HP 2100S Computer에 依한 Alpha-Computer의 Program Assembly를 爲한 Cross-Assembler의 開發

(A Cross-Assembler for Assembly of Programs for an Alpha-Computer on a HP 2100S Computer)

洪 玉 壽
(Hong, Ok Soo)

要 約

HP 2100 S computer 의 disc operating system 을 使用한 본 cross-assembler 는 alpha-minicomputer 의 assembly language program을 source 入力으로 하여 이 alpha-computer 에 依한 實行 (execution)을 目的으로 16進數 code 의 等價 object program 을 出力토록 設計되어 있다.

Abstract

A cross-assembler for use with a HP 2100S computer using its disc operating system is designed to accept an alpha-minicomputer's assembly language program as a source input and produces its equivalent object program in the form of hexadecimal for the alpha-computer.

1. 序 論

assembler 는 assembly language 로 쓰여진 program을 source 入力으로 이의 等價 object program의 發生을 目的으로한 language processor로서의 機能을 가진다. cross-assembler는 一種의 assembler로서 一般 assembler의 機能을 가지는 것은 마찬가지이나 source program 의 處理 (process)를 自體 computer에 依하여 遂行하는 것이 아니라 第3의 computer 에 依하여 處理하고 이의 結果, 즉 object program 을 다시 원래의 computer 에 依하여 實行할 수 있도록 한 language processer (translator)이다.

cross-assembler 는 minicomputer 의 programming 道具 (tool)로서 人氣가 높아가고 있으며 制限된 minicomputer의 最少 peripheral (周邊裝置) 또는 處理能力 (processing capability)과 programming에 所要되는 經費따위와 關聯하여 이의 重要性이 高調되고 있다.

본 cross-assembler는 一般 目的用 UNIVAC 1106 computer에 依하여 처음 實現되고, 다시 HP 2100 S computer의 disc operating system 을 使用토록 修正開發한 것이다. 이는 alpha-computer의 instruction set로 構成되는 任意의 alpha-source program을 入力으로 받아 16進數 (hexadecimal)形態의 等價 object program 으로 出力시킨다.

*正會員, 京畿工業專門大學 電子科
(Dept. of Electronice Engineering Gyeung-Gi Tecn. college, Korean)

接受日字: 1979年 5月 14日

2. Alpha-assembly Language

assembly language는 program writing을 直接으로 computre 內部 code와 machine address를 意識하지 않고서 할 수 있도록 設計된 言語이다. 이의 source program은 operation code (mnemonic machine operation code, OP code)와 pseudo code (assembly-directing pseudo code, assembly directive)및 symbolic addressing을 基本 要素로 한다. alpha-computer의 alpha-assembly language 는 177 opcode 와 10여개의 pseudo code (assembly directive)를 包含한다. 그림 1은 alpha-source program의 한 例를 보여 준다.

source statement는 symbolic address field (label), operation code field (opcode), operand field (operand), 그리고 選擇的으로 comment field (comment)를 包含한다. 이 program은 alpha-computer의 byte mode 動作의 한 例이다. 簡單히 設明을 첨가하면, 여기서 ORG, DATA, RES 및 END는 assembly directive 들이다. ORG (origin)은 program

control instruction으로 location counter의 初期 값을 設定한다. DATA (DATA definition)는 program에 data를 導入시켜 준다. RES (Reserve storage)는 memory space를 設定한다. END(End of assembly)는 assembly의 完了 (completion)를 알린다. 나머지 instruction은 모두 computer 動作을 決定짓는 object code로 번역 되어야 할 instruction이다. 첫째, SBM (Set Byte Mode)는byte mode 動作으로 set하라는 命令으로 풀이 된다. 둘째, LDAB (Load A Byte)는 symbolic address TBL을 참조하여 그곳 왼쪽 byte를 register A의 8-LSB의 위치에 load시키라는 命令으로 풀이 된다. 셋째, STAB (Store A Byte)는 LDAB의 逆過程으로 symbolic address RSLT를 참조하여 그곳 첫번째 왼쪽 byte위치에 register A의 8-LSB를 load 시키라는 instruction으로 풀이 된다. 마지막으로 HLT (Holt)는 processor control instruction으로 processing을 멈추라는 命令으로 풀이 된다.

```
    ****SOURCE LINES****
NO     LOCATION  LABEL OPCODE OPERAND        COMMENT
 1                *  PROGRAM FOR CHECKING BYTE MODE
 2                *  ADDRESSING.EXPECTED RESULTS
 3                *  MAY BE STORED AS THE SAME
 4                *  MANNER AS THOSE OF TBL IN RSLT
 5       42            ORG   :002A
 6       42            SBM
 7       43            LDAB  TBL
 8       44            STAB  RSLT
 9       45            LDAB  TBL+1
10       46            STAB  RSLT+1
11       47            LDAB  TBL+2
12       48            STAB  RSLT+2
13       49            LDAB  TBL+3
14       50            STAB  RSLT+3
15       51            HLT
16       52       TBL  DATA  :0102,:0304
17       54       RSLT RES   2,0
18                     END
```

그림 1. 알파어셈블리 프로그램(보기)

Fig. 1. An alpha-assembly language program.

3. Alpha-computer의 Instruction Format

이 computer는 program 記憶式 16-bit word format과 177個의 基本 instrution을 包含한다.instruction은 memory對 register와 register對register의 data移動, machine control, conditional jump,single/double-register shift,register change, 그리고 input/output instruction等의 7種으로 構成된다. computer가 利用하는 addressing mode는 direct scratchpad (base page), direct relative forward/backward, direct indexed,indirect scratchpad, indirect relative forward/backward,그리고 indirect post-indexed addressing等의 8種이고 word mode와 byte mode addressing이 可能하다. 여기서 instruction set과 addressing mode에 關하여 일일이 언급하는 것은 意味가 없으므로 어느 特定한 種類의 한 instruction에 對하여 이의 hardware實現에 關聯 言及하겠다. 그림 2는 alpha-computer의 single memory-reference instruction을 보여 주고 있다. op-code는 computer 動作을 決定하고 displacement는 operand의 所在에 關한 情報를 提供하며, tag bits는 그림 2와 3에 關聯하여 addressing mode를 결정하는데 기여하는 비트 (bit)로 displacement (operand part)로부터 實效 (effective) address를 決定하는데 기여된다. 各 field는 可能한 bit組合 規則에따라 32個의 opcode, 8個의 tag bits組合, 256個의 addressable location이 可能하다. 實際 이 computer는 displacement값에 順應하여 또한 tag bits 組合으로 決定되는 8個의 addressing mode를 利用하고 있다. 그

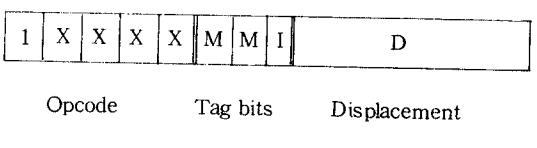

그림 2. 알파컴퓨터의 메모리참고 명령의 기계언어 코드형태

Fig. 2. The machine code format for the single memory reference instruction of the alpha-computer.

Addressing mode	M M I	Word mode	Byte mode
1.	0 0 0	Y= (D) Words: 00- : FF	Y=(D), Byte: 00- : FF
2.	0 1 0	Y= (D)+(P) + 1	Y=(D)+(P) +1, Byte 0
3.	1 0 0	Y= (D)+(X)	Y=(D)+(X)
4.	1 1 0	Y= (P)-(D)	Y=(D)+(P) +1, Byte 1
5.	0 0 1	AP=(D), AP=(AP), Y=(AP)	AP=(D), Y=(AP)
6.	0 1 1	AP=(D)+(P) +1, AP=(AP), Y=(AP)	AP=(D) + (P) + 1, Y=(AP)
7.	1 0 1	AP=(D), AP=(AP), Y=(AP)+(X)	AP=(D), Y=(AP) + (X)
8.	1 1 1	AP=(P)-(D), AP=(AP), Y=(AP)	AP=(P) - (D), Y=(AP)

* 여기서 ; : : Hexadecimal identifier
D: Displacement
P : Program lacation counter
X: Index register
AP: Address pointer
Y: Effective address

그림 3. 單메모리 참고 명령의 tag 비트에 따른 가능한 8-번지 모드
Fig. 3. The 8 possible addressing modes in accordance with tag bits for single-memory reference instructions.

ADD (ADDB) : 8800

Scratchpad (Base page)	Relative Forward	Relative Backward	Out-of-range (Multi-level)
• Direct mode			
8800 (8800)	8A00 (8A00)	8E00 (8E00)	8900 (8900) 0000 (0000)
• Indirect mode			
8900 (8900)	8B00 (8B00)	8F00 (8F00)	8900 (••••) 8000 (••••)
• Indexed mode			
8C00 (8C00)	•••• (••••)	•••• (••••)	8D00 (8D00) 0000 (0000)
• Indirect post-indexed mode			
8D00 (8D00)	•••• (••••)	•••• (••••)	8D00 (••••) 8000 (••••)

그림 4. ADD (ADDB) 명령의 수정가능성
Fig. 4. Modification of the ADD (ADDB) instruction depending on addressing mode and range.

림 3은 그림 2에 對한 addressing mode와 關聯 實效 address의 다양성을 보여 주며 그림 4는 그림 2에 대한 addressing mode와 範圍에 따른 한 例로 ADD (ADDB) instruction에 對한 instruction format의 확대 可能性과 융통성을 보여 준다.

4. Cross-assembler의 設計

4.1 總 括

본 cross-assembler는 alpha-source program을 assemble 하기 爲하여 HP 2100 S computer의 disc operating system을 使用하여 開發한 2-pass as-

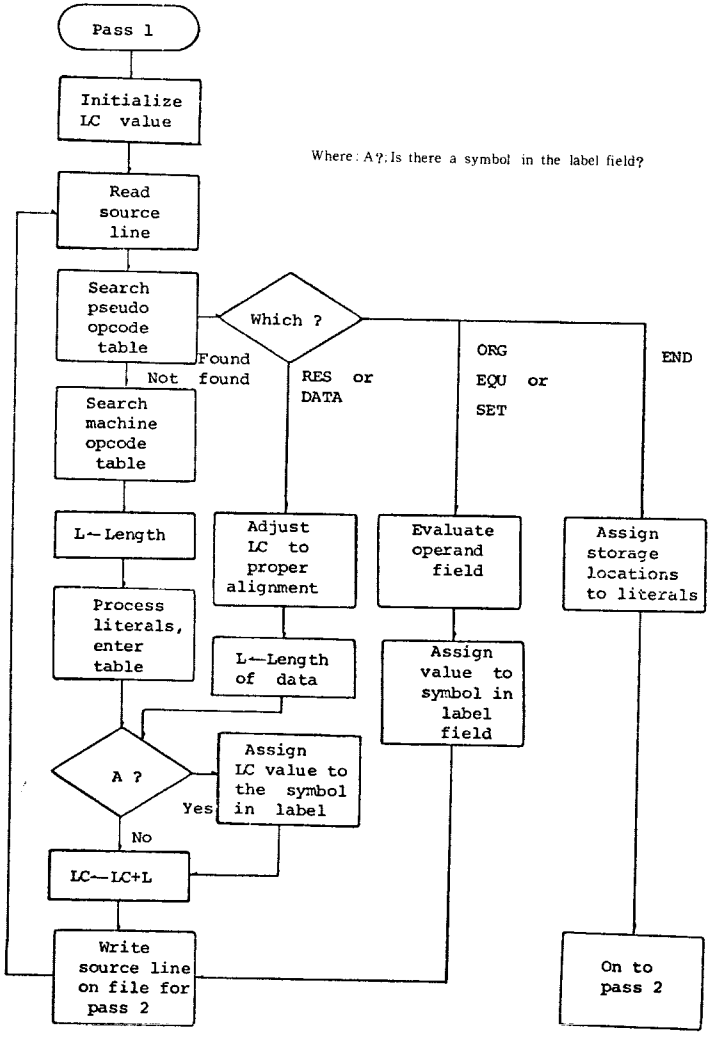

그림 5. 전형적인 패스 1의 흐름

Fig. 5. Typical flow of pass 1.

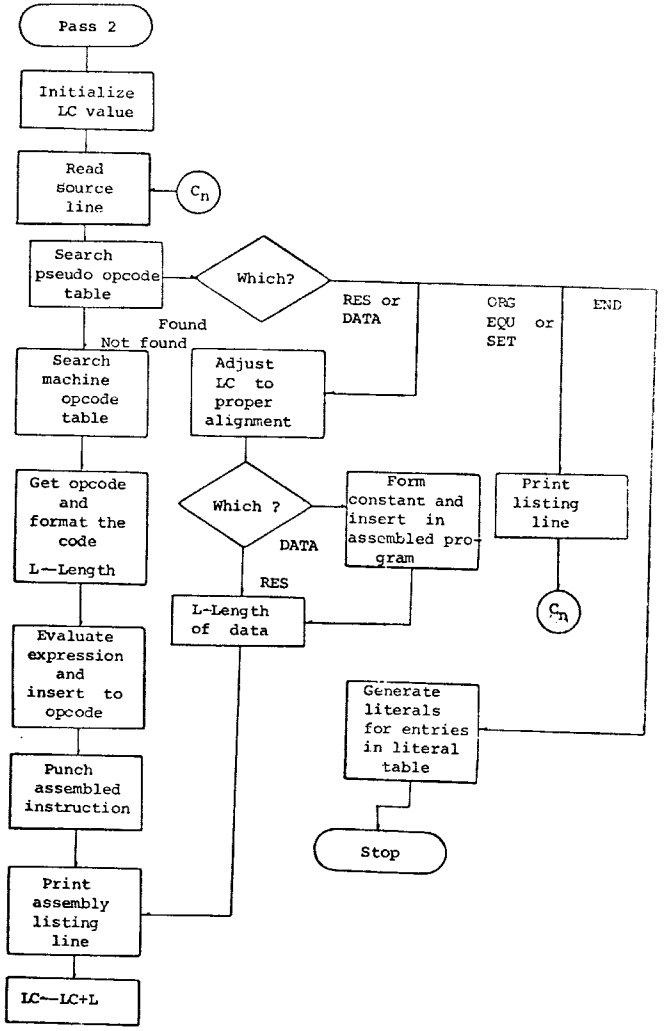

그림 6. 전형적인 패스 2의 흐름

Fig. 6. Typical flow of pass 2.

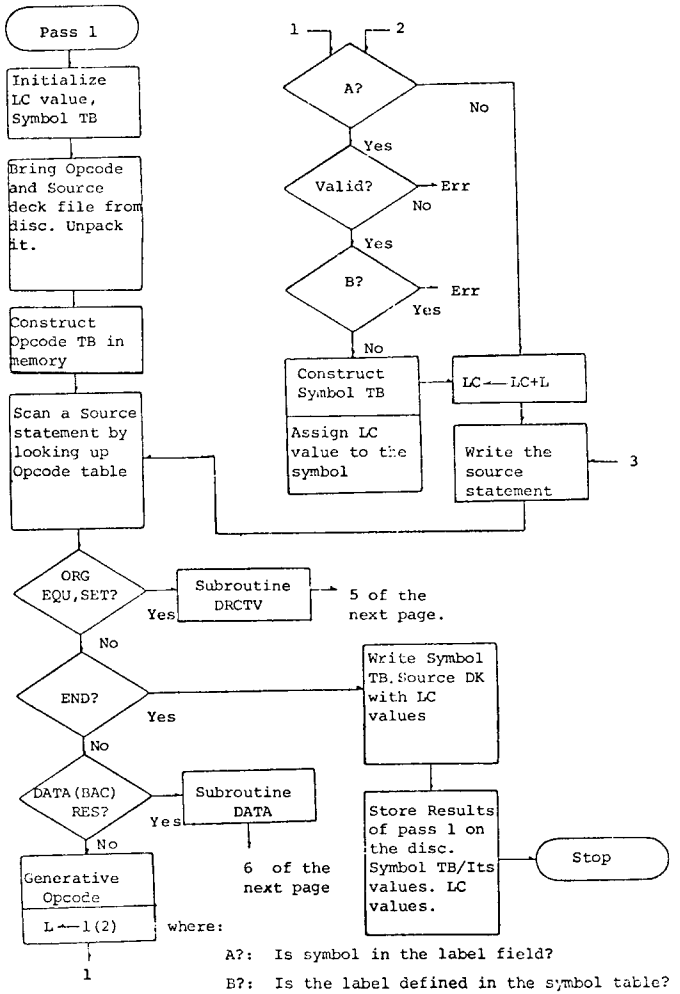

그림 7. 본 cross-assembler 패스 1의 흐름(7-1)

Fig. 7. Pass 1 of the cross-assembler (7-1).

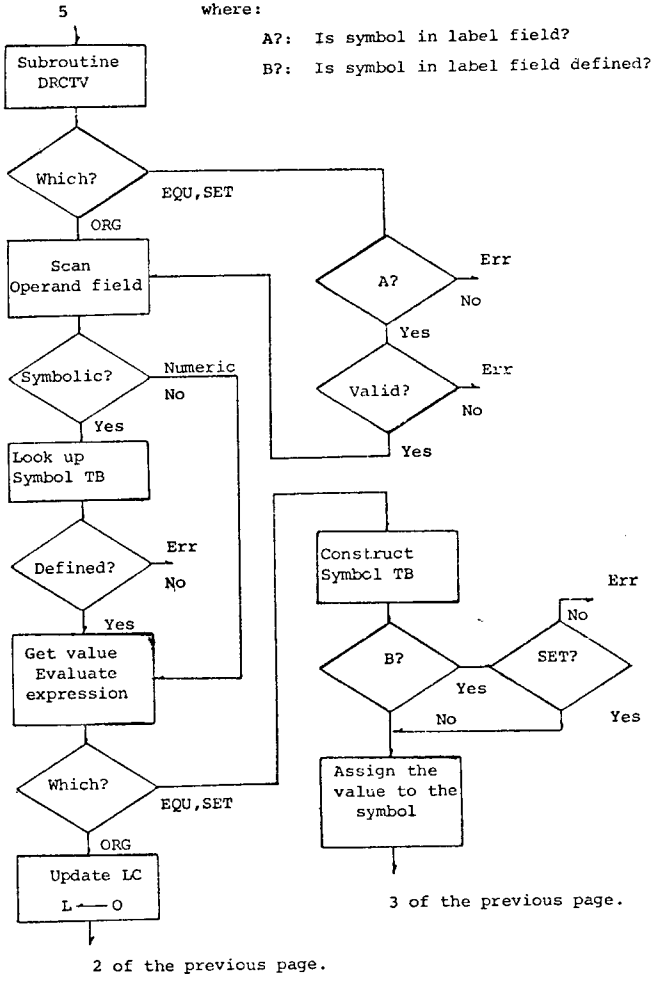

그림 7. 본 cross-assembler 패스 1의 흐름 (7-2)

Fig. 7. Pass 1 of the cross-assembler (7-2).

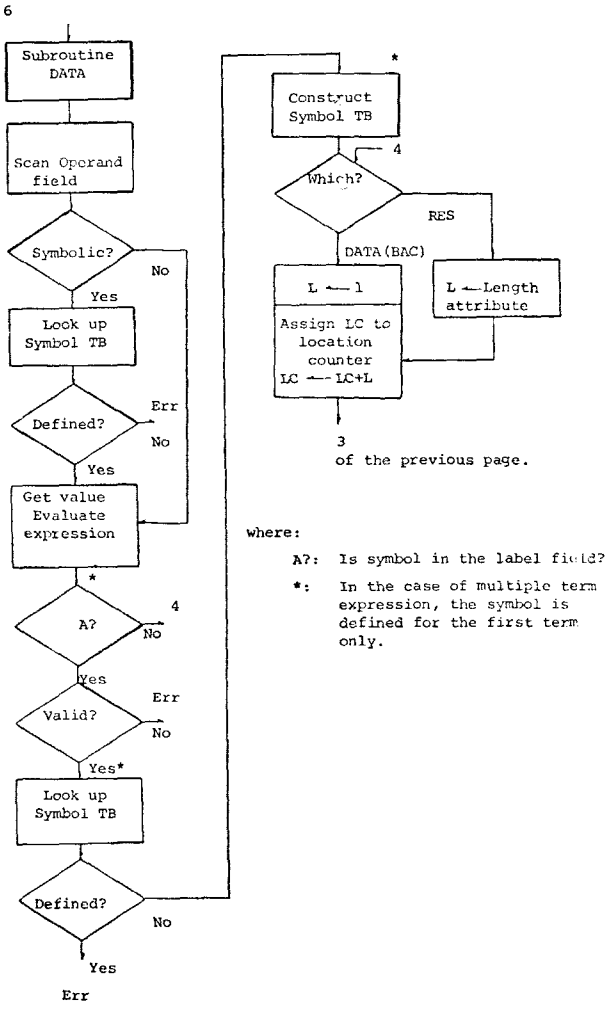

그림 7. 본 cross-assembler 패스 1 의 흐름(7-3)

Fig. 7. Pass 1 of the cross-assembler (7-3).

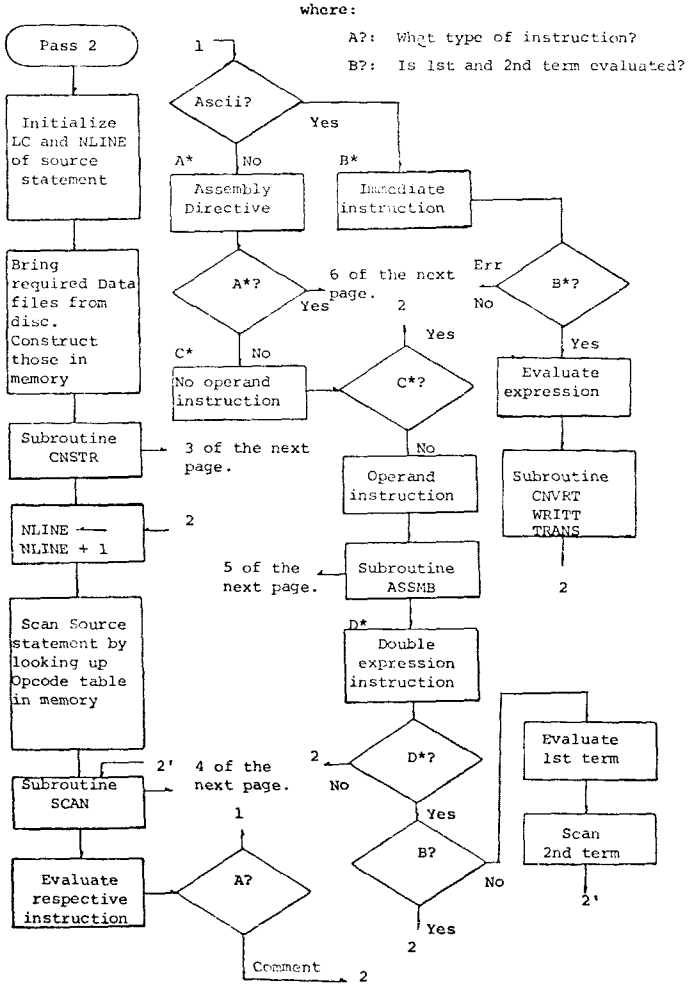

그림 8. 본 cross-assembler 패스 2 의 흐름(8-1)

Fig. 8. Pass 2 of the cross-assembler (8-1).

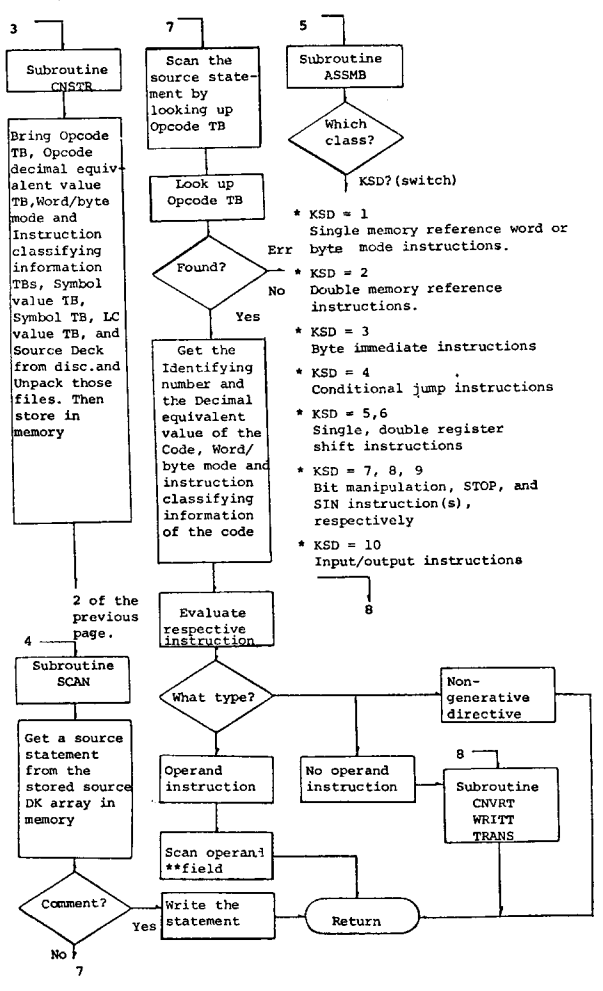

그림 8. 본 cross-assembler 패스 2 의 흐름(8-2)

Fig. 8. Pass 2 of the cross-assembler (8-2).

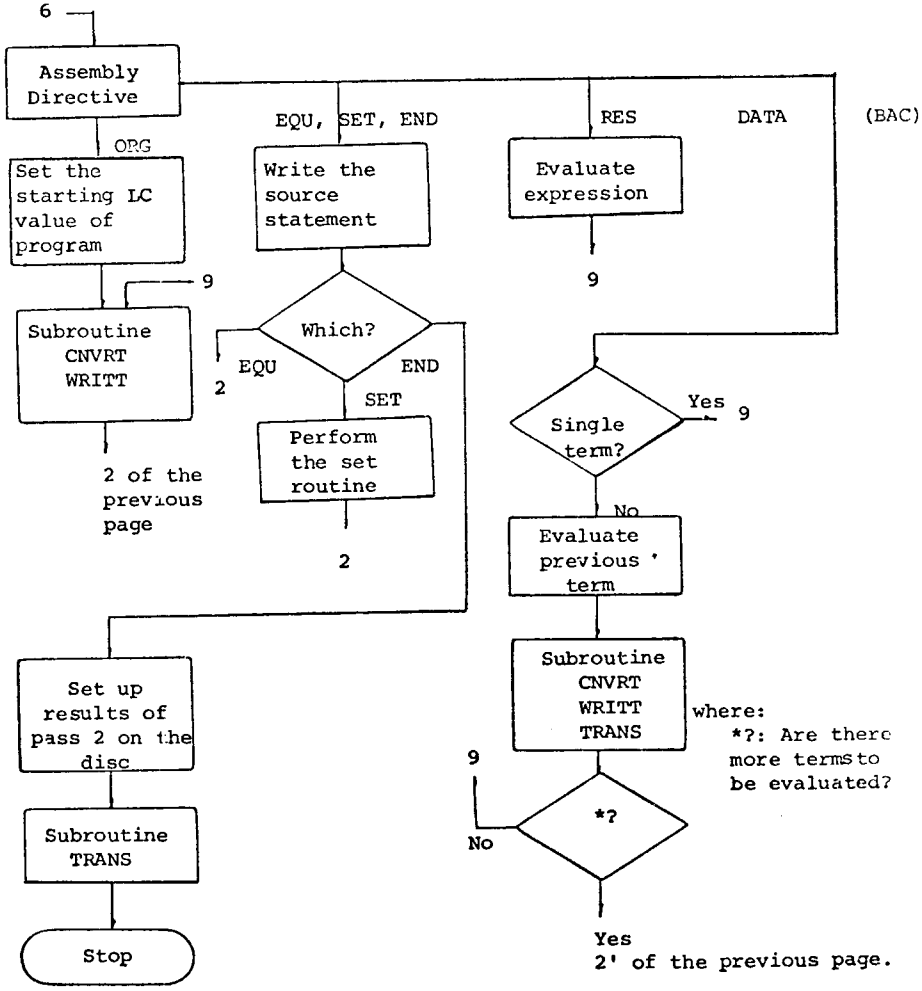

그림 8. 본 cross-assembler 패스 2 의 흐름(8-3)

Fig. 8. Pass 2 of the cross-assembler (8-3).

sembler이다. pass 1과 pass 2는 各各 1063과 1380 line의 fortran 文으로 構成된다. 먼저 본 cross-assembler의 內部的인 組織과 特性을 一般 assembler의 그것과 比較하기 위하여 flowchart로 提示하면 各各 그림5와 6 및 그림7과8과 같다. 그리고 各 pass의 內容을 全部는 소개하기 어려우므로 이들의 처음과 나중 부분을 절취하여(overlapping) 소개하면 그림 9와 10과 같다.

4.2 Cross-assembler의 능력과 Alpha-source program의 Syntax

본 cross-assembler가 認知(recognize)할 수 있는 alpha-computer의 source program은 3節에서 言及한 7種에 걸친 instruction repertoire 및 몇개의 assembly directive를 包含한 도합 180여 instruction과 8種의 基本 addressing mode를 使用하며, 英文字와 숫자(8진수, 10진수, 16진수) 및 10여개의 特殊文字와 記號의 組合으로 symbol 또는 operand expression을 形成한다.

source文은 그림 1에 보인 program과 같이 label field를 비롯하여 comment field까지의 4個 field를 包含한다. lebel field는 source文의 첫 6文字 위치의 field 內에서 어느 한 英文字의 始作으로 確認되며 첫 6文字 위치의 field가 空白(blank)이면 이 statement는 label을 包含하지 않는 것으로 認知된다. opcode field는 source文의 第8번째 column에서 始作하며 合法的으로 定義된 mnemonic opcode를 包含하고 하나 또는 그 이상의 空白에 依하여 이 field의 終了(termination)가 認知된다. operand field는 opcode field 다음에 오며 이 field의 syntax는 關聯 source文의 instruction種類에 따라 다르다. 만약 operand field가 算術式(arithmetic expression)을 包含하면 이는 다음 expression term중의 어느 하나로 構成 可能하다.

- 美貨 symbol($) ……現 location counter의 값을 나타낸다.
- symbolic term ……英文字와 數字의 組合으로 構成되는 6文字 限度의 記號項을 나타낸다.
- numeric term …… 8진수, 10진수, 16진수의 絶對數値項을 나타낸다.
- 項의 組合 …… 項의 組合은 symbolic term, numeric term, currency($) symbol 따위를 因子로 算術연산자 +나 -로 最大 2項까지 可能하다.
- single character string… 하나의 ascii文字로 英文字, 數字, 特殊文字를 包含할 수 있다.

operand field의 各種 operand는 첫 文字로 始作하는 identifier에 依하여 그 性質과 種類가 認知된다. 이를테면 identifier가 英文字이면 symbolic term의 始作으로 認知되며, $symbol이면 現 location counter의 값을 割當(assigne)하라는 信號로 풀이 된다. 또한 identifier가 *이면 address pointer에 依한 indirect addressing, ◎이면 indexed addressing, *◎이면 post-indexed addressing, :이면 hexadecimal term, ○이면 octal term, 1-9이면 decimal term따위의 始作이라는 信號로 認知된다. 첫 項 다음에 +나 -가 發見되면 算術 operand expression의 第2項이 存在한다는 信號로 풀이 된다. 그런데 계속 空白이 確認되면 operand를 要하지 않는 instruction이거나 또는 잘못 使用된 source文으로 일단 認定되어 이에 關聯한 opcode의 妥當性 여부가 table look-up routine과 關聯하여 check되고, source文의 착오가 確認되면 적절한 assembler diagnostics가 發生된다.

4.3 Assemble 過程

assemble過程은 2-pass 段階를 밟는다. pass 1의 主任務는 source文을 scan하면서 symbol table을 구축하는 일이고 pass 2는 pass 1에서 구축한 symbol table을 참조(table look-up routine)하여 8진수(octal), 10진수(decimal), 그리고 16진수(hexadecimal) 形態로 表現된 等價機械言語(object) program을 發生시키는 일이다. 이때 pass 1과 pass 2 過程은 HP 2100S computer의 working area內에서 自動的으로 連結된다. 본 assemble過程을 通하여 처리된 program을 이의 symbol table과 object code와 함께 소개하면 그림 11~13과 같다. 그림 13의 第2列은 object code의 개수를 가리킨다.

各 pass에 使用된 routine을 범주(category) 별로 要約하면;

- pass 1 *) 主 루틴(main routine)
  *) 주사 루틴(scan routine)
  *) 표참조 루틴(table look-up routine)
  *) 變換 루틴(convert routine)
- pass 2 *) 主 루틴(main routine)
  *) 주사 루틴(scan routine)
  *) 表참조 루틴(table look-up routine)
  *) 變換 루틴(convert routine)

```
      FTN
      PROGRAM AP1
C     HONG OK SOO: THE CROSS-ASSEMBLER PASS 1
C     USING THE HP2100S COMPUTER
      DIMENSION ICODE(4),ICODTB(4,177),ICD(384),
     -ITEMP(128),IRES(6),IFILN(3),IOUT(32)
      COMMON ICHAR(45),NAMLIN(500),NLINE,KSYMTB,
     -ISYMTB(6,250),NAMSYM(250),LINE(64),IRESV(6)
      DATA LC,IK,KIK,NSCT/0,1,2,1/
C*********************************************
      KSYMTB=0
      NLINE=0
C     FILL ICHAR WITH CHARACTER SET
      CALL CHFIX
C     INITIALISE SYMBOL TABLE
      DO 1 K=1,250
      DO 1 I=1,6
1     ISYMTB(I,K)=2H;;
C*********************************************
C     BRING THE STORED OPCODE FILE FROM DISC
C     AND UNPACK THE FILE, THEN CONSTRUCT OPCODE TABLE
      IFILN(1)=2HIC
      IFILN(2)=2HDT
      IFILN(3)=2HB
      CALL EXEC(14,3,ICD,384,IFILN,0)
      DO 2 K=1,177
      DO 2 I=1,2
      II=I+I
      IK=K+K+I-2
      ICODTB(II-1,K)=IAND(ICD(IK),177400B)+40B
2     ICODTB(II,K)=256*IAND(ICD(IK),255)+40B
C*********************************************
C     BRING A SOURCE LINE FROM THE SOURCE DECK ON DISC
      WRITE(1,84)
84    FORMAT(" SOURCE NAME ? ")
      READ(1,85)(IFILN(I),I=1,3)
85    FORMAT(3A2)
C     STORE FILE NAME IN ICD(1-3)
C*********************************************
      SUBROUTINE NUMM(ISTART,INUM,MN)
      COMMON ICHAR(45),NAMLIN(500),NLINE,KSYMTB,
     -ISYMTB(6,250),NAMSYM(250),LINE(64),IRESV(6)
C     MN=MODULUS=8,10,OR 16
      INUM=0
      IBEG=1
      IF(MN.EQ.16)IBEG=2
      DO 1 K=IBEG,ISTART
      DO 2 I=1,MN
      IF(IRESV(K).EQ.ICHAR(I))GO TO 3
2     CONTINUE
      GO TO 4
3     INUM=INUM*MN+I-1
1     CONTINUE
      GO TO 5
4     WRITE(6,6)NLINE,(LINE(I),I=1,64)
      WRITE(1,6)NLINE,(LINE(I),I=1,64)
6     FORMAT(1X,I2,3X,11HERR IN OPND,2X,64A1)
5     RETURN
      END
      END$
```

그림 9. 크로스 어셈블러 패스 1 (절취부분)

Fig. 9. Pass 1 of the cross-assembler (overlapping portion).

```
      FTN
      PROGRAM AP2
C     HONG OK SOO: THE CROSS-ASSEMBLER PASS 2
C     USING THE HP2100S COMPUTER
      COMMON ICHAR(45),IRESV(6),ICHARC(4),KSD,KWB,
     -KSDKSS,KSDASS,KSWEXP,IVALUE,ISYMTB(6,250),NLINE,
     -NAMSYM(250),IOUT(32),ICODTB(4,177),ITRNS,
     -KWBCD(177),KSDCD(177),IVALCD(177),LINE(64),
     -NAMDEK(500),ITEMP(128),ICD(500),KSYMTB,LNGTH
      DATA MINUS,ISTART,KODE,NSET,INOUT,ITRANS,IASCIM/0,
     -0,0,0,0,0,0/
C     INITIALISE VARIABLES
C     FILL ICHAR WITH CHARACTER SET
      GO TO 1555
C     CONSTRUCT OP-CODE AND SYMBOL TABLE WITH
C     VALUES, AND SOURCE DECK WITH LOCATION VALUES
1     CALL CNSTR(INOUT)
C*********************************************
C     UNPACK A SOURCE LINE AND SCAN THE LINE
      NLINE=NLINE+1
      DO 15 K=1,32
      KK=K+K
      LINE(KK-1)=IAND(IOUT(K),177400B)+40B
15    LINE(KK)=256*IAND(IOUT(K),255)+40B
11    CALL SCAN(IASCIM,IRESC,ITRANS,MINUS,KOUNT,
     -IRSLT,KODE)
      IF(LINE(1).EQ.ICHAR(37))GO TO 1
      IF(IASCIM.EQ.1)GO TO 2
      IF(IRSLT.EQ.1)GO TO 155
      DO 3 K=2,3
      IF(IRSLT.EQ.K)GO TO 14
3     CONTINUE
      IF(IRSLT.EQ.4)GO TO 53
      DO 4 K=5,7
      IF(IRSLT.EQ.K)GO TO 5
4     CONTINUE
      DO 6 K=78,120
      IFIN=KOUNT
      KWRD=KWRD-128
      GO TO 61
56    NWRD=ITEMP(KWRD)/256
      IF(ITEMP(KWRD).NE.0)GO TO 62
      KWRD=1
      GO TO 58
62    IF(ITEMP(KWRD).EQ.-1)GO TO 47
      KWRD=0
      CALL EXEC(14,3,ITEMP,128,IFILN,NSCT)
      NSCT=NSCT+1
      GO TO 63
48    KWRD=KWRD+1
      GO TO 57
C*********************************************
5     WRITE(6,64)
      WRITE(1,64)
64    FORMAT(3X,"ERROR IN CONSTRUCT ROUTINE")
C*********************************************
47    STOP
54    RETURN
      END
      END$
```

그림 10. 크로스 어셈블러 패스 2 (절취부분)

Fig. 10. Pass 2 of the cross-assembler (overlapping portion).

```
*****SOURCE FILE NAME FOR PASS 1 IS SUMFV ****

    ****SOURCE LINES****
NO    LOCATION  LABEL OPCODE OPERAND       COMMENT
 1                *  PROGRAM FOR CALCULATING ZI VALUES
 2                *  USING ADDRESS POINTER
 3                *  I.E.,ZI=XI+YI (I=1,5)
 4      350              ORG  350
 5      350              LDA  CONST
 6      351              STA  COUNT
 7      352              LDA  AXX
 8      353              STA  BXX
 9      354              LDA  AYY
10      355              STA  BYY
11      356              LDA  AZZ
12      357              STA  BZZ
13      358       POINT  LDA  *BXX
14      359              ADD  *BYY
15      360              STA  *BZZ
16      361              IMS  BXX
17      362              IMS  BYY
18      363              IMS  BZZ
19      364              IMS  COUNT
20      365              JMP  POINT
21      366              HLT
22      367       AXX    DATA 375
23      368       AYY    DATA 380
24      369       AZZ    DATA 385
25      370       BXX    DATA 000
26      371       BYY    DATA 000
27      372       BZZ    DATA 000
28      373       COUNT  DATA 000
29      374       CONST  DATA -5
30      375              DATA 1,2,3,4,5
31      380              DATA 1,2,3,4,5
32      385              DATA 0,0,0,0,0
33                       END


    ****SYMBOL TABLE****
NO    VALUE    SYMBOL
 1      358    POINT
 2      367    AXX
 3      368    AYY
 4      369    AZZ
 5      370    BXX
 6      371    BYY
 7      372    BZZ
 8      373    COUNT
 9      374    CONST
```

그림 11. 알파 어셈블리 프로그램과 패스 1 루틴을 통하여 발생된 심볼테이블

Fig. 11. An alpha-assembly program with its symbol table generated by the pass 1 routine.

```
****SOURCE FILE NAME FOR PASS 2 IS SUMFV ****

      *SOURCE PROGRAM WITH CODES*
NO  LOCATION  CODE          SOURCE PROGRAM
 1                *  PROGRAM FOR CALCULATING ZI VALUES
 2                *  USING ADDRESS POINTER
 3                *  I.E.,ZI=XI+YI (I=1,5)
 4      350    015E          ORG  350
 5      350    B217          LDA  CONST
 6      351    9A15          STA  COUNT
 7      352    B20E          LDA  AXX
 8      353    9A10          STA  BXX
 9      354    B20D          LDA  AYY
10      355    9A0F          STA  BYY
11      356    B20C          LDA  AZZ
12      357    9A0E          STA  BZZ
13      358    B30B  POINT   LDA  *BXX
14      359    8B0B          ADD  *BYY
15      360    9B0B          STA  *BZZ
16      361    DA0B          IMS  BXX
17      362    DA0B          IMS  BYY
18      363    DA0B          IMS  BZZ
19      364    DA0B          IMS  COUNT
20      365    F607          JMP  POINT
21      366    0800          HLT
22      367    0177  AXX     DATA 375
23      368    017C  AYY     DATA 380
24      369    0181  AZZ     DATA 385
25      370    0000  BXX     DATA 000
26      371    0000  BYY     DATA 000
27      372    0000  BZZ     DATA 000
28      373    0000  COUNT   DATA 000
29      374    FFFB  CONST   DATA -5
30      375    0001          DATA 1,2,3,4,5
30      376    0002          DATA 1,2,3,4,5
30      377    0003          DATA 1,2,3,4,5
30      378    0004          DATA 1,2,3,4,5
30      379    0005          DATA 1,2,3,4,5
31      380    0001          DATA 1,2,3,4,5
31      381    0002          DATA 1,2,3,4,5
31      382    0003          DATA 1,2,3,4,5
31      383    0004          DATA 1,2,3,4,5
31      384    0005          DATA 1,2,3,4,5
32      385    0000          DATA 0,0,0,0,0
32      386    0000          DATA 0,0,0,0,0
32      387    0000          DATA 0,0,0,0,0
32      388    0000          DATA 0,0,0,0,0
32      389    0000          DATA 0,0,0,0,0
33                           END
```

그림 12. 알파 어셈블리 프로그램과 패스 2 루틴을 통하여 발생된 기계언어 코드(그림 11과 관련)

Fig. 12. An alpha-assembly program with its machine codes, generated by the pass 2 routine. (related to Fig 11)

CODES		
DECIMAL	OCTAL	HEXADECIMAL
350	000536	015E
40	000050	0028
-19945	131027	B217
-26091	115025	9A15
-19954	131016	B20E
-26096	115020	9A10
-19955	131015	B20D
-26097	115017	9A0F
-19956	131014	B20C
-26098	115016	9A0E
-19701	131413	B30B
-29941	105413	8B0B
-25845	115413	9B0B
-9720	155010	DA08
-9720	155010	DA08
-9720	155010	DA08
-9720	155010	DA08
-2553	173007	F607
2048	004000	0800
375	000567	0177
380	000574	017C
385	000601	0181
0	000000	0000
0	000000	0000
0	000000	0000
0	000000	0000
-5	177773	FFFB
1	000001	0001
2	000002	0002
3	000003	0003
4	000004	0004
5	000005	0005
1	000001	0001
2	000002	0002
3	000003	0003
4	000004	0004
5	000005	0005
0	000000	0000
0	000000	0000
0	000000	0000
0	000000	0000
0	000000	0000

그림 13. 패스 2 루틴을 통하여 발생된 10진수, 8진수 및 16진수형의 기계언어 코드(그림 11과 12에 관련)

Fig. 13. Generated machine codes in decimal, octal, and hexadecimal form through the pass 2 routine. (*related to Fig 11 & Fig 12*)

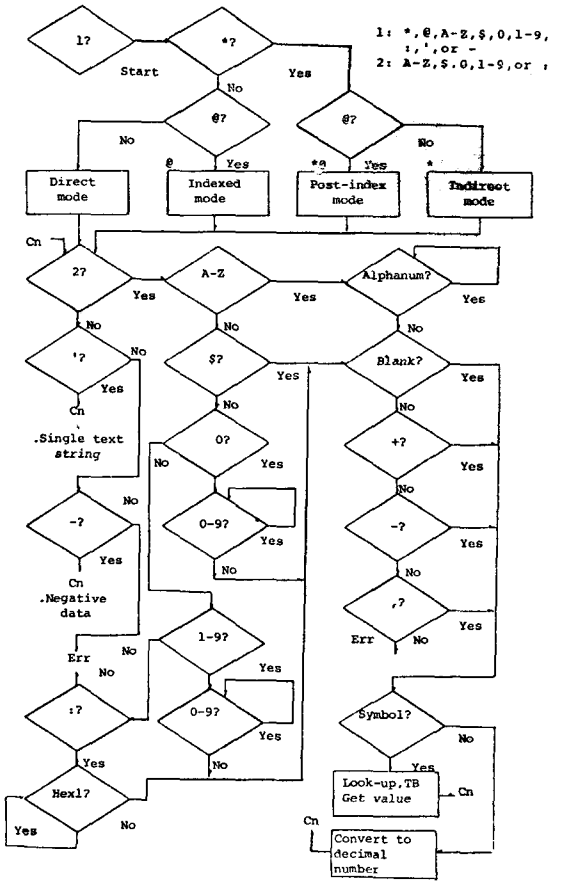

그림 14. 알파 어셈블리 프로그램의 operand field의 *syntax scam algorithm.*

Fig. 14. Typical scan algorithm for the operand field of an alpha assembly-language program.

*) 어셈블 루틴(assemble routine)이다. 이들 routine중 많은 領域을 차지하는 것은 scan routine이다. 그런데 이들 routine은 根本的으로 一般 assembler의 그것과 개념상 대동소이 하므로 일일이 說明하는 것은 省略하기로 한다. 特히 scan routine의 algoritbm을 소개하면 그림 14와 같다.

또한 본 cross-assembler는 alpha-computer 의 instruction과 hardware本質에 順應하여 設計되었으므로 任意의 instruction으로 구축되는 이의 source program은 memory 범위 0～16 K (16384)의 어느 領域에서든지 assemble可能하고 object code또한 alpha-computer의 이 범위 memory 領域 어느 곳에서든지 load되어 實行(execute)될 수 있다. 본 cross-assembler는 最大 250個의 symbol name과 500line의 alpha-source文을 HP 2100S computer의 working area內에 保有할 수 있다.

4.4 Disc에 data 수립

본 cross assembler는 이의 assembly routine을 容易하게 하기 위하여 主program과 別途로 disc 使用에 關聯한 몇가지 예비 routine을 使用하였다. 이들은 assembly에 앞서 본 cross-assembler가 必要로 하는 data source를 disc에 記憶시켜 必要時마다 呼出하여 使用토록 하기 爲한 data수립을 爲한 routine들이다. 이들 routine에 依하여 disc에 수립된 data file은 표1과 같이 要約된다. 한 例로 disc 에 opcode table을 수립하기 爲한 예비 routine을 소개하면 그림 15와 같다.

Sectors	Specifications	File Names
	1. Fixed data for the assembler	
3	Mnemonic opcode instructions including directives (Opcode table)	ICDTB
2	Opcode binary equivalent decimal numbers (Opcode value table)	OPVAL
2(each)	Mode and instruction classifying information table	KWBVL and KSDVL
	2. Data passed from pass 1 to pass 2	
4	Symbol table	SYMTB
	3. End result of pass 2	
8	Generated machine code table	CDTAB

표 1. 어셈블러에 요하는 테일러 화일

Table 1. Required data files for the assembler.

```
0001        FTN
0002        PROGRAM XYZA
0003        INTEGER ICD(384), IFILN(3), IC(4)
0004        DATA IFILN/2HIC, 2HDT, 2HB/
0005        CALL EXEC(14, 3, ICD, 384, IFILN,
            0)
0006        DO 2 I=1, 177
0007        WRITE (1, 3) I
0008    3   FORMAT("CODE", I3, "?")
0009        READ(1, 4) (IC(J), J=1, 4)
0010    4   FORMAT(4A1)
0011        II=I+I
0012        ICD(II-1) = IAND(IC(1), 177400B)
            +IAND(255, IC(2)/256)
0013    2   ICD(II) = IAND(IC(3), 177400B) +
            IAND(255, IC(4)/256)
0014        CALL EXEC (15, 3, ICD, 384, IFI-
            LN, 0)
0015        END
0016        END $

****LIST END****
```

그림 15. 크로스 어셈블러를 위하여 디스크에 Op-code 테이블을 수립하기 위한 프로그램.

Fig. 15. A program for establishing Opcode table file on disc for the cross-assembler.

4.5 Program Assembling

마지막으로 alpha-source program을 assemble하는데 다음 順序로 할 수 있도록 하였다.

1) HP 2100S computer의 VDU keyboard를 通하여 alpha-source program (assembly program)을 key in시켜으로 disc에 alpha-source file을 創造(create)한다. (:ST, S, file name, 1)

2) disc에서 cross-assembler pass 1(AP 1)을 呼出하여 주어진 pass 1의 과업을 이행한다. (:PR, AP 1)

이때 computer는 앞서 disc에 記憶시킨 alpna-source file名을 VDU에 key in시켜 주기를 要求한다. 즉 "SOURCE NAME?".

그러면 단순히 要하는 file name을 key in해 주면 된다.

3) disc에서 cross-assembler pass 2(AP 2)를 呼出하여 주어진 pass2의 과업을 이행한다.(:PR,

AP 2).

4) 만약 alpha-source program의 source statement를 修正할 경우는 단순히 edit機能을 利用하여 一般 program edit과 같은 節次로 행한다.(: ED, filename, 1)

5. 結 論

본 cross-assembler는 alpha-source program을 assemble하는 過程에서 HP 2100 S computer의 file handling특징중 다음 몇 가지 장점을 利用하였다.

1) CALL EXEC function의 導入으로 disc memory 資源을 效果的으로 利用하여 HP2100S computer의 主 memory 使用을 實効性 있게 하고 assembling을 實現하는 본 cross-assembler의 source 文 數를 한층 간결히 할 수 있었다.
(그림 9의 25번 line과 그림 10의 1367번 line 및 그림 15 참조)

2) edit function을 alpha-source 文에 적용토록 하여 必要時마다 즉각적으로 source 文의 修正을 可能케 했다.

3) typeless expression의 論理 AND 機能을 利用하여 文字 packing과 unpacking테크니크를 導入하여 使用 memory數를 줄이고 assembling routine을 한층 간단화 하였다.(그림 9의 30, 31번 line과 그림 10의 23, 24 line 및 그림 15 참조)

그러나 본 cross-assembler는 크기가 방대하고 또한 많은 變數의 使用으로 가끔 現在 memory page로부터 過多頻度의 變數참조(reference)로 因한 base page overflow를 超來하여 compiling 不能의 경우도 있었으나 이는 source文의 수정과 再配列을 通하여 克服하였다.

參 考 文 獻

1. Alpha LSI-2 Manual, COMPUTER AUTOMATION INC.
2. INTRODUCTION TO COMPUTER SCIENCE. Harry Katzan, Jr., Mason/Charter Publishers Inc. 1975.
3. MICROPROCESSORS & MICROCOMPUTERS. Branko Souček. A Wiley-Interscience Publication, John Wiley & Sons. 1976.
4. DESIGN OF DIGITAL SYSTEMS. P.C. Pittman. BSc. Cambridge University Press. 1974.
5. COMPUTER ORGANIZATION & PROGRAMMING. C. William Gear. Mc Graw-Hill, 1974.
6. SYSTEMS PROGRAMMING. John J. Donovan. McGraw-Hill, 1972.
7. ADVANCED PROGRAMMING (PROGRAMMING & OPERATING SYSTEMS). Harry Karry Katzan, Jr., 1970.
8. AN INTRODUCTION TO COMPUTER LANGUAGES. H. S. Heaps. Prentice Hall Inc. 1972.
9. INTRODUCTION TO ELECTRONIC COMPUTERS. Gordon B. Davis. McGraw-Hill, 1971.
10. FORTRAN IV. Jehosua Friedmann, Philip Greenber & Alan Hoffberg. John Wiley and Sons Inc., 1975.
11. FORTRAN TRECHNIQUES WITH SPECIAL REFERENCE TO NONNUMERICAL APPLICATIONS. A. Colin Day. Cambridge University Press, 1972.
12. COMPUTER HARDWARE & SOFTWARE(AN INTERDISCIPLINARY INTRODUCTION). Marshall D. Abrams & Philip G. Stein. Addison-Wesley Publishing Company, 1973.
13. PRINCIPLES OF DIGITAL COMPUTER DESIGN (Volume 1). ABD-Elfattah M. ABD-Alla. Arnold C. Meltzer. Prentice-Hall Inc., 1976.
14. THE DESIGN OF DIGITAL SYSTENS. John B. Peatman. McGraw-Hill, 1972.