)
- 簡單的SIC組譯程式之功能(參考圖2.1-2.2)
  - 將助憶碼轉成機器碼
  - 將符號運算元轉成相對之位址
  - 建立適當的機器碼格式
  - 將資料常數轉成機器內部表示式
  - 產生目的碼及報表檔

假指令(Assembler Directive) (1)

- (Pseudo-instruction , Assembler Directive)
- 並非機器程式語言的助記碼
- 乃告訴組合程式如何做組繹工作的指令。(亦即可改變組合程式本身的狀況)

假指令(Assembler Directive) (2)

- 並不會被轉入到 Object Program
  - Provide instructions to the assembler itself.
- 例如：
  - START
  - END
  - BYTE、WORD（宣告型態）
  - RESB、RESW（保留緩衝區）

內容

- 2.1節:型組譯器的基本動作
- 2.2節:一些擴充機制
- 2.3節:機器無關的組合語言特性和實作
- 2.4節:重要而不同的一些設計方法

2.1 基本組譯器功能

- 組譯器指引（assembler directives）
  - START 指定程式名稱和起始位址
  - END　指示原始程式的結束處，並指定程式中第一個可執行的指令
  - BYTE　定義字元或十六進位的常數，並且指出其可佔用之位元組的數量
  - WORD　定義一個字組的整數常數
  - RESB　保留所示數量的位元組，供資料區使用
  - RESW　保留所示數量的字組，供資料區使用

SIC組合語言程式的範例

```
  5        COPY        START     1000              COPY FILE FROM INPUT TO OUTPUT
 10        FIRST       STL       RETADR            SAVE RETURN ADDRESS
 15        CLOOP       JSUB      RDREC             READ INPUT RECORD
 20                    LDA       LENGTH            TEST FOR EOF (LENGTH = 0)
 25                    COMP      ZERO
 30                    JEQ       ENDFIL            EXIT IF EOF FOUND
 35                    JSUB      WRREC             WRITE OUTPUT RECORD
 40                    J         CLOOP             LOOP
 45        ENDFIL      LDA       EOF               INSERT END OF FILE MARKER
 50                    STA       BUFFER
 55                    LDA       THREE             SET LENGTH = 3
 60                    STA       LENGTH
 65                    JSUB      WRREC             WRITE EOF
 70                    LDL       RETADR            GET RETURN ADDRESS
 75                    RSUB                        RETURN TO CALLER
 80        EOF         BYTE      C'EOF'
 85        THREE       WORD      3
 90        ZERO        WORD      0
 95        RETADR      RESW      1
100        LENGTH      RESW      1                 LENGTH OF RECORD
105        BUFFER      RESB      4096              4096-BYTE BUFFER AREA
110        .
115        .           SUBROUTINE TO READ RECORD INTO BUFFER
120        .
125        RDREC       LDX       ZERO              CLEAR LOOP COUNTER
130                    LDA       ZERO              CLEAR A TO ZERO
135        RLOOP       TD        INPUT             TEST INPUT DEVICE
140                    JEQ       RLOOP             LOOP UNTIL READY
145                    RD        INPUT             READ CHARACTER INTO REGISTER A
150                    COMP      ZERO              TEST FOR END OF RECORD (X'00')
155                    JEQ       EXIT              EXIT LOOP IF EOR
160                    STCH      BUFFER,X          STORE CHARACTER IN BUFFER
165                    TIX       MAXLEN            LOOP UNLESS MAX LENGTH
170                    JLT       RLOOP               HAS BEEN REACHED
175        EXIT        STX       LENGTH            SAVE RECORD LENGTH
180                    RSUB                        RETURN TO CALLER
185        INPUT       BYTE      X'F1'             CODE FOR INPUT DEVICE
190        MAXLEN      WORD      4096
195        .
200        .           SUBROUTINE TO WRITE RECORD FROM BUFFER
205        .
210        WRREC       LDX       ZERO              CLEAR LOOP COUNTER
215        WLOOP       TD        OUTPUT            TEST OUTPUT DEVICE
220                    JEQ       WLOOP             LOOP UNTIL READY
225                    LDCH      BUFFER,X          GET CHARACTER FROM BUFFER
230                    WD        OUTPUT            WRITE CHARACTER
235                    TIX       LENGTH            LOOP UNTIL ALL CHARACTERS
240                    JLT       WLOOP               HAVE BEEN WRITTEN
245                    RSUB                        RETURN TO CALLER
250        OUTPUT      BYTE      X'05'             CODE FOR OUTPUT DEVICE
255                    END       FIRST
```

簡單的SIC組譯器

```
5    COPY     START     1000
10   FIRST    STL       RETADR
15   CLOOP    JSUB RDREC
20            LDA       LENGTH
25            COMP      ZERO
30            JEQ       ENDFIL
35            JSUB WRREC
40            J         CLOOP
```

簡單的SIC組譯器(1)

- 1.把助記碼(mnemonic code)轉換成機器碼(machine code)。
  - 如 STL －＞ 14 ； J －＞ 3C
- 2.把符號式運算元或符號式地址(Symbolic operand / Address)轉換成內部表示法或記憶地址。
  - 如 5 －＞0000 0000 0000 0000 0000 0101 ；
  - RETADR －＞1033H

簡單的SIC組譯器(2)

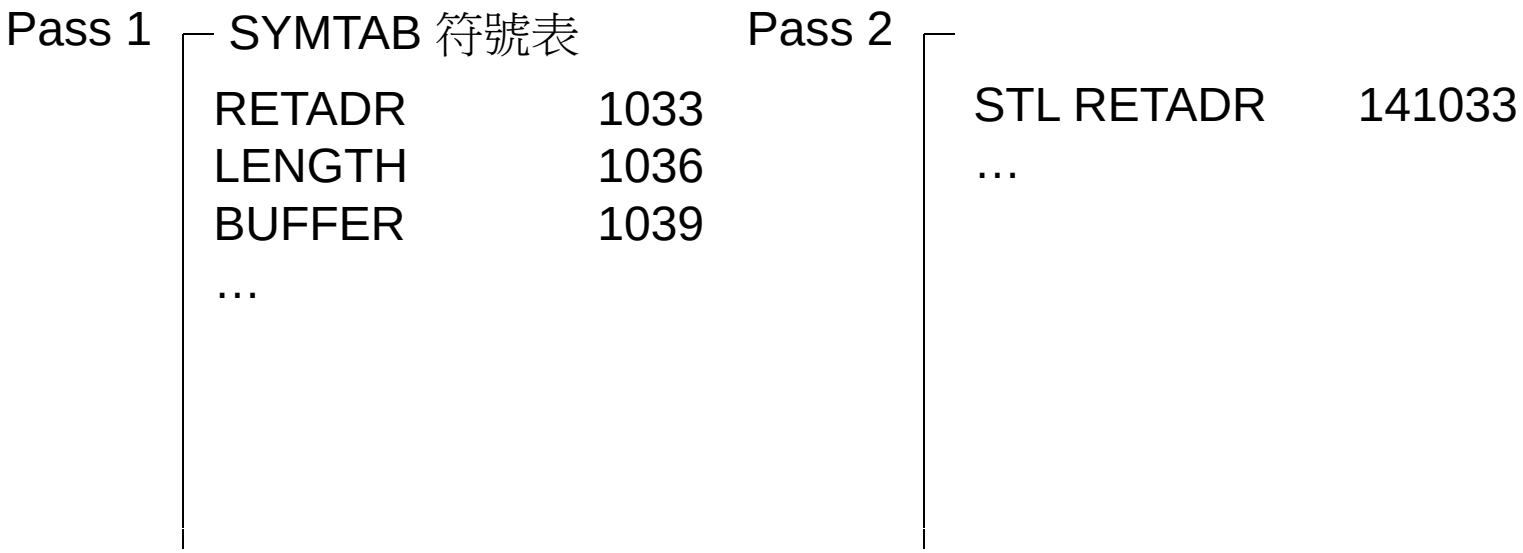
- 3.按照指令格式(Instruction Format)，把運算碼(operation code)和取址訊息(Address Information)組合成完整的機器語言指令。
- 4.把資料常數轉換成內部表示法。
  - 如　'EOF' －＞ 454F46H
- 5.產生列表程式及目的程式(*訂好格式*)
- 處理假指令 START、END、BYTE、RESW、RESB、WORD

簡單的SIC組譯器(3)

- 最後產出的 Object Program :
  - 即原始程式的機器語言碼。
  - 可以被載入程式(Loader)載入記憶體中等待執行。
  - Object Program 的結構如下頁所示。

簡單的SIC組譯器

- 組譯器需要掃描兩次程式, 第一次決定每一行的機器位址, 第二次產生目的碼



2.1.1 簡易的SIC組譯器

1. 將助憶碼轉換成對應的機器語言，例如將第10行的STL轉換成14。
2. 把符號運算元轉換成對應的機器位址，例如將第10行的RETADR轉換成1033。
3. 依適當的格式，建立機器指令。
4. 將原始程式內的常數資料，轉換成機器內部的表示方式，例如將第80行的EOF轉換成454F46。
5. 產生目的碼程式和組譯列表。

組譯圖2.1程式所產生之具目的碼的程式

```
  5        1000        COPY        START     1000
 10        1000        FIRST       STL       RETADR              141033
 15        1003        CLOOP       JSUB      RDREC               482039
 20        1006                    LDA       LENGTH              001036
 25        1009                    COMP      ZERO                281030
 30        100C                    JEQ       ENDFIL              301015
 35        100F                    JSUB      WRREC               482061
 40        1012                    J         CLOOP               3C1003
 45        1015        ENDFIL      LDA       EOF                 00102A
 50        1018                    STA       BUFFER              0C1039
 55        101B                    LDA       THREE               00102D
 60        101E                    STA       LENGTH              0C1036
 65        1021                    JSUB      WRREC               482061
 70        1024                    LDL       RETADR              081033
 75        1027                    RSUB                          4C0000
 80        102A        EOF         BYTE      C'EOF'              454F46
 85        102D        THREE       WORD      3                   000003
 90        1030        ZERO        WORD      0                   000000
 95        1033        RETADR      RESW      1
100        1036        LENGTH      RESW      1
105        1039        BUFFER      RESB      4096
110                    .
115                    .           SUBROUTINE TO READ RECORD INTO BUFFER
120                    .
125        2039        RDREC       LDX       ZERO                041030
130        203C                    LDA       ZERO                001030
135        203F        RLOOP       TD        INPUT               E0205D
140        2042                    JEQ       RLOOP               30203F
145        2045                    RD        INPUT               D8205D
150        2048                    COMP      ZERO                281030
155        204B                    JEQ       EXIT                302057
160        204E                    STCH      BUFFER,X            549039
165        2051                    TIX       MAXLEN              2C205E
170        2054                    JLT       RLOOP               38203F
175        2057        EXIT        STX       LENGTH              101036
180        205A                    RSUB                          4C0000
185        205D        INPUT       BYTE      X'F1'               F1
190        205E        MAXLEN      WORD      4096                001000
195                    .
200                    .           SUBROUTINE TO WRITE RECORD FROM BUFFER
205                    .
210        2061        WRREC       LDX       ZERO                041030
215        2064        WLOOP       TD        OUTPUT              E02079
220        2067                    JEQ       WLOOP               302064
225        206A                    LDCH      BUFFER,X            509039
230        206D                    WD        OUTPUT              DC2079
235        2070                    TIX       LENGTH              2C1036
240        2073                    JLT       WLOOP               382064
245        2076                    RSUB                          4C0000
250        2079        OUTPUT      BYTE      X'05'               05
255                    END         FIRST
```

簡單的SIC組譯器

Line	Loc				Object code
5	1000	COPY	START	1000	
10	1000	FIRST	STL	RETADR	141033
15	1003	CLOOP	JSUB	RDREC	482039
20	1006		LDA	LENGTH	001036
25	1009		COMP	ZERO	281030
30	100C		JEQ	ENDFIL	301015
35	100F		JSUB	WRREC	482061
40	1012		J	CLOOP	3C1003
		…			
80	102A	EOF	BYTE	C’EOF’	
85	102D	THREE	WORD	3	
90	1030	ZERO	WORD	0	
95	1033	RETADR	RESW	1	
100	1036	LENGTH	RESE	1	
105	1039	BUFFER	RESB	4096	

2.1.1 簡易的SIC組譯器

- 一個簡易目的程式的格式中，包括三種記錄：表頭（Header）、本文（Text）、結束（End）。

2.1.1 簡易的SIC組譯器

- 表頭記錄：
  - 欄 1　　H
  - 欄 2－7　程式名稱
  - 欄 8－13 目的程式的起始位址（16進位值）
  - 欄 14－19　　目的程式的長度，以位元組為單位

2.1.1 簡易的SIC組譯器

- 本文記錄：
  - 欄 1　　T
  - 欄 2－7　此記錄之目的碼的起始位址
  - 欄 8－9　此記錄之目的碼的長度(位元組)
  - 欄 10－69　　目的碼，以16進位表示

2.1.1 簡易的SIC組譯器

- 結束記錄：
  - 欄 1　　E
  - 欄 2–7　目的程式中第一個可執行指令的位址

Header Record

H	Program Name	Starting Add	Length	

Text Record

Starting Add in this record

Object Code

End Record

Address of First Executable instruction

目的程式

- 產生目的碼的格式

HCOPY 00100000107A

T0010001E141033482039001036…

T00101E150C10364820610810 33…

T0020391E041030001030E0205D…

…

E001000

2.1.1 簡易的SIC組譯器

- 對應圖2.2的目的程式

```
HCOPY  00100000107A
T0010001E14103348203900103628103030101548206 13C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D28103030205754903 92C205E38203F
T0020571C1010364C0000F100100041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000
```

目的程式

- 程式長度 = 207A-1000 = 107A (見圖2.2, 2.3)
- 並沒有1033-2038間的目的碼, 這部份由載入器 (loader, 第三章) 保留, 供程式執行時使用

Forward Reference

- A reference to a label that is defined later in the program
  - 1000 FIRST STL RETADR 141033
  - 1033 RETADR RESW 1
- 解決方法：Two Pass scanning.
  - Pass 1：Define symbols.
  - Pass 2：Assemble instructions and generate object program.
- Two Passes Assembler 演算法 (2.1.2)

2.1.1 簡易的SIC組譯器-二階段之簡易組譯器

- 第一階段（定義符號）

1. 為程式中的所有指令設置其位址
2. 記載程式中所有標記符號的值（位址），以供第二階段處理之用
3. 處理組譯器指引（此項處理會影響位址配置，如決定 BYTE、RESW 所定義之資料段的長度）

2.1.1 簡易的SIC組譯器-二階段之簡易組譯器

- 第二階段（組譯指令並產生目的程式）

1. 組譯指令（將指令轉譯成機器碼，並尋找位址）
2. 產生 BYTE、WORD 所定義的資料值
3. 處理在第一階段尚未完成之組譯器指引
4. 輸出目的程式和組譯器列表

2.1.2組合語言與資料結構

- 組譯器使用了兩個內部資料結構運算碼表 OPTAB 和符號表SYMTAB
- 運算碼表的作用是紀錄運算碼
- 符號表的作用是儲存給標籤的位址
- 另外還需要位置計數器 LOCCTR, 這是用來協助指派位址的變數, 其會被出使化為 START 敘述的起始位址

運算碼表 OPTAB

- 由許多Entry組成。每一可能包含下列特性：助記碼,機器碼,長度,格式, ……
- 必須至少含有助憶碼和其機器碼, 更複雜的組譯器則會含有更多資訊 (例如指令長度)
- 通常是雜湊 (HASH) 表格, 此表的資訊在組譯器建置時就定義好了, 而不是執行時才載入
- 通常是靜態表格, 意思是不會新增或刪除資料

2.1.2 組譯器演算法與資料結構

- OPTAB是用來查詢助憶碼，並將其轉譯成對應的機器語言
  - 在第一階段時，OPTAB是用來檢查原始程式碼中的指令是否正確；而在第二階段中，則是利用OPTAB將指令轉換成機器語言
- SYMTAB是用來存放程式中標記符號的值（位址）
- 程式位置計數器（Location Counter）LOCCTR

符號表 SYMTAB

- 由許多Entry(Record)組成。每一Entry可能包含下列特性：
  - 符號名稱
  - 符號的值(可能是地址的值,也可能是資料上的常數值)
  - 符號的長度(佔幾個Byte)
  - 符號的特性(跟Run Time 程式所存放的位置有關或無關)
- 儲存標記(Label)的名稱及位址值、錯誤旗標(重複定義)、含標記的資料區或指令的型態及長度等資訊
- 在各pass中的用途
  - Pass 1中將標記符號及指定位址(由LOCCTR指定)填入SYMTAB中
  - Pass 2中將標記符號之值插入機器碼中，形成目的碼

符號表 SYMTAB

- 標示錯誤的旗標 (例如同一個標籤定義兩次)
- 組譯器第一次執行時, 會將其標籤及位址 (由位置計數器取得) 輸入符號表
- 組譯器第二次執行時, 若碰到標籤符號時, 會在符號表中尋找其位址以供插入

位置計數器(Location Counter)

- PC: Run Time,下一個要執行的指令在哪裡？
- LOCCTR:
  - 指定位址之變數(variable) ,用來count下一個指令或變數的記憶體位址
  - 處理每一敘述後，將組譯後的指令或資料區之長度加到LOCCTR
  - At Assembly Time,組合程式組譯出來的下一個Byte,要放在哪裡
  - 它同時可用來統計整個Object Program共佔幾個Bytes

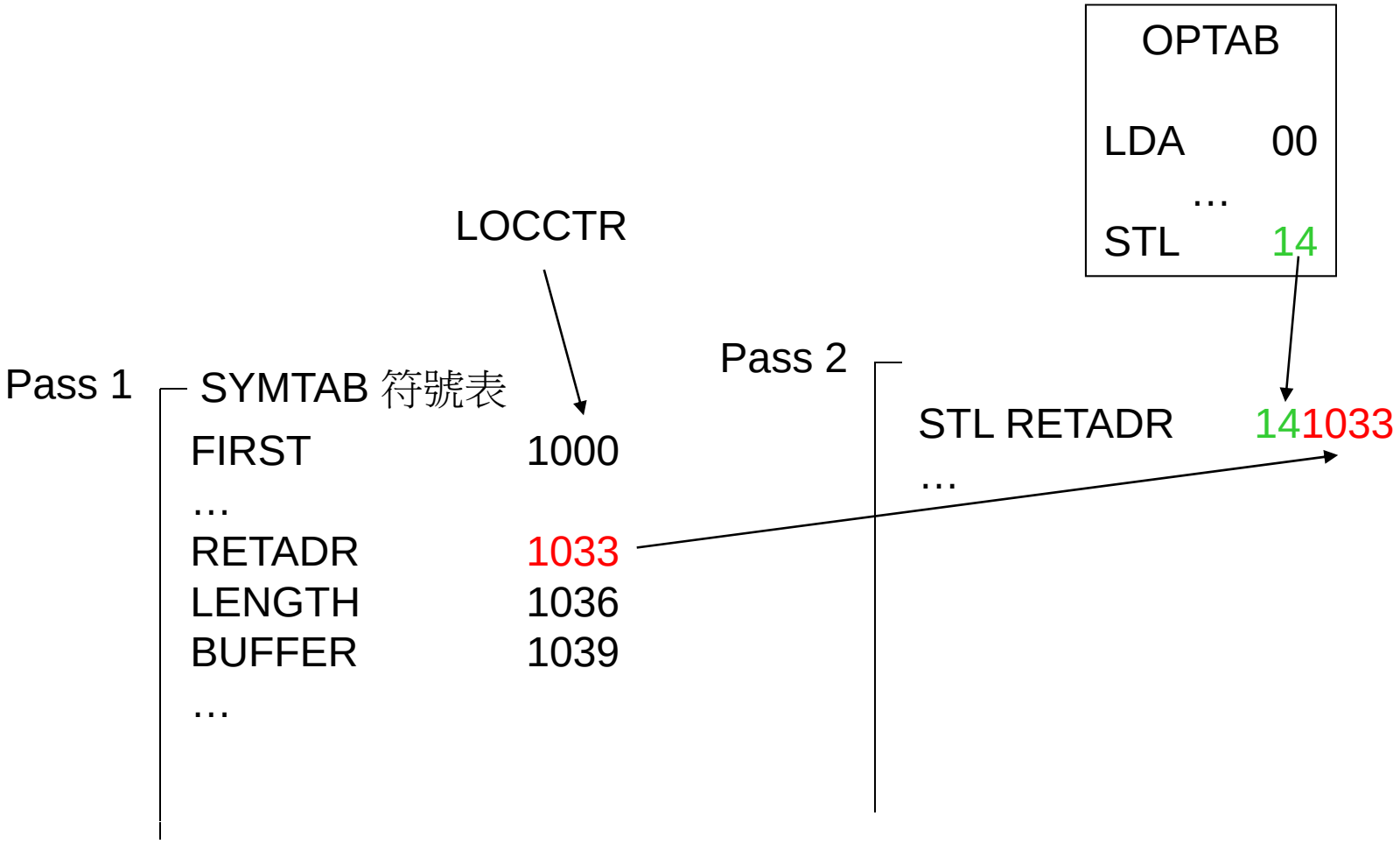
其他資料結構

- 一些Buffer(Array of Characters)
  - Object Record: 每一 Record 對應object file 一個 line:
    - SourceLine
    - ListLine/Intermediate Line
- 其他

其他設計上的考慮因素

- 是Free Format 或 Fixed Format?
- 每一回合的輸入是什麼?
- 尚需考慮Team Work , Load Balance Replication , Communication……

範例 – 組譯器的運作


OPTAB
LDA 00
…
STL 14
LOCCTR
Pass 1
SYMTAB 符號表
FIRST 1000
…
RETADR 1033
LENGTH 1036
BUFFER 1039
…
Pass 2
STL RETADR 141033
…

Two Passes Assembler (1)

- Pass 1:
  - Assign address to all statements in the program.
  - Save values (addresses) assigned to all labels for use in Pass 2.
  - Perform some processing of assembler directives.

Two Passes Assembler (2)

- Pass 2:
  - Assemble instructions (translating operation codes and looking up addresses).
  - Generate data values defined by BYTE, WORD, etc.
  - Perform processing of assembler directives not done during pass 1.
  - Write the object program and the assembly listing.

2 pass 組譯程式演算法
Pass 1之演算法

```
begin
讀取第一行程式
If  OPCODE='START' then
    begin
      將#[operand]存為起始位址
      LOCCTR = #[operand]
      將該行寫至中間檔(intermediate file)
      讀取下一行
    end
  else
    LOCCTR = 0
  while OPCIDE ≠ 'END' do
    begin
      if 不是註解行  then
        begin
          if 標記行欄有定義符號 then
            begin
              到SYMTAM中尋找該標記
              if 找到 then
                設定錯誤旗標(重複符號)
              else
                將(LABEL、LOCCTR)插入SYMTAB
            end
          到OPTAB中找尋運算碼
          if 找到
            LOCCTR = LOCCTR + 3 (指令長度)
          else if  OPCODE = 'WORD' then
            LOCCTR = LOCCTR + 3
          else if  OPCODE = 'RESW' then
            LOCCTR = LOCCTR + 3 * #[operand]
          else if  OPCODE = 'RESB' then
            LOCCTR = LOCCTR +  #[operand]
          else if  OPCODE = 'BYTE' then
            begin
              找尋常數之位元組數目(長度)
              LOCCTR = LOCCTR + 數目
            end
          else
            設定錯誤旗標(無效之運算碼)
        end(不是註解行)
      將該行寫入中間檔
      讀取下一行
    end(while)
  將最後一行寫入中間檔
  儲存程式長度(LOCCTR – 起始位址)
end(Pass 1)
```

2 pass 組譯程式演算法

- **Pass 2之演算法**

```
begin
讀取第一行程式(從中間檔)
If  OPCODE='START' then
   begin
     將該行寫入列表檔
     讀取下一行
   end
  else
  將表頭記錄寫入目的程式
①  開始第一個本文記錄
②  while OPCODE ≠ 'END' do
    begin
     if 不是註解行  then
③      begin
        到OPTAB找尋運算碼
④       if 找到 then
          begin
           if 運算元欄為符號 then
            begin
              到SYMTAB中尋找該符號
               if 找到 then
                將符號值存為運算元位址
⑤             else
                begin
                 將0存為運算元位址
⑤                設定錯誤旗標(未定義之符號)
④               end
              end(if symbol)
            else
              將0存為運算元位址
③           組譯成目的碼
          end(if opcode found)
        else if OPCODE = 'BYTE' or 'WORD' then
         轉換常數為目的碼
        if 目的碼不符合目前本文記錄
          begin
            將本文記錄寫入目的程式
            開始另一新的本文記錄
          end
②       將目的碼加入本文記錄
      end(不是註解行)
     將該行寫入列表檔
①    讀取下一行
   end(while not END)
  將最後本文資料寫入目的程式
  將結尾記錄寫入目的程式
  將最後一行寫入列表檔
end(Pass 2)
```

第一階段-1

```
begin
  read first input line
  if OPCODE = 'START' then
     begin
        save #[OPERAND] as starting address
        initialize LOCCTR to starting address
        write line to intermediate file
        read next input line
     end {if START}
  else
     initialize LOCCTR to 0
  while OPCODE ≠ 'END' do
     begin
        if this is not a comment line then
           begin
              if there is a symbol in the LABEL field then
                 begin
                    search SYMTAB for LABEL
                    if found then
                       set error flag (duplicate symbol)
                    else
                       insert (LABEL,LOCCTR) into SYMTAB
                 end {if symbol}
              search OPTAB for OPCODE
              if found then
                 add 3 {instruction length} to LOCCTR
```

第一階段-2

```
                else if OPCODE = 'WORD' then
                    add 3 to LOCCTR
                else if OPCODE = 'RESW' then
                    add 3 * #[OPERAND] to LOCCTR
                else if OPCODE = 'RESB' then
                    add #[OPERAND] to LOCCTR
                else if OPCODE  = 'BYTE' then
                    begin
                        find length of constant in bytes
                        add length to LOCCTR
                    end {if BYTE}
                else
                    set error flag (invalid operation code)
            end {if not a comment}
        write line to intermediate file
        read next input line
    end {while not END}
  write last line to intermediate file
  save (LOCCTR - starting address) as program length
end {Pass 1}
```

第二階段-1

```
begin
  read first input line {from intermediate file}
  if OPCODE = 'START' then
     begin
        write listing line
        read next input line
     end {if START}
  write Header record to object program
  initialize first Text record
  while OPCODE ≠ 'END' do
     begin
        if this is not a comment line then
           begin
              search OPTAB for OPCODE
              if found then
                 begin
                    if there is a symbol in OPERAND field then
                       begin
                          search SYMTAB for OPERAND
                          if found then
                             store symbol value as operand address
                          else
                             begin
                                store 0 as operand address
                                set error flag (undefined symbol)
                             end
                       end {if symbol}
```

第二階段-2

```
                    else
                        store 0 as operand address
                    assemble the object code instruction
                end {if opcode found}
            else if OPCODE = 'BYTE' or 'WORD' then
                convert constant to object code
            if object code will not fit into the current Text record then
                begin
                    write Text record to object program
                    initialize new Text record
                end
            add object code to Text record
        end {if not comment}
        write listing line
        read next input line
    end {while not END}
    write last Text record to object program
    write End record to object program
    write last listing line
end {Pass 2}
```

2.2 與機器有關的組譯器功能

- 主要是進行SIC/XE機器組譯器的設計與實做
- 檢視擴充機器的硬體和功能的成效

SIC/XE 不同於 SIC (1)

- 立即定址型式(Immediate Addressing Mode)
- 間接定址型式(Indirect Addressing Mode)
- PC相對定址型式(Program counter relative)

SIC/XE 不同於 SIC (2)

- 基底相對定址型式(Base relative)
- 暫存器－暫存器指定
- 採用格式四的指令

Why ?

- 為了增加程式執行的速度。
  - 指令變短了，擷取指令的時間減少。
  - 運算元已經在指令本身了，無須再做 Memory Read 以取出運算元。
  - 運算元已經在CPU的暫存器內，Access Time較短。
  - 可能避免另一指令的執行。

SIC/XE 不同於 SIC (3)

- SIC/XE有較大的主記憶體空間，意味者可能會有足夠的空間以同時載入和執行數個程式。這種數個程式共享機器資源的操作使用狀況稱為Multi-programming 。
- Multiprogramming的操作方式，可提高機器的使用效率。但是，程式究竟載入記憶體的什麼地方，並非組一時指明固定的位置，而是將執行的狀況，動態決定的，因此需要有重新定址的觀念。
- 組譯時目的程式的初始定址不等於執行時Core Image的初始定址！

SIC/XE Assembler

- 新增符號：
  - @：Indirect Addressing.
  - #：Immediate Operand.
  - +：Format 4 Instruction.
- 定址模式的考量：
  - 若無 BASE 宣告，則優先考慮 PC Relative
  - 若 PC Relative 無法使用
    （如 disp 超過範圍）則使用 Base Relative

2.2 與機器相關的組譯器特性

- 「間接定址」是在運算元之前加一個「@」符號來表示
- 「立即運算元」則前置一個「#」符號來表示
- 引用到記憶體:使用「程式計數器相對模式」或「基底相對模式」
- 大多數的「暫存器到記憶體」（register-to-memory）指令，是使用「程式計數器相對位址」或「基底相對定址」來進行組譯。
- 「基底相對定址模式」下，位移值只能由0到4095；而在「程式計數器相對定址模式」下，位移值是 –2048 到 +2047 之間。
- 定此類的定址模式。

2.2.1 指令格式和定址模式

- 如果位移需要「程式計數器相對模式」及「基底相對位址模式」時，則因所需空間太大而無法適合於3位元組指令，必須使用4位元組的擴展格式
- 在擴展指令格式上，原始程式碼的運算碼前必須加一個「+」的前置符號
- 程式設計師必須自行指

SIC/XE 指令

- 有兩種：
  - Register-to-Register Instruction
    - (ex) COMPR A, S　　(A004)
    - (解析) COMPR = A0,
      Register A = 0,
      Register S = 4.
  - 一般的 Register-to-Memory Instruction
    - Disp = Target Addr – PC 或 Target Addr - Base

如何組譯 R-R 指令?

- 如 line 125, 130, 132, 150, 210, 235……
- **(Step 1)**查表把助記碼換成機器碼
- **(Step 2)**把暫存器的符號換成代號
  - 可借助 Symbol Table，事先塞入這些暫存器的符號

使用定址型式的原則

- 盡量採用相對定址型式
  - PC relative
  - Bass relative
- 如果範圍太遠，才採用格式四的指令。
- 組譯格式四的指令，並不造成問題。
  - Simple, Simple Indexed, Immediate, Indirect

機器相關(Machine-dependent)之特性

- SIC/XE之製作
  - 定址模式
    - 間接定址：在***運算元***之前加上@表示
      - 圖2-6第70行: J @RETADR 3E2003
    - 立即定址：在***運算元***之前加上＃表示
      - 圖2-6第12、25、55、133行: CMP #0 290000
    - 相對定址：
      - 相對基底:位移量:0 ~ 4095
        - 由programmer用**BASE**宣告使用基底相對定址：BASE LENGTH
        - NOBASE用以解除暫存器B作為基底相對定址
      - 相對PC:由組譯器自行決定，位移量:-2048 ~ 2047
      - 格式四(4 bytes，需由programmer指定):若基底/PC相對定址無法定址時使用
        1.在***運算碼***前加上以＋表示
        圖2-6第15、35、65、133行: +LDT #4096 75101000
    - 組譯器決定定址模式之檢查順序: 格式四 ➔ PC相對 ➔ 基底相對 ➔錯誤
  - 優點：增進程式效率
    - 速度較快：使用暫存器存取資料
    - 程式較小：較多的定址模式

機器相關(Machine-dependent)之特性

- 範例：圖2.5 ~ 2.6

  ❶10 0000 FIRST STL RETADR 17202D ; 140030 (?)

  140030 = 0001 01*00 0000* 0000 0011 0000

  組譯器採用PC相對定址，且為SIC/XE指令，故 n=i=1，p=1

  TA=(PC) + disp ，而PC=003、TA=030 ➜ disp=030 – 003=002D

  ∴目的碼=0001 0111 0010 0000 0010 1101 = 17202D

  ❷40 0017 J CLOOP 3F2FEC ; 3C0006 (?)

  3C0006 = 0011 11*00 0000* 0000 0000 0110

  組譯器採用PC相對定址，且為SIC/XE指令，故 n=i=1，p=1

  TA=(PC) + disp ，而PC=01A、TA=006 ➜ disp=006 – 01A=FEC (?)

  ∴目的碼=0011 1111 0010 1111 1110 1100 = 3F2FEC

  ❸ 160 104E STCH BFFER,X 57C003 ; 540036 (?)

  組譯器採用基底相對定址，且為SIC/XE指令，故 n=i=1，b=1，x=1

  TA=(B) + disp ，而B=033、TA=036 ➜ disp=036 – 003=003

  ∴目的碼=0101 0111 1100 0000 0000 0011 = 57C003

機器相關(Machine-dependent)之特性

- ❹ 55 0020 LDA #3 010003
- ❺ 133 103C +LDT #4096 75101000
- ❻ 12 0003 LDB #LENGTH 69202D

SIC/XE 程式片段範例

```
 5     0000    COPY     START   0
10     0000    FIRST    STL     RETADR          17202D
12     0003             LDB     #LENGTH         69202D
13                      BASE    LENGTH
15     0006    CLOOP    +JSUB   RDREC           4B101036
20     000A             LDA     LENGTH          032026
25     000D             COMP    #0              290000
30     0010             JEQ     ENDFIL          332007
35     0013             +JSUB   WRREC           4B10105D
40     0017             J       CLOOP           3F2FEC
45     001A    ENDFIL   LDA     EOF             032010
50     001D             STA     BUFFER          0F2016
55     0020             LDA     #3              010003
60     0023             STA     LENGTH          0F200D
65     0026             +JSUB   WRREC           4B10105D
70     002A             J       @RETADR         3E2003
80     002D    EOF      BYTE    C'EOF'          454F46
95     0030    RETADR   RESW    1
100    0033    LENGTH   RESW    1
105    0036    BUFFER   RESB    4096
110                     .
```

2.2.2 程式重定位

- 希望許多個程式能夠同時共享一部電腦的記憶體及其他資源。
- 55 101B LDA THREE 00102D
- 在圖2.3的目的碼程式中，此指令是轉譯成00102D，而且位址 102D的內容將載入到A暫存器中。

2.2.2 程式重定位

- 當組譯器要產生JSUB指令的目的碼時，必須先插入「RDREC的位址是相較於程式起始位址」的資訊（這就是把位址計數器的初始值設為0的原因）。
- 組譯器也會為載入器產生一些指令，指示在載入器時，將起始位址加到JSUB指令的位址欄上。

機器相關(Machine-dependent)之特性

- 程式重定位(Program Relocation)-multiprogramming
  - 絕對位址程式(Absolute program)
    - 程式之起始位址於組譯時即決定，且每次載入之執行位址都相同
    - 其缺點為
      - 記憶體無法有效利用與共享(share)
      - 記憶體中之目的碼不易辨別其值為位址(address)或資料(data)
  - 可重定位程式(Relocatable program)
    - 程式之起始位址於載入時決定，且每次載入之位址未必相同
    - 包含由組譯器註明需作位址修正的資訊之目的程式
  - 組譯器產生可重定位程式之目的碼時，會加入一些資訊(修正記錄)供載入程式作為調整位址用(加入程式之起始位址)

程式重定位(Relocation) (1)

- 為什麼要重新定址(Relocation)?
- 組譯程式不知道程式將載入的實際位置，因此，它對程式所使用的位置也無法做必要的改變。不過，組譯程式卻能替載入程式看出目的程式中哪裡需要修改。
- 若目的程式中包含此種修改所要的資訊，我們稱為re-locatable object program

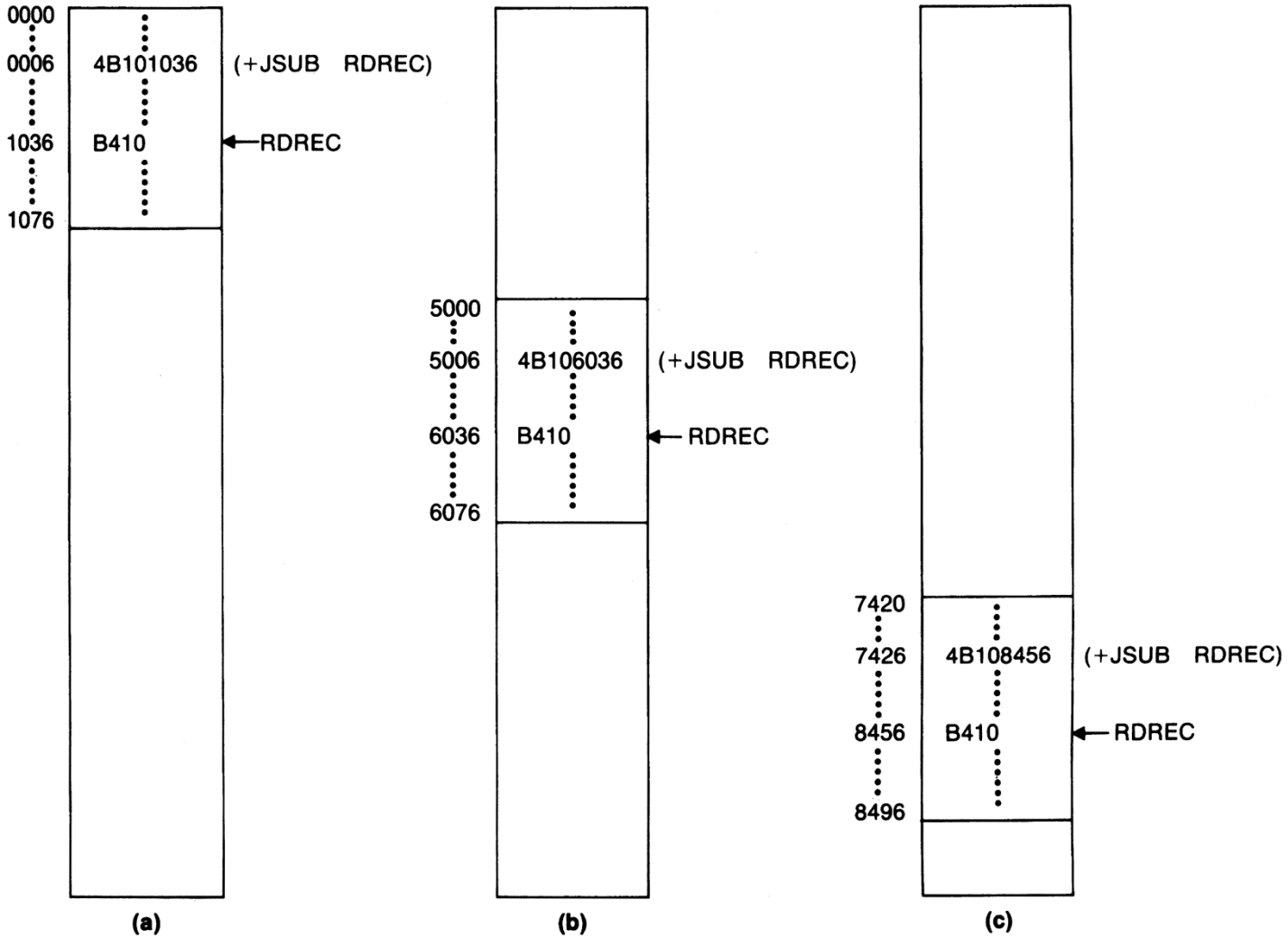
機器相關(Machine-dependent)之特性

- 修正記錄(Modification record)
  - 修正欄之長度以半位元組(half-byte)為單位(Not for all machines)

範例：課本圖**2.7, 2.8**

```
0006   CLOOP   +JSUB   RDREC   4B101036          M 000007 05
0013           +JSUB   WRREC   4B10105D          M 000014 05
0026           +JSUB   WRREC   4B10105D          M 000027 05
```

圖2.7　程式重定址的範例



修正記錄

- 欄 1M
- 欄 2－7　需要修改之位址欄的起始位址，此位址乃相對於程式的起始處（16進制）
- 欄 8－9　需要修改之位址欄的長度，以半位元組表示（16進制）
- 以JSUB指令為例，其修改記錄如下：
- M00000705

圖2.8　對應圖2.6的目的程式

```
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200 74B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B41077 4000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E000000
```

2.3 與機器無關的組譯器特性

- 2.3.1 節中，將討論組譯器如何處理「常值」（literal
- 2.3.2節中，討論二個組譯器指引（EQU及ORG）
- 2.3.3節簡介組合語言中的運算式，並且討論不同型態運算式及其運算和用法。
- 2.3.4節及2.3.5節，將介紹程式區段（program block）和控制段（control section）二項重要的主題。

2.3.1 常值

- 在程式中到處定義常數，並且可以賦予常數一個標記（label）。此種運算元稱之為「常值」（literal）。
- 45　001A　ENDFIL　LDA　=C'EOF'　032010
- 215　1062　WLOOP　TD　=X’05’　E32011

圖2.9 展示更多組譯器特性的程式範例

```
 5        COPY       START    0                 COPY FILE FROM INPUT TO OUTPUT
10        FIRST      STL      RETADR            SAVE RETURN ADDRESS
13                   LDB      #LENGTH           ESTABLISH BASE REGISTER
14                   BASE     LENGTH
15        CLOOP      +JSUB    RDREC             READ INPUT RECORD
20                   LDA      LENGTH            TEST FOR EOF (LENGTH = 0)
25                   COMP     #0
30                   JEQ      ENDFIL            EXIT IF EOF FOUND
35                   +JSUB    WRREC             WRITE OUTPUT RECORD
40                   J        CLOOP             LOOP
45        ENDFIL     LDA      =C'EOF'           INSERT END OF FILE MARKER
50                   STA      BUFFER
55                   LDA      #3                SET LENGTH = 3
60                   STA      LENGTH
65                   +JSUB    WRREC             WRITE EOF
70                   J        @RETADR           RETURN TO CALLER
93                   LTORG
95        RETADR     RESW     1
100       LENGTH     RESW     1                 LENGTH OF RECORD
105       BUFFER     RESB     4096              4096-BYTE BUFFER AREA
106       BUFEND     EQU      *
107       MAXLEN     EQU      BUFEND-BUFFER     MAXIMUM RECORD LENGTH
110       .
115       .          SUBROUTINE TO READ RECORD INTO BUFFER
120       .
125       RDREC      CLEAR    X                 CLEAR LOOP COUNTER
130                  CLEAR    A                 CLEAR A TO ZERO
132                  CLEAR    S                 CLEAR S TO ZERO
133                  +LDT     #MAXLEN
135       RLOOP      TD       INPUT             TEST INPUT DEVICE
140                  JEQ      RLOOP             LOOP UNTIL READY
145                  RD       INPUT             READ CHARACTER INTO REGISTER A
150                  COMPR    A,S               TEST FOR END OF RECORD (X'00')
```

圖2.9 展示更多組譯器特性的程式範例

```
155                 JEQ     EXIT            EXIT LOOP IF EOR
160                 STCH    BUFFER,X        STORE CHARACTER IN BUFFER
165                 TIXR    T               LOOP UNLESS MAX LENGTH
170                 JLT     RLOOP             HAS BEEN REACHED
175     EXIT        STX     LENGTH          SAVE RECORD LENGTH
180                 RSUB                    RETURN TO CALLER
185     INPUT       BYTE    X'F1'           CODE FOR INPUT DEVICE
195     .
200     .           SUBROUTINE TO WRITE RECORD FROM BUFFER
205     .
210     WRREC       CLEAR   X               CLEAR LOOP COUNTER
212                 LDT     LENGTH
215     WLOOP       TD      =X'05'          TEST OUTPUT DEVICE
220                 JEQ     WLOOP           LOOP UNTIL READY
225                 LDCH    BUFFER,X        GET CHARACTER FROM BUFFER
230                 WD      =X'05'          WRITE CHARACTER
235                 TIXR    T               LOOP UNTIL ALL CHARACTERS
240                 JLT     WLOOP             HAVE BEEN WRITTEN
245                 RSUB                    RETURN TO CALLER
255                 END     FIRST
```

組譯圖2.9程式所產生之具目的碼的程式

```
 5     0000   COPY     START   0
10     0000   FIRST    STL     RETADR          17202D
13     0003            LDB     #LENGTH         69202D
14                     BASE    LENGTH
15     0006   CLOOP    +JSUB   RDREC           4B101036
20     000A            LDA     LENGTH          032026
25     000D            COMP    #0              290000
30     0010            JEQ     ENDFIL          332007
35     0013            +JSUB   WRREC           4B10105D
40     0017            J       CLOOP           3F2FEC
45     001A   ENDFIL   LDA     =C'EOF'         032010
50     001D            STA     BUFFER          0F2016
55     0020            LDA     #3              010003
60     0023            STA     LENGTH          0F200D
65     0026            +JSUB   WRREC           4B10105D
70     002A            J       @RETADR         3E2003
93                     LTORG
       002D   *        =C'EOF'                 454F46
95     0030   RETADR   RESW    1
100    0033   LENGTH   RESW    1
105    0036   BUFFER   RESB    4096
106    1036   BUFEND   EQU     *
107    1000   MAXLEN   EQU     BUFEND-BUFFER
110                    .
115                    .       SUBROUTINE TO READ RECORD INTO BUFFER
120                    .
125    1036   RDREC    CLEAR   X               B410
130    1038            CLEAR   A               B400
132    103A            CLEAR   S               B440
133    103C            +LDT    #MAXLEN         75101000
135    1040   RLOOP    TD      INPUT           E32019
140    1043            JEQ     RLOOP           332FFA
145    1046            RD      INPUT           DB2013
150    1049            COMPR   A,S             A004
155    104B            JEQ     EXIT            332008
160    104E            STCH    BUFFER,X        57C003
165    1051            TIXR    T               B850
170    1053            JLT     RLOOP           3B2FEA
175    1056   EXIT     STX     LENGTH          134000
180    1059            RSUB                    4F0000
185    105C   INPUT    BYTE    X'F1'           F1
195                    .
200                    .       SUBROUTINE TO WRITE RECORD FROM BUFFER
205                    .
210    105D   WRREC    CLEAR   X               B410
212    105F            LDT     LENGTH          774000
215    1062   WLOOP    TD      =X'05'          E32011
220    1065            JEQ     WLOOP           332FFA
225    1068            LDCH    BUFFER,X        53C003
230    106B            WD      =X'05'          DF2008
```

2.3.1 常值

BASE　*

LDB　=*

- 位於程式的第一行時，會將程式的起始位址載入到B暫存器中，這個值便可以供「基底相對定址」之用。
- 常值表（LITTAB）:針對每個常值，此表格包含常值名稱、運算元值及長度，以及運算元位於常值區的位址。
- 第一階段辨識每個常值運算元
- 第二階段時可以在LITTAB表中搜尋每一個遇到的常值運算元，來產生目的碼所需的運算元位址

2.3.2 符號定義敘述

- 符號　EQU　值
- 此敘述定義一個符號（將置入SYMTAB），而且分配一個特定的資料值
- +LDT　#4096

- MAXLEN　EQU　4096
- +LDT　#MAXLEN

2.3.2 符號定義敘述

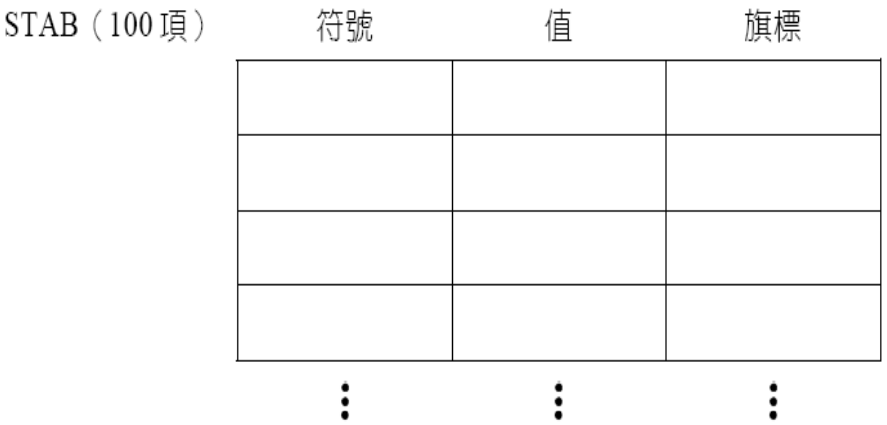
- A EQU 0
- X EQU 1
- L EQU 2
- 接受「RMO A, X」

2.3.2 符號定義敘述

- ORG　　值
- 其中的值可以是常數或含有常數及既已定義符號的運算式，當組譯器遇到此敘述時，會將「位址計數器」（LOCCTR）的內容重置為該指定值。

2.3.2 符號定義敘述

- STAB RESB 1100
- SYMBOL EQU STAB
- VALUE EQU STAB+6
- FLAGS EQU STAB+9



- LDA VALUE,X

- STAB RESB 1100
- ORG STAB
- SYMBOL RESB 6
- VALUE RESW 1
- FLAGS RESB 2
- ORG STAB+1100

2.3.2 符號定義敘述

- 下列的指令順序是允許的

```
ALPHA    RESW    1
BETA          EQU     ALPHA
```

- 下列的指令順序是不允許的

```
BETA          EQU     ALPHA
ALPHA    RESW     1
```

2.3.2 符號定義敘述

- 無法處理:向前引用

```
ALPHA     EQU     BETA
BETA      EQU     DELTA
DELTA     RESW    1
```

2.3.3 運算式

- 允許由 + - * 和 / 等運算子所形成之算術運算式

```
106   BUFEND      EQU          *
107   MAXLEN   EQU          BUFEND-BUFFER
```

- BUFEND + BUFFER、100 – BUFFER或 3 * BUFFER: 錯誤

符號	型別	值
RETADR	R	0030
BUFFER	R	0036
BUFEND	R	1036
MAXEEN	A	1000

2.3.4 程式區塊

- 使用「程式區塊」（program blocks）來表示可以在個別目的程式單元中重新安排的區段碼，而使用「控制段」（control sections）表示可以組譯成獨立目的程式單元的區段
- 每個程式區塊（program block）實際上可以包含原始程式中的數個獨立區段（segment）。組譯器（邏輯上）會重新安排這些區段，把每個區塊的片段都集合起來，然後目的程式的這些區塊將依其於原始程式中的順序，來指定位址。

```
 5    COPY       START    0                 COPY FILE FROM INPUT TO OUTPUT
10    FIRST      STL      RETADR            SAVE RETURN ADDRESS
15    CLOOP      JSUB     RDREC             READ INPUT RECORD
20               LDA      LENGTH            TEST FOR EOF (LENGTH = 0)
25               COMP     #0
30               JEQ      ENDFIL            EXIT IF EOF FOUND
35               JSUB     WRREC             WRITE OUTPUT RECORD
40               J        CLOOP             LOOP
45    ENDFIL     LDA      =C'EOF'           INSERT END OF FILE MARKER
50               STA      BUFFER
55               LDA      #3                SET LENGTH = 3
60               STA      LENGTH
65               JSUB     WRREC             WRITE EOF
70               J        @RETADR           RETURN TO CALLER
92               USE      CDATA
95    RETADR     RESW     1
100   LENGTH     RESW     1                 LENGTH OF RECORD
103              USE      CBLKS
105   BUFFER     RESB     4096              4096-BYTE BUFFER AREA
106   BUFEND     EQU      *                 FIRST LOCATION AFTER BUFFER
107   MAXLEN     EQU      BUFEND-BUFFER     MAXIMUM RECORD LENGTH
110   .
115   .          SUBROUTINE TO READ RECORD INTO BUFFER
120   .
123              USE
125   RDREC      CLEAR    X                 CLEAR LOOP COUNTER
130              CLEAR    A                 CLEAR A TO ZERO
132              CLEAR    S                 CLEAR S TO ZERO
133              +LDT     #MAXLEN
135   RLOOP      TD       INPUT             TEST INPUT DEVICE
140              JEQ      RLOOP             LOOP UNTIL READY
145              RD       INPUT             READ CHARACTER INTO REGISTER A
150              COMPR    A,S               TEST FOR END OF RECORD (X'00')
155              JEQ      EXIT              EXIT LOOP IF EOR
160              STCH     BUFFER,X          STORE CHARACTER IN BUFFER
```

```
165                 TIXR      T               LOOP UNLESS MAX LENGTH
170                 JLT       RLOOP             HAS BEEN REACHED
175     EXIT        STX       LENGTH          SAVE RECORD LENGTH
180                 RSUB                      RETURN TO CALLER
183                 USE       CDATA
185     INPUT       BYTE      X'F1'           CODE FOR INPUT DEVICE
195     .
200     .           SUBROUTINE TO WRITE RECORD FROM BUFFER
205     .
208                 USE
210     WRREC       CLEAR     X               CLEAR LOOP COUNTER
212                 LDT       LENGTH
215     WLOOP       TD        =X'05'          TEST OUTPUT DEVICE
220                 JEQ       WLOOP           LOOP UNTIL READY
225                 LDCH      BUFFER,X        GET CHARACTER FROM BUFFER
230                 WD        =X'05'          WRITE CHARACTER
235                 TIXR      T               LOOP UNTIL ALL CHARACTERS
240                 JLT       WLOOP             HAVE BEEN WRITTEN
245                 RSUB                      RETURN TO CALLER
252                 USE       CDATA
253                 LTORG
255                 END       FIRST
```

程式區塊

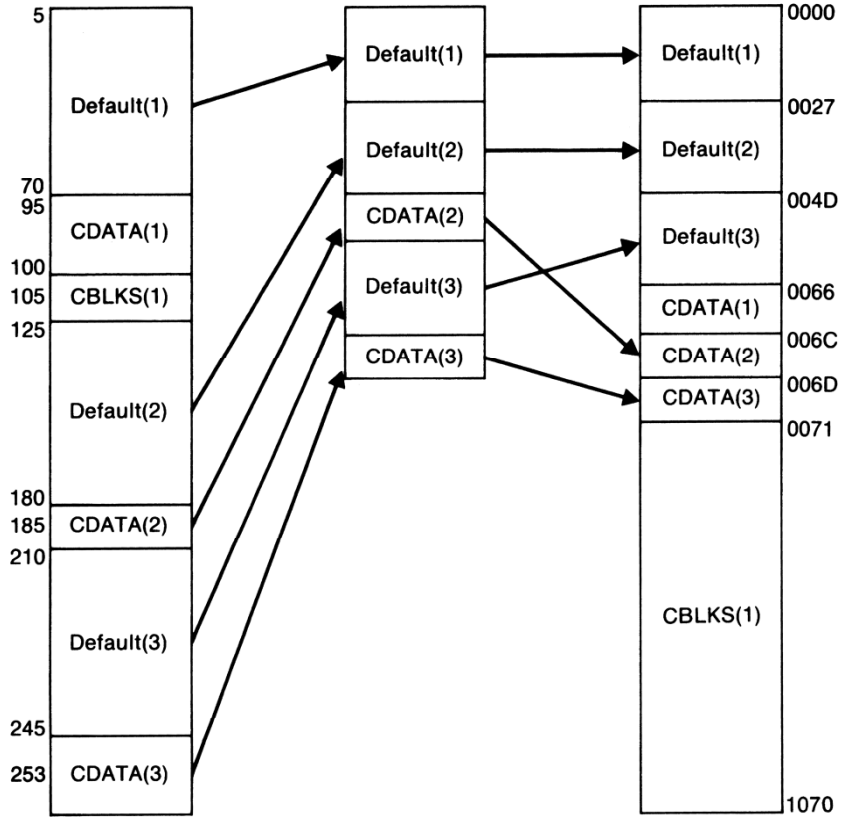
區塊名稱	區塊編號	位址	長度
（預設）	0	0000	0066
CDATA	1	0066	000B
CBEKS	2	0071	1000

```
 5      0000  0     COPY      START   0
10      0000  0     FIRST     STL     RETADR          172063
15      0003  0     CLOOP     JSUB    RDREC           4B2021
20      0006  0               LDA     LENGTH          032060
25      0009  0               COMP    #0              290000
30      000C  0               JEQ     ENDFIL          332006
35      000F  0               JSUB    WRREC           4B203B
40      0012  0               J       CLOOP           3F2FEE
45      0015  0     ENDFIL    LDA     =C'EOF'         032055
50      0018  0               STA     BUFFER          0F2056
55      001B  0               LDA     #3              010003
60      001E  0               STA     LENGTH          0F2048
65      0021  0               JSUB    WRREC           4B2029
70      0024  0               J       @RETADR         3E203F
92      0000  1               USE     CDATA
95      0000  1     RETADR    RESW    1
100     0003  1     LENGTH    RESW    1
103     0000  2               USE     CBLKS
105     0000  2     BUFFER    RESB    4096
106     1000  2     BUFEND    EQU     *
107     1000        MAXLEN    EQU     BUFEND-BUFFER
110                 .
115                 .         SUBROUTINE TO READ RECORD INTO BUFFER
120                 .
123     0027  0               USE
125     0027  0     RDREC     CLEAR   X               B410
130     0029  0               CLEAR   A               B400
132     002B  0               CLEAR   S               B440
133     002D  0               +LDT    #MAXLEN         75101000
135     0031  0     RLOOP     TD      INPUT           E32038
140     0034  0               JEQ     RLOOP           332FFA
145     0037  0               RD      INPUT           DB2032
150     003A  0               COMPR   A,S             A004
155     003C  0               JEQ     EXIT            332008
160     003F  0               STCH    BUFFER,X        57A02F
165     0042  0               TIXR    T               B850
170     0044  0               JLT     RLOOP           3B2FEA
175     0047  0     EXIT      STX     LENGTH          13201F
180     004A  0               RSUB                    4F0000
183     0006  1               USE     CDATA
185     0006  1     INPUT     BYTE    X'F1'           F1
195                 .
200                 .         SUBROUTINE TO WRITE RECORD FROM BUFFER
205                 .
208     004D  0               USE
210     004D  0     WRREC     CLEAR   X               B410
212     004F  0               LDT     LENGTH          772017
215     0052  0     WLOOP     TD      =X'05'          E3201B
220     0055  0               JEQ     WLOOP           332FFA
225     0058  0               LDCH    BUFFER,X        53A016
230     005B  0               WD      =X'05'          DF2012
235     005E  0               TIXR    T               B850
240     0060  0               JLT     WLOOP           3B2FEF
245     0063  0               RSUB                    4F0000
252     0007  1               USE     CDATA
253                           LTORG
        0007  1     *         =C'EOF                  454F46
        000A  1     *         =X'05'                  05
255                           END     FIRST
```

圖2.13　對應圖2.11的目的程式

```
HCOPY  000000001071
T00000001E1720634B2021032060290000332006 4B203B3F2FEE0320550F2056010003
T00001E090F20484B20293E203F
T0000271DB410B400B44075101000E32038332FFADB2032A00433200857A02FB850
T000044093B2FEA13201F4F0000
T00006C01F1
T00004D19B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000
T00006D04454F4605
E000000
```

圖2.14　依組譯和載入程序追蹤圖2.11的程式區塊



2.3.5 控制段與程式連結

- 控制段（control section）是程式的一部份，並且在經過組譯後仍然保有原來的本質；各個控制段皆可獨自載入和重定址
- 不同的控制段通常是扮演副程式，或程式內的邏輯單元
- 程式設計師可以各別組譯、載入和處理各個控制段
- 控制段之間的引用，稱之為「外部引用」

```
  5    COPY      START     0               COPY FILE FROM INPUT TO OUTPUT
  6              EXTDEF    BUFFER,BUFEND,LENGTH
  7              EXTREF    RDREC,WRREC
 10    FIRST     STL       RETADR          SAVE RETURN ADDRESS
 15    CLOOP    +JSUB      RDREC           READ INPUT RECORD
 20              LDA       LENGTH          TEST FOR EOF (LENGTH = 0)
 25              COMP      #0
 30              JEQ       ENDFIL          EXIT IF EOF FOUND
 35             +JSUB      WRREC           WRITE OUTPUT RECORD
 40              J         CLOOP           LOOP
 45    ENDFIL    LDA       =C'EOF'         INSERT END OF FILE MARKER
 50              STA       BUFFER
 55              LDA       #3              SET LENGTH = 3
 60              STA       LENGTH
 65             +JSUB      WRREC           WRITE EOF
 70              J         @RETADR         RETURN TO CALLER
 95    RETADR    RESW      1
100    LENGTH    RESW      1               LENGTH OF RECORD
103              LTORG
105    BUFFER    RESB      4096            4096-BYTE BUFFER AREA
106    BUFEND    EQU       *
107    MAXLEN    EQU       BUFEND-BUFFER

109    RDREC     CSECT
110    .
115    .         SUBROUTINE TO READ RECORD INTO BUFFER
120    .
122              EXTREF    BUFFER,LENGTH,BUFEND
125              CLEAR     X               CLEAR LOOP COUNTER
130              CLEAR     A               CLEAR A TO ZERO
132              CLEAR     S               CLEAR S TO ZERO
133              LDT       MAXLEN
135    RLOOP     TD        INPUT           TEST INPUT DEVICE
140              JEQ       RLOOP           LOOP UNTIL READY
145              RD        INPUT           READ CHARACTER INTO REGISTER A
150              COMPR     A,S             TEST FOR END OF RECORD (X'00')
155              JEQ       EXIT            EXIT LOOP IF EOR
160             +STCH      BUFFER,X        STORE CHARACTER IN BUFFER
165              TIXR      T               LOOP UNLESS MAX LENGTH
170              JLT       RLOOP             HAS BEEN REACHED
175    EXIT      +STX      LENGTH          SAVE RECORD LENGTH
180              RSUB                      RETURN TO CALLER
185    INPUT     BYTE      X'F1'           CODE FOR INPUT DEVICE
190    MAXLEN    WORD      BUFEND-BUFFER

193    WRREC     CSECT
195    .
200    .         SUBROUTINE TO WRITE RECORD FROM BUFFER
205    .
207              EXTREF    LENGTH,BUFFER

210              CLEAR     X               CLEAR LOOP COUNTER
212             +LDT       LENGTH
215    WLOOP     TD        =X'05'          TEST OUTPUT DEVICE
220              JEQ       WLOOP           LOOP UNTIL READY
225             +LDCH      BUFFER,X        GET CHARACTER FROM BUFFER
230              WD        =X'05'          WRITE CHARACTER
235              TIXR      T               LOOP UNTIL ALL CHARACTERS
240              JLT       WLOOP             HAVE BEEN WRITTEN
245              RSUB                      RETURN TO CALLER
255              END       FIRST
```

圖2.15　控制區段和程式連結的說明

```
 5      0000     COPY      START   0
 6                         EXTDEF  BUFFER,BUFEND,LENGTH
 7                         EXTREF  RDREC,WRREC
10      0000     FIRST     STL     RETADR          172027
15      0003     CLOOP    +JSUB    RDREC           4B100000
20      0007              LDA     LENGTH          032023
25      000A              COMP    #0              290000
30      000D              JEQ     ENDFIL          332007
35      0010             +JSUB    WRREC           4B100000
40      0014              J       CLOOP           3F2FEC
45      0017     ENDFIL    LDA    =C'EOF'          032016
50      001A              STA     BUFFER          0F2016
55      001D              LDA     #3              010003
60      0020              STA     LENGTH          0F200A
65      0023             +JSUB    WRREC           4B100000
70      0027              J       @RETADR         3E2000
95      002A     RETADR    RESW    1
```

```
100     002D     LENGTH    RESW    1
103                        LTORG
        0030     *        =C'EOF'                  454F46
105     0033     BUFFER    RESB    4096
106     1033     BUFEND    EQU     *
107     1000     MAXLEN    EQU     BUFEND-BUFFER

109     0000     RDREC     CSECT
110              .
115              .         SUBROUTINE TO READ RECORD INTO BUFFER
120              .
122                        EXTREF  BUFFER,LENGTH,BUFEND
125     0000               CLEAR   X               B410
130     0002               CLEAR   A               B400
132     0004               CLEAR   S               B440
133     0006               LDT     MAXLEN          77201F
135     0009     RLOOP     TD      INPUT           E3201B
140     000C               JEQ     RLOOP           332FFA
145     000F               RD      INPUT           DB2015
150     0012               COMPR   A,S             A004
155     0014               JEQ     EXIT            332009
160     0017              +STCH    BUFFER,X        57900000
165     001B               TIXR    T               B850
170     001D               JLT     RLOOP           3B2FE9
175     0020     EXIT     +STX     LENGTH          13100000
180     0024               RSUB                    4F0000
185     0027     INPUT     BYTE    X'F1'           F1
190     0028     MAXLEN    WORD    BUFEND-BUFFER   000000

193     0000     WRREC     CSECT
195              .
200              .         SUBROUTINE TO WRITE RECORD FROM BUFFER
205              .
207                        EXTREF  LENGTH,BUFFER
210     0000               CLEAR   X               B410
212     0002              +LDT     LENGTH          77100000
215     0006     WLOOP     TD      =X'05'          E32012
220     0009               JEQ     WLOOP           332FFA
225     000C              +LDCH    BUFFER,X        53900000
230     0010               WD      =X'05'          DF2008
235     0013               TIXR    T               B850
240     0015               JLT     WLOOP           3B2FEE
245     0018               RSUB                    4F0000
255                        END     FIRST
        001B     *        =X'05'                   05
```

圖2.16　組譯圖2.15程式所產生之具目的碼的程式

定義記錄

欄 1	D
欄 2－7	本控制段中所定義之外部符號的名稱
欄 8－13	本控制段之符號的相對位址（16進位值）
欄 14－73	重覆2－13列的資訊，供其他外部符號所用

引用記錄

欄 1	R
欄 2－7	本控制段所引用之外部符號的名稱
欄 8－73	其他外部引用符號的名稱

修正記錄（已修改過）

欄 1	M
欄 2－7	被修改欄位的起始位址，相對於控制段的開頭（16進位值）
欄 8－9	被修改欄位的長度，單位為半位元組（16進位值）
欄 10	修改旗標（+ 或 - ）
欄 11－16	外部符號的資料值應要加到指定的欄位，或是指定的欄位減掉外部符號的資料值

對應圖2.15 的目的程式

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T000000 1D 1720274B100000032023290000332007 4B1000003F2FEC0320160F2016
T00001D0D010003 0F200A4B1000003E2000
T0000300 3454F46
M00000405+RDREC
M00001105+WRREC
M00002405+WRREC
E000000


HRDREC 000000000002B
RBUFFERLENGTHBUFEND
T000000 1DB410B400B440 77201FE3201B332FFADB2015A00433200957900000B850
T00001D0E3B2FE9131000004F0000F1000000
M00001805+BUFFER
M00002105+LENGTH
M00002806+BUFEND
M00002806-BUFFER
E


HWRREC 000000000001C
RLENGTHBUFFER
T000000 1CB41077100000E32012332FFA53900000DF2008B8503B2FEE4F000005
M00000305+LENGTH
M00000D05+BUFFER
E
```

修正記錄

M00000705+COPY

M00001405+COPY

M00002705+COPY

2.4 組譯器設計選項

- 2.4.1 節描述單階段組譯器的結構和邏輯
- 2.4.2節則介紹多階段組譯器的概念

2.4.1 單階段組譯器

- 主要嘗試使用單階段組譯，並考量向前引用。
- 要求程式在引用資料項之前，必定在原始程式中先行定義。
  - 程式的邏輯經常需要向前跳躍
- 二種主要的類型
  - 一種是直接在記憶體裡產生目的碼
  - 另一種類型是產生通常的目的碼程式

```
 0     1000    COPY     START   1000
 1     1000    EOF      BYTE    C'EOF'          454F46
 2     1003    THREE    WORD    3               000003
 3     1006    ZERO     WORD    0               000000
 4     1009    RETADR   RESW    1
 5     100C    LENGTH   RESW    1
 6     100F    BUFFER   RESB    4096
 9             .
10     200F    FIRST    STL     RETADR          141009
15     2012    CLOOP    JSUB    RDREC           48203D
20     2015             LDA     LENGTH          00100C
25     2018             COMP    ZERO            281006
30     201B             JEQ     ENDFIL          302024
35     201E             JSUB    WRREC           482062
40     2021             J       CLOOP           302012
45     2024    ENDFIL   LDA     EOF             001000
50     2027             STA     BUFFER          0C100F
55     202A             LDA     THREE           001003
60     202D             STA     LENGTH          0C100C
65     2030             JSUB    WRREC           482062
70     2033             LDL     RETADR          081009
75     2036             RSUB                    4C0000
110            .
115            .        SUBROUTINE TO READ RECORD INTO BUFFER
120            .
121    2039    INPUT    BYTE    X'F1'           F1
122    203A    MAXLEN   WORD    4096            001000
124            .
125    203D    RDREC    LDX     ZERO            041006
130    2040             LDA     ZERO            001006
135    2043    RLOOP    TD      INPUT           E02039
140    2046             JEQ     RLOOP           302043
145    2049             RD      INPUT           D82039
150    204C             COMP    ZERO            281006
155    204F             JEQ     EXIT            30205B
160    2052             STCH    BUFFER,X        54900F
165    2055             TIX     MAXLEN          2C203A
170    2058             JLT     RLOOP           382043
175    205B    EXIT     STX     LENGTH          10100C
180    205E             RSUB                    4C0000
195            .
200            .        SUBROUTINE TO WRITE RECORD FROM BUFFER
205            .
206    2061    OUTPUT   BYTE    X'05'           05
207            .
210    2062    WRREC    LDX     ZERO            041006
215    2065    WLOOP    TD      OUTPUT          E02061
220    2068             JEQ     WLOOP           302065
225    206B             LDCH    BUFFER,X        50900F
230    206E             WD      OUTPUT          DC2061
235    2071             TIX     LENGTH          2C100C
240    2074             JLT     WLOOP           382065
245    2077             RSUB                    4C0000
255                     END     FIRST
```

圖2.18
單階段組譯器的範例程式

載入即執行（load-and-go）

- 首先所討論的單階段組譯器，可以在記憶體產生目的碼，以便立即執行。既不會產生目的碼程式，也不需要載入器。
- 當組譯器掃瞄原始程式時，只產生目的碼指令，如果一個指令的運算元是一個尚未定義的符號，該指令被組譯時，將先忽略此運算元的位址，但是會將此運算元的符號置入符號表內
- 當遇到此符號的定義時，將會掃瞄該符號的向前引用串列（如果存在時），而且將正確的位址置入先前的目的碼指令中。

圖2.19(a)　掃瞄圖2.18的第40行之後，記憶體內的目的碼和符號表項目的狀況

```
1000    454F4600  00030000  00xxxxxx  xxxxxxxx
1010    xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx
  •
  •
  •
2000    xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxx14
2010    100948--  --00100C  28100630  -----48--
2020    --3C2012
  •
  •
  •
```

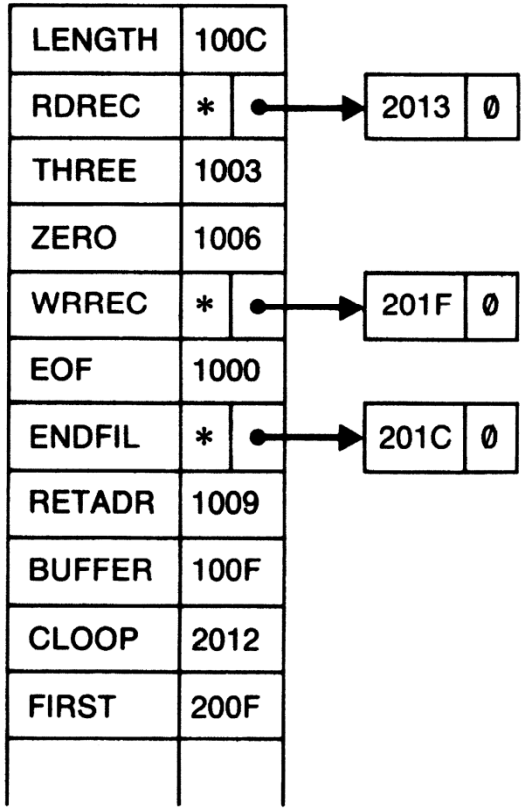


圖2.19(b)　掃瞄圖2.18的第160行之後，記憶體內的目的碼和符號表項目的狀況

```
1000    454F4600  00030000  00xxxxxx  xxxxxxxx
1010    xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx
  •
  •
  •
2000    xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxx14
2010    10094820  3D00100C  28100630  202448--
2020    --3C2012  0010000C  100F0010  030C100C
2030    48----08  10094C00  00F10010  00041006
2040    001006E0  20393020  43D82039  28100630
2050    ----5490  0F
  •
  •
  •
```

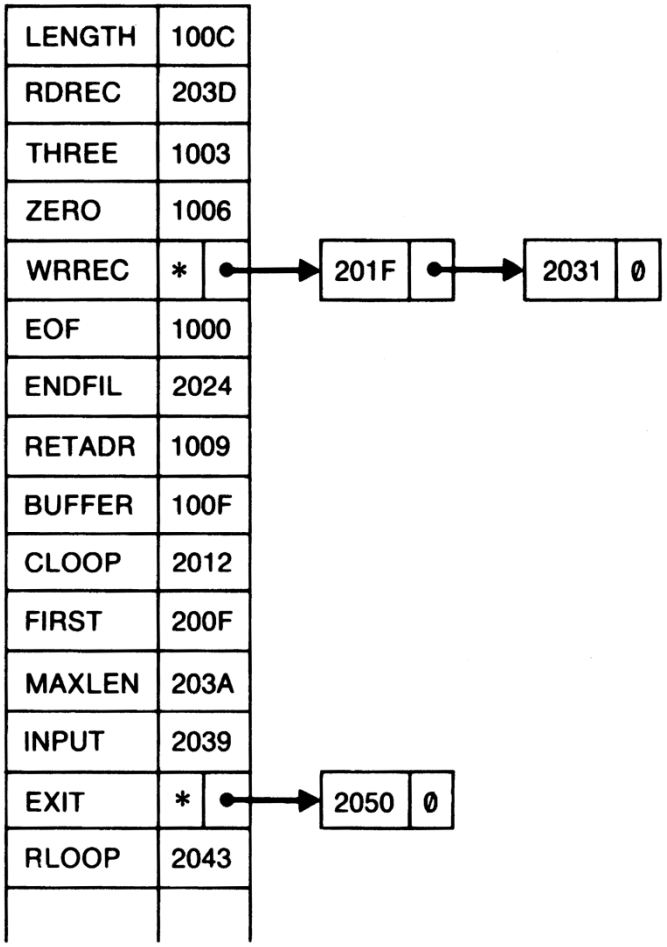


圖2.20　顯示圖2.18之程式經單階段組譯器處理後的目的程式

```
HCOPY  001000001 07A
T001000094 54F460000 03000000
T00200F151410094800000 0100C2810063000004800003C2012
T00201C022024
T0020241900100000C100F0010030C100C480000081009 4C0000F1001000
T00201302203D
T00203D1E041006001006E02039302043D82039281006300000549 00F2C203A382043
T00205002205B
T00205B0710100C4C000005
T00201F022062
T00203102 2062
T00206218041006E02061302065509 00FDC20612C100C3820654C0000
E00200F
```

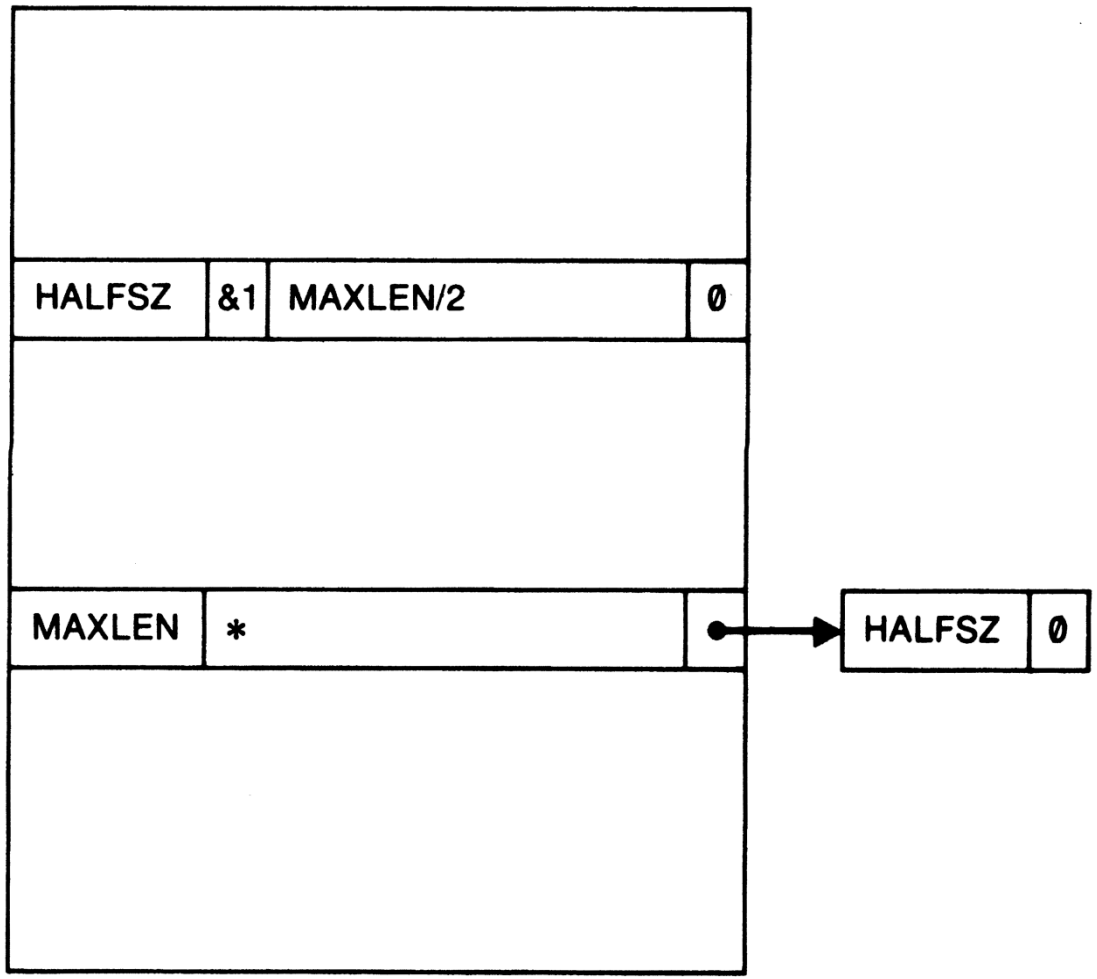
2.4.2 多階段組譯器

- 在EQU組譯器指引的討論中，任何在EQU右邊的符號都必須在原始程式中事前有所定義。
- 對ORG來說，也有類似的需求。
- 多階段組譯器
  - 第一階段時，將程式中涉及符號定義之向前引用的部份儲存起來，而後再依所儲存的定義進行額外處理，接著才執行正常的第二階段處理。

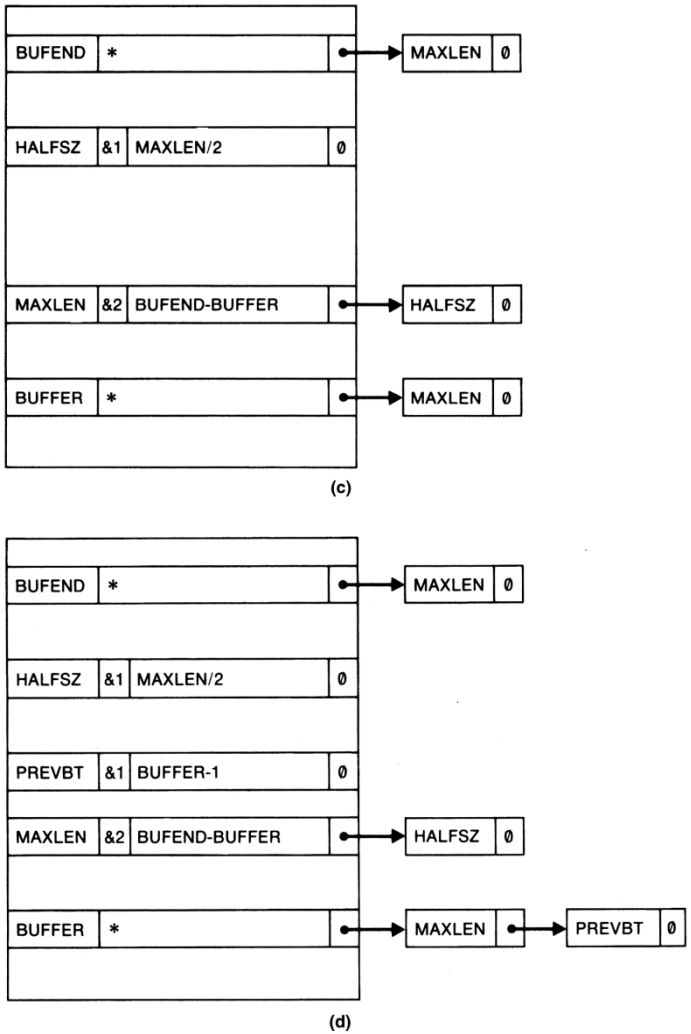
多階段組譯器的運作範例

```
1       HALFSZ      EQU     MAXLEN/2
2       MAXLEN      EQU     BUFEND-BUFFER
3       PREVBT      EQU     BUFFER-1
                            .
                            .
                            .
4       BUFFER      RESB    4096
5       BUFEND      EQU     *
```

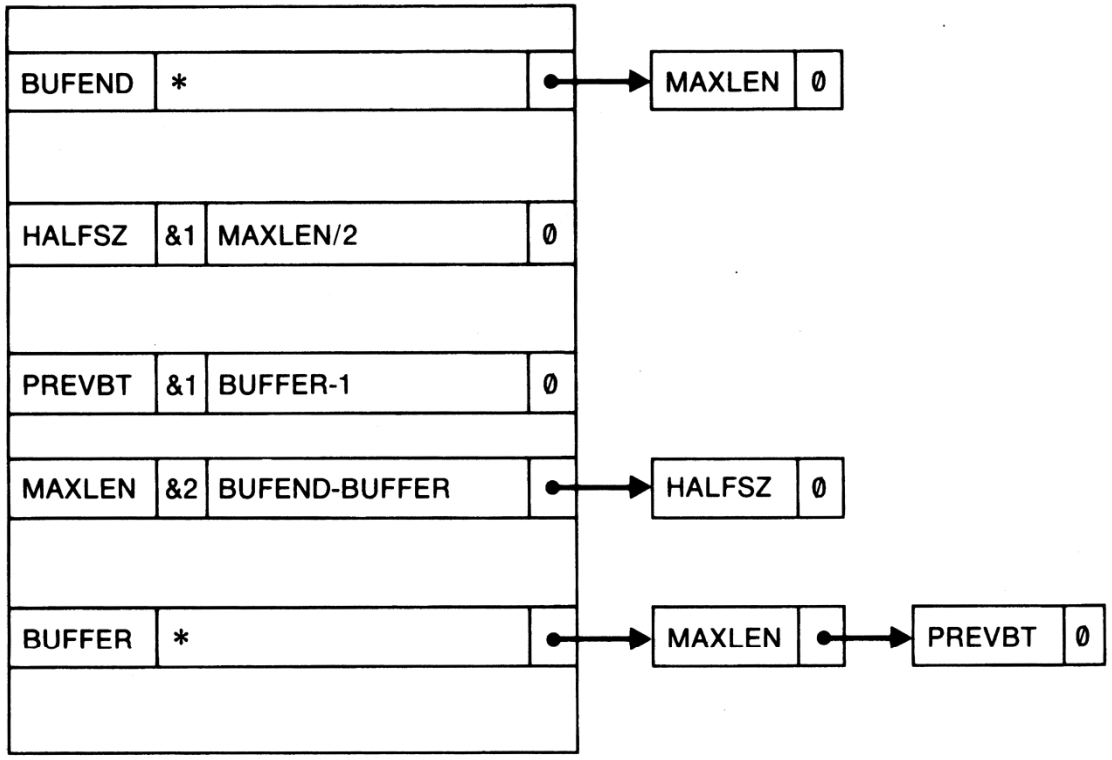
多階段組譯器的運作範例



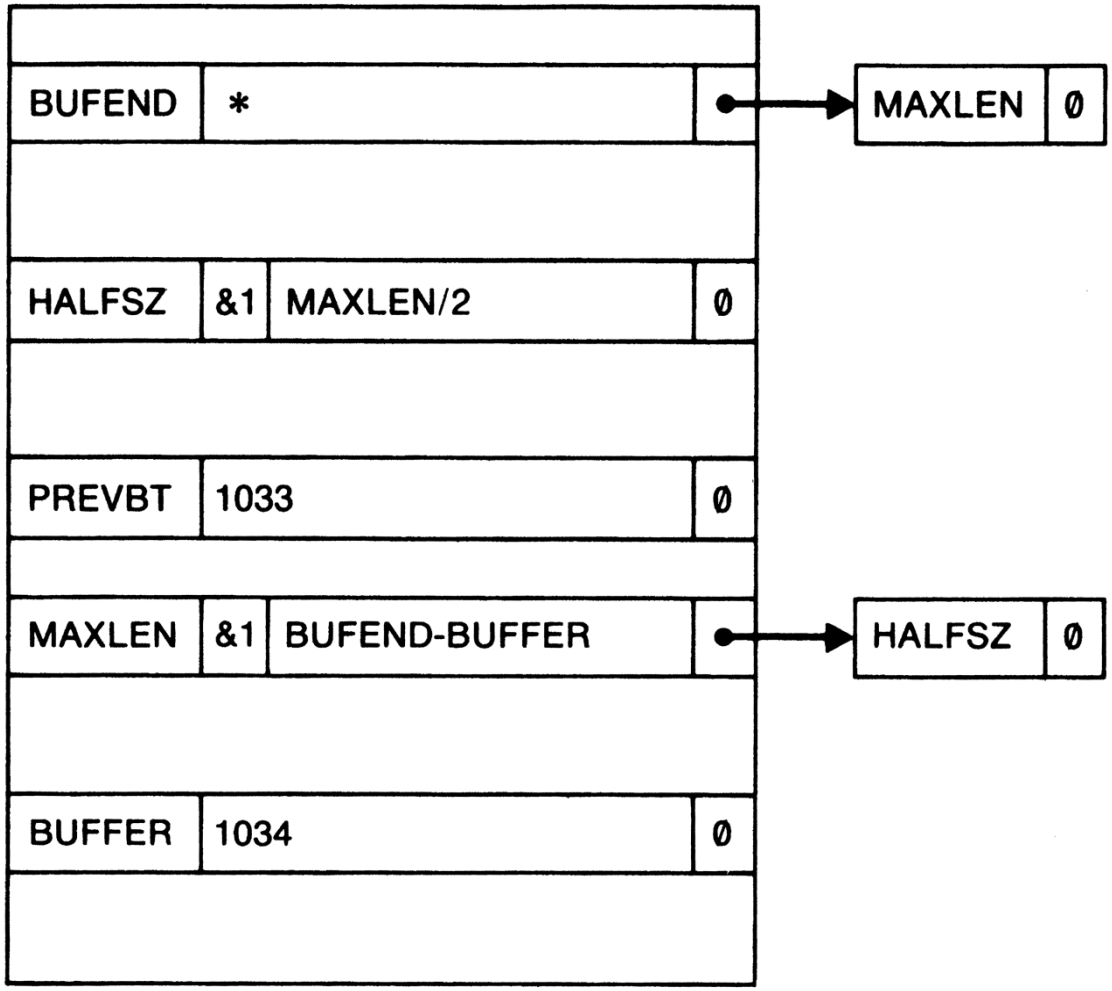
多階段組譯器的運作範例



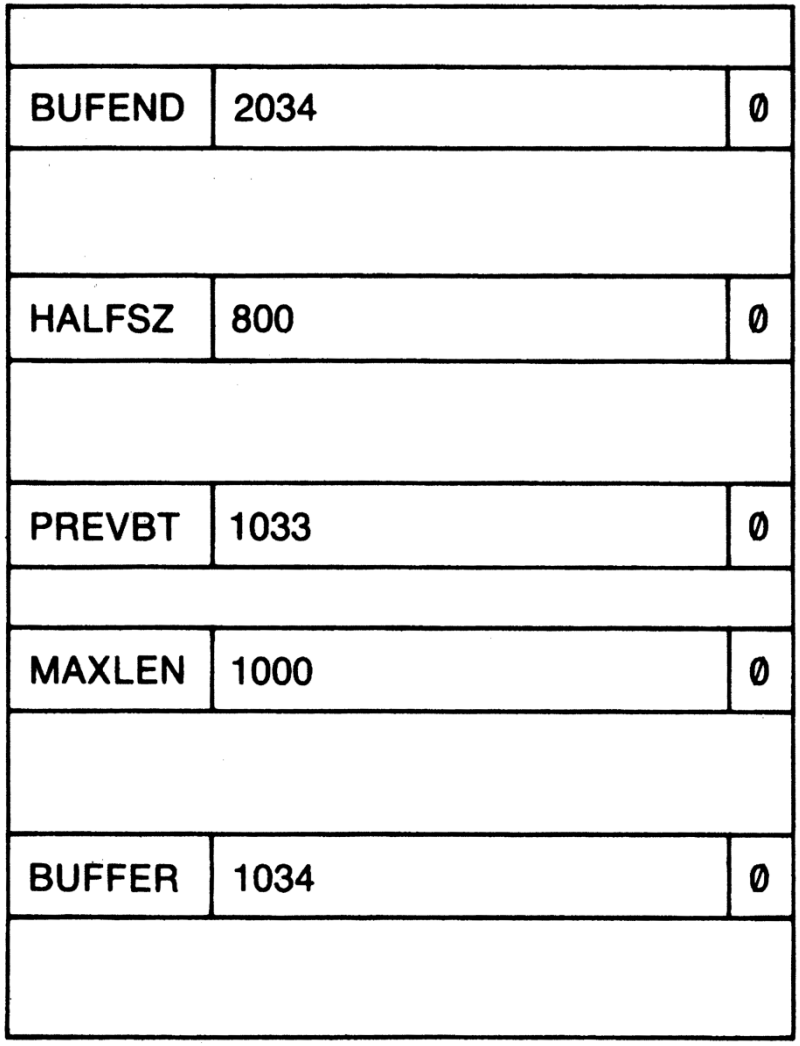
多階段組譯器的運作範例



多階段組譯器的運作範例



多階段組譯器的運作範例



2.5 實作範例

- Pentium(x86)
- SPARC
- PowerPC

2.5.1 MASM組譯器

- x86系統的程式設計師是將記憶體視為一些區段（segment）的集合。
  - CODE、DATA、CONST和STACK
  - 程式碼區段是用CS暫存器來定址，堆疊區段是用SS暫存器來定址。
  - 資料區段（包含常數區段）通常是使用DS、ES、FS或GS暫存器來定址。

2.5.1 MASM組譯器

ASSUME　ES:DATASEG2

- 告訴組譯器採用ES暫存器來指向DATASEG2區段的位址

```
MOV         AX,DATASEG2
MOV         ES,AX
```

- 將ES暫存器設定成指向DATASEG2資料區段

2.5.1 MASM組譯器

- 「近跳躍」
  - JMP　SHORT TARGET
- 「遠跳躍」
  - JMP　　FAR PTR TARGET
- 同一區段間的引用（References），將由組譯器自動處理。分開模組間的外部引用，則由連結器來處理。

2.5.2 SPARC組譯器

- 一個SPARC組合程式是分開成若干個單元，稱之為段落（sections）
  - .TEXT　　可執行的指令
  - .DATA　　初始讀／寫資料
  - .RODATA　　唯讀資料
  - .BSS　　未初始資料區
- 不同段落之間的引用是由連結器，而非組譯器來解決。

2.5.2 SPARC組譯器

- SPARC組譯器所產生的目的檔，包含轉譯的程式區段版本，以一系列需要執行的重定址和連結動作。
- 延遲分歧（delay branch）

```
CMP        %LO,   10
BLE        LOOP
ADD        %L2,   %L3,   %L4
```

2.5.2 SPARC組譯器

```
LOOP:       .
            .
            .
            ADD    %L2,  %L3,   %L4
            CMP    %L0,  10
            BLE    LOOP
            NOP
```

2.5.2 SPARC組譯器

```
LOOP:    ADD   %L2, %L3,  %L4
         .
         .
         CMP   %L0, 10
         BLE   LOOP
         NOP
```

2.5.2 SPARC組譯器

```
LOOP:       .
            .
            .
            CMP     %LO, 10
            BLE,A   LOOP
            ADD     %L2, %L3 , %L4
```

2.5.3 AIX組譯器

- 載入和儲存指令是使用一個基底暫存器和一個位移值來進行定址

  .USING　LENGTH,1

  .USING　BUFFER,4

- 會將GPR1 和GPR4設定為基底暫存器
- 使用一個基底暫存器表（base register table），以記載那些一般暫存器是用來做為基底暫存器使用，及其所含的基底位址

2.5.3 AIX組譯器

L　2,8(4)

- 明確指出運算元的位址是GPR4的內容加上8位元組的位址
- 一個AIX組合語言程式可以使用 .CSECT組譯指引，將其劃分為若干個控制區段（control sections）。
- 每一個控制區段都有個特定的儲存對映類別: PR（可執行的指令）、RO（唯讀資料）、RW（讀/寫資料）和BS（未初始化之讀/寫資料）。

2.5.3 AIX組譯器

- AIX組合語言提供了一種特殊型式的控制區段，稱之為「虛擬區段」，虛擬區段中的資料並不會真的變成目的程式的一部份，它們只供用於定義虛擬區段中的符號。
- AIX也提供一些共同區塊（common blocks），它們是一些未初始化的區塊空間，可供一些已經組譯的各別程式之間所共用。

2.5.3 AIX組譯器

- .GLOBL組譯指引可以將符號提供給連結器使用
- .EXTERN組譯指引則用於宣告一個符號是定義於其他的原始模組中。

2.5.3 AIX組譯器

- 已經組譯的控制區段會依據其儲存的對映類別，置放到目的程式之中。
- 可執行指令、唯讀資料和各式的除錯表都會被指定到 .TEXT的目的程式區段中
- 讀/寫資料和TOC項目會被指定到 .DATA的目的程式區段中
- 未初始化資料則會被指定到 .BSS區段。