```
%-------------------------------------------------------------------------------
% CI_1
% VER 1.4
%
% COMPUTATION INTERPRETER FOR PERSEUS-3 (CPU:MC6802)
%
% SPECIFICATION: SEE APPENDIX
%
% HAND ASSEMBLED
%
% VERSION           26/APR/2010  VER 1.0  ALMOST FINISHED
%                   20/JUN/2013  VER 1.1  MODIFIED OUT_PROGRAM_2  (OUTPUT PROGRAM LIST)
%                   26/APR/2014  VER 1.2  MODIFIED OUT_1CHA, (NEW LINE CR + LF)
%                   03/MAY/2014  VER 1.3  ADDED (‘¥’ PROCESS)
%                   30/JUN/2018  VER 1.4  ADDED (‘BS’ PROCESS)
%
% CLEAN COPY        11/FEB/2021
%-------------------------------------------------------------------------------
% COPYRIGHT (C) 2021 MITSURU YAMADA. ALL RIGHTS RESERVED.
%
%-------------------------------------------------------------------------------
% ADDRESS MAPPING:
%             SYSTEM VARIABLES          (ADDRESS  $0000 - $002F)
%             USER VARIABLES            (ADDRESS  $0080 - $00FF)
%             SYSTEM STACK              (ADDRESS  $0100 - $01FF)
%             WORKING AREA              (ADDRESS  $0200 - $03FF)
%             USER PROGRAM AREA         (ADDRESS  $1800 - $1B00)
%             SYSTEM PROGRAM            (ADDRESS  $F800 - $FFFF)
%-------------------------------------------------------------------------------
% SYSTEM VARIABLES
%             SYMBOL                    DATA
              PR_POINTER_0              $0000        POINTER OF NEW LINE
              ALU_POINTER               $0002        ALU POINTER
              VAR_POINTER               $0004        VARIABLE POINTER
              TEMP_0                    $0006
              TEMP_1                    $0007
              TEMP_2                    $0008
              TEMP_3                    $0009
              OUT_POINTER               $000A        OUTPUT POINTER
              LINE_LENGTH_0             $000C
              LINE_LENGTH_1             $000D
              DEF_LINE_0                $000E
              DEF_LINE_1                $000F
              PR_POINTER_1              $0010        PROGRAM POINTER BY 1 CHARACTER
              END_ADDRESS               $0012        END ADDRESS OF PROGRAM
              PR_POINTER_2              $0014
              PR_POINTER_3              $0020        HEAD ADDRESS OF LINE
              PR_POINTER_4              $0022        HEAD ADDRESS OF PROGRAM
              PR_POINTER_5              $0024        HEAD ADDRESS OF NEW LINE
              RUN_FLAG                  $0026        RUN/PROGRAM FLAG
              OUT_POINTER_1             $0028        OUTPUT POINTER #1
              TEMP_4H                   $002A
              TEMP_4L                   $002B
              TEMP_6                    $002C
              TEMP_7                    $002D
              TEMP_8H                   $002E
              TEMP_8L                   $002F
%
              ACIA_STATUS               $A000        ACIA(6850) STATUS REGISTER
              ACIA_DATA                 $A001        ACIA(6850) DATA REGISTER
%-------------------------------------------------------------------------------
%                           ADDRESS(HEX)  DATA(HEX)
%-------------------------------------------------------------------------------
% PROGRAM START ADDRESS           START   $F800
%
% RESET VECTOR
              .ORIGIN  $FFFE
                                FFFE    F8 00
%-------------------------------------------------------------------------------
% CI-1 MAIN PROGRAM
              .ORIGIN  $F800
%
```

```
START         LDS  #$01FF          F800   8E 01 FF      INITIALIZE STACK POINTER
              JSR  INIT_ACIA       F803   BD F8 20      INITIALIZE ACIA(6850)
              JSR  GREETING        F806   BD F9 72      GREETING MESSAGE
              JSR  INIT_REGISTER   F809   BD F8 30      INITIALIZE REGISTERS
L28           JSR  PROMPT          F80C   BD F8 D2      DISPLAY PROMPT
              JSR  IN_1LINE        F80F   BD F9 20      INPUT 1 LINE
              JSR  MAIN            F812   BD F8 60      PROCESS MAIN
              BRA  L28             F815   20 F5         REPEAT FROM PROMPT
%
%-------------------------------------------------------------------------------
              .ORIGIN  $F81A
%
INIT_ALU_PO   JMP  INIT_ALU_PO_2   F81A   7E FF 40
%
%-------------------------------------------------------------------------------
% INITIALIZE ACIA (SERIAL INTERFACE)
              .ORIGIN  $F820
%
INIT_ACIA     LDAA #$03            F820   86 03
              STAA ACIA_STATUS     F822   B7 A0 00
              LDAA #$15            F825   86 15         4800 BIT/S, 8 BIT, NO-PARITY
              STAA ACIA_STATUS     F827   B7 A0 00
              RTS                  F82A   39
%
%-------------------------------------------------------------------------------
% INITIALIZE REGISTERS
              .ORIGIN  $F830
%
INIT_REGISTER JSR  INIT_ALU_PO_2   F830   BD FF 40      INITIALIZE ALU_POINTER, CLEAR R0
              NOP                  F833   01
              NOP                  F834   01
              LDX  #$0200          F835   CE 02 00
              STX  PR_POINTER_0    F838   DF 00
              STX  PR_POINTER_5    F83A   DE 24
              LDX  #$1800          F83C   CE 18 00
              STX  PR_POINTER_1    F83F   DF 10
              STX  PR_POINTER_3    F841   DF 20
              STX  PR_POINTER_4    F843   DF 22
              LDX  #$1B00          F845   CE 1B 00
              STX  END_ADDRESS     F848   DF 12
              LDX  #$0080          F84A   CE 00 80
              STX  VAR_POINTER     F84D   DF 04
              LDX  #$0300          F84F   CE 03 00
              STX  OUT_POINTER     F852   DF 0A
              LDX  #$0304          F854   CE 03 04
              STX  OUT_POINTER_1   F857   DF 28
              CLR  RUN_FLAG        F859   7E 00 26
              RTS                  F85C   39
%
%-------------------------------------------------------------------------------
% MAIN
%      PROGRAM LINE SYNTAX ANALYSIS PART 1
              .ORIGIN  $F860
%
MAIN          LDX  PR_POINTER_5    F860   DE 24
              STX  PR_POINTER_0    F862   DF 00
MAIN_1        JMP  MAIN_2          F864   7E F8 E0
              NOP                  F867   01
              NOP                  F868   01
              NOP                  F869   01
              NOP                  F86A   01
L13           CMPB #$20            F86B   C1 20         ‘ ’?
              BNE  L01             F86D   26 06
              JSR  SHIFT_ALU_PO    F86F   BD F9 40
              JMP  MAIN_1          F872   7E F8 64
L01           CMPB #$3D            F875   C1 3D         ‘=’?
              BNE  L02             F877   26 06
              JSR  STORE_VAR       F879   BD FC 60      GO TO SUBSTITUTE PROCESS
              JMP  MAIN_1          F87C   7E F8 64
L02           CMPB #$3A            F87F   C1 3A         ‘:’?
              BNE  L03             F881   26 06
              JMP  EDITOR_ANA_5    F883   7E FF 5B      GO TO LINE EDITOR
```

```
             NOP                 F886   01
             NOP                 F887   01
             NOP                 F888   01
L03          CMPB #$0D           F889   C1 0D        'CR'?
             BNE  L04            F88B   26 06
             JMP  ANSWER_2       F88D   7E FC 30     GO TO VALUE EVALUATE OUTPUT
             NOP                 F890   01
             NOP                 F891   01
             NOP                 F892   01
L04          CMPB #$2A           F893   C1 26        '&'?
             BNE  L05            F895   26 06
             JSR  PROC_GOTO_2    F897   BD FF 50     GO TO 'GOTO' PROCESS
             JMP  MAIN_1         F89A   7E F8 64
L05          CMPB #$2B           F89D   C1 2B        '+'?
             BNE  L06            F89F   26 06
             JSR  ADD16          F8A1   BD F9 A0     GO TO ADDITION
             JMP  MAIN_1         F8A4   7E F8 64
L06          CMPB #$2D           F8A7   C1 2D        '-'?
             BNE  L07            F8A9   26 06
             JSR  SUB16          F8AB   BD FD 20     GO TO SUBTRACTION
             JMP  MAIN_1         F8AE   7E F8 64
L07          CMPB #$2F           F8B1   C1 2F        '/'?
             BNE  L08            F8B3   26 06
             JSR  DIV_COMPL16_2  F8B5   BD FE 5A     GO TO DIVISION
             JMP  MAIN_1         F8B8   7E F8 64
L08          CMPB #$3C           F8BB   C1 3F        '?'?
             BNE  L09            F8BD   26 06
             JSR  CONDITION      F8BF   BD FC 80     GO TO CONDITION BRANCH
%
             JMP MAIN_1          F8C2   7E F8 64
L09          CMPB #$2A           F8C5   C1 2A        '*'
             BNE  L11            F8C7   26 06
             JSR  MPY_COMPL16_2  F8C9   BD FD B5     GO TO MULTIPLICATION
             JMP  MAIN_1         F8CC   7E F8 64
L11          JMP  MAIN_3         F8CF   7E FC 00
%
%-------------------------------------------------------------------------------
PROMPT       LDAB #$3E           F8D2   C6 3E
             JSR  OUT_1CHA       F8D4   BD F9 90     OUTPUT CHARACTER '>'
             JSR  INIT_REGISTER  F8D7   BD F8 30     RESTORE PR_POINTER_1,3 TO INIT POSITION
             JSR  SHIFT_ALU_PO   F8DA   BD F9 40
             RTS                 F8DD   39
%
%-------------------------------------------------------------------------------
% PROGRAM LINE SYNTAX ANALYSIS PART 2
             .ORIGIN  $F8E0
%
MAIN_2       LDX  PR_POINTER_0   F8E0   DE 00
             LDAB X,$00          F8E2   E6 00
             INX                 F8E4   08
             STX  PR_POINTER_0   F8E5   DF 00
             CMPB #$21           F8E7   C1 21        '!'?
             BNE  L15            F8E9   26 06
             JSR  RUN            F8EB   BD F8 FC     GO TO 'RUN' PROCESS
             JMP  MAIN_1         F8EE   7E F8 64     RETURN TO MAIN_1
L15          CMPB #$25           F8F1   C1 25        '%'?
             BNE  L12            F8F3   26 04
             JSR  END            F8F5   BD F9 05
             RTS                 F8F8   39           RETURN TO INPUT ONE LINE
L12          JMP  L13            F8F9   7E F8 6B
%
%-------------------------------------------------------------------------------
% RUN PROCESS
%
RUN          LDAA #$80           F8FC   86 80
             STAA RUN_FLAG       F8FE   97 26
             LDX  PR_POINTER_4   F900   DE 22        SET THE TOP VALUE OF THE PROGRAM AREA
             STX  PR_POINTER_0   F902   DF 00        AS THE POINTER VALUE OF THE NEW LINE
             RTS                 F904   39           (THE LINE TO BE EXECUTED)
%
END          CLRA                F905   4F
             STAA RUN_FLAG       F906   97 26
```

```
             RTS                 F908   39
%
%-------------------------------------------------------------------------------
EDITOR ANALISYS 2
             .ORIGIN  $F90D
%
EDITOR_ANA_2 TST  RUN_FLAG       F90D   7D 00 26
             BPL  L14            F910   2A 09        IF RUN_FLAG='0', GO TO LINE EDITOR
             LDX  ALU_POINTER    F912   DE 02        ELSE RESTORE ALU POINTER
             INX                 F914   08
             INX                 F915   08
             STX_ALU_POINTER     F916   DF 02
             JMP MAIN_1          F918   7E F8 64     RETURN TO MAIN_1
L14          JMP EDITOR_ANA_4    F91B   7E FA AD
             RTS                 F91E   39
%-------------------------------------------------------------------------------
INPUT_ONE_LINE
             .ORIGIN  $F920
%
IN_1LINE     JMP IN_1LINE2       F920   7E FF C0
%
%-------------------------------------------------------------------------------
% VALUE EVALUATE OUTPUT
             .ORIGIN  $F935
%
ANSWER       JSR  CMPL16_TO_ASCII F935   BD FA 53
             JSR  OUT_1LINE      F938   BD F9 E0
             JSR  INIT_ALU_PO    F93B   BD F8 1A
             RTS                 F93E   39
%
%-------------------------------------------------------------------------------
% SHIFT ALU POINTER
             .ORIGIN  $F940
%
SHIFT_ALU_PO LDX  ALU_POINTER    F940   DE 02
             DEX                 F942   09
             DEX                 F942   09
             STX  ALU_POINTER    F944   DF 02
             CLR  X,$00          F946   6F 00
             CLR  X,$01          F948   6F 01
             RTS                 F94A   39
%
%-------------------------------------------------------------------------------
% NUM_1
             .ORIGIN  $F950
%
NUM_1        SUBB #$30           F950   C0 30
             STAB TEMP_2         F952   D7 08
             CLRB                F954   5F
             LDX  ALU_POINTER    F955   DE 02
             LDAA X,$01          F957   A6 01
             LDAB X,$00          F959   E6 00
             ASLA                F95B   48
             ROLB                F95C   59
             STAA TEMP_1         F95D   97 07
             STAB TEMP_0         F95F   D7 06
             ASLA                F961   48
             ROLB                F962   59
             ASLA                F963   48
             ROLB                F964   59
             ADDA TEMP_1         F965   9B 07
             ADCB TEMP_0         F967   D9 06
             ADDA TEMP_2         F969   9B 08
             ADCB #$00           F96B   C9 00
             STAA X,$01          F96D   A7 01
             STAB X,$00          F96F   E7 00
             RTS                 F971   39
%
%-------------------------------------------------------------------------------
% GREETING MESSAGE
             .ORIGIN  $F972
%
```

```
GREETING      LDX  #$FEF8          F972   CE FE F8
              STX  OUT_POINTER     F975   DF 0A
              JSR  OUT_1LINE       F977   BD F9 E0
              RTS                  F97A   39
%
%-------------------------------------------------------------------------------
% INPUT 1 CHARACTER FROM SERIAL INTERFACE (RS-232C)
% RECEIVED DATA IS STORED IN ACCA
              .ORIGIN  $F980
%
IN_1CHA       LDAA ACIA_STATUS     F980   B6 A0 00
              LSRA                 F983   44
              BCC  IN_1CHA         F984   24 FA
              LDAA ACIA_DATA       F986   B6 A0 01
              RTS                  F989   39
%
%-------------------------------------------------------------------------------
% OUTPUT 1 CHARACTER TO SERIAL INTERFACE (RS-232C)
% TRANSMIT DATA IS ACCB
              .ORIGIN  $F990
%
OUT_1CHA      LDAA ACIA_STATUS     F990   B6 A0 00
              LSRA                 F993   44
              LSRA                 F994   44
              BCC  OUT_1CHA        F995   24 F9
              STAB ACIA_DATA       F997   F7 A0 01
              BRA OUT_1CHA_2       F99A   20 17         (MODIFY 'RTS' DUE TO NEW LINE 'CR','LF')
%
%-------------------------------------------------------------------------------
% OUTPUT ONE CHARACTER (NEW LINE 'CR','LF')
              .ORIGIN  $F99C
%
L16           LDAB #$0A            F99C   C6 0A         PUT 'LF' INTO ACCB
              BRA  OUT_1CHA        F99E   20 F0         OUTPUT ONE CHARACTER AGAIN
%
%-------------------------------------------------------------------------------
% 16 BIT ADDITION
              .ORIGIN  $F9A0
%
ADD16         LDX  ALU_POINTER     F9A0   DE 02
              LDAA X,$01           F9A2   A6 01
              ADDA X,$03           F9A4   AB 03
              STAA X,$03           F9A6   A7 03
              LDAA X,$00           F9A8   A6 00
              ADCA X,$02           F9AA   A9 02
              STAA X,$02           F9AC   A7 02
              INX                  F9AE   08
              INX                  F9AF   08
              STX  ALU_POINTER     F9B0   DF 02
              RTS                  F9B2   39
%
%-------------------------------------------------------------------------------
% OUTPUT ONE CHARACTER (NEW LINE 'CR','LF')
              .ORIGIN  $F9B3
%
OUT_1CHA_2    CMPB #$0D            F9B3   C1 0D         'CR'?
              BEQ  L16             F9B5   27 E5         IF 'CR' CODE THEN BRANCH
              CMPB #$0A            F9B7   C1 0A         'LF'?
              BEQ  L17             F9B9   27 01         IF NOT 'LF' THEN EXIT
L18           RTS                  F9BB   39
%
L17           LDAB #$0D            F9BC   C6 0D         IF 'LF', RETURN ACCB TO CR THEN EXIT
              BRA  L18             F9BE   20 FB
%
%-------------------------------------------------------------------------------
% 16 BIT SUBTRACTION
              .ORIGIN  $F9C0
%
SUB16_1       LDAA TEMP_1          F9C0   96 07
              SUBA TEMP_3          F9C2   90 09
              STAA TEMP_1          F9C4   97 07
              LDAA TEMP_0          F9C6   96 06
```

```
             SBCA TEMP_2         F9C8   92 08
             STAA TEMP_0         F9CA   97 06
             RTS                 F9CC   39
%
%-------------------------------------------------------------------------------
% 16 BIT ADDITION
             .ORIGIN  $F9D0
%
ADD16_1      LDAA TEMP_1         F9D0   96 07
             ADDA TEMP_3         F9D2   9B 09
             STAA TEMP_1         F9D4   97 07
             LDAA TEMP_0         F9D6   96 06
             ADCA TEMP_2         F9D8   99 08
             STAA TEMP_0         F9DA   97 06
             RTS                 F9DC   39
%
%-------------------------------------------------------------------------------
% OUTPUT ONE LINE
             .ORIGIN  $F9E0
%
OUT_1LINE    LDX  OUT_POITER     F9E0   DE 0A
L19          LDAB X,$00          F9E2   E6 00
             INX                 F9E4   08
             JSR  OUT_1CHA       F9E5   BD F9 90
             CMPB #$0D           F9E8   C1 0D
             BNE L19             F9EA   26 F6
             RTS                 F9EC   39
%
%-------------------------------------------------------------------------------
% OUTPUT PROGRAM LIST
             .ORIGIN  $F9F0
%
OUT_PROGRAM  LDX  PR_POINTER_4   F9F0   DE 22
L20          LDAB X,$00          F9F2   E6 00
             INX                 F9F4   08
             JSR  OUT_1CHA       F9F5   BD F9 90
             CMPB #$25           F9F8   C1 25
             BNE  L20            F9FA   26 F6
             RTS                 F9FC   39
%
%-------------------------------------------------------------------------------
% CONVERT HEX TO ASCII
             .ORIGIN  $FA00
%
HEX2ASCII    LDX  ALU_POINTER    FA00   DE 02
             LDAA X,$00          FA02   A6 00
             STAA TEMP_0         FA04   97 06
             LDAA X,$01          FA06   A6 01
             STAA TEMP_1         FA08   97 07
             LDX  #10000(DEC)    FA0A   CE 27 10
             JSR  DIGIT_COUNT    FA0D   BD FA 40
             STAB X,$01          FA10   E7 01
             LDX  #1000(DEC)     FA12   CE 03 E8
             JSR  DIGIT_COUNT    FA15   BD FA 40
             STAB X,$02          FA18   E7 02
             LDX  #100(DEC)      FA1A   CE 00 64
             JSR  DIGIT_COUNT    FA1D   BD FA 40
             STAB X,$03          FA20   E7 03
             LDX  #10(DEC)       FA22   CE 00 0A
             JSR  DIGIT_COUNT    FA25   BD FA 40
             STAB X,$04          FA28   E7 04
             LDX  #01(DEC)       FA2A   CE 00 01
             JSR  DIGIT_COUNT    FA2D   BD FA 40
             STAB X,$05          FA30   E7 05
             LDAB #$0D           FA32   C6 0D
             STAB X,$06          FA34   E7 06
             RTS                 FA36   39
%
%-------------------------------------------------------------------------------
% COUNTING DIGIT
             .ORIGIN  $FA40
%
```

```
DIGIT_COUNT   STX  TEMP_2          FA40   DF 08
              CLRB                FA42   5F
L22           JSR  SUB16_1         FA43   BD F9 C0
              BCS  L21             FA46   25 03
              INCB                FA48   5C
              BRA  L22             FA49   20 F8
L21           JSR  ADD16_1         FA4B   BD F9 D0
              ADDB #$30            FA4E   CB 30
              LDX  OUT_POINTER     FA50   DE 0A
              RTS                 FA52   39
%
CMP16_2_ASCI  JSR  DET_SIGN        FA53   BD FA 60      DET. SIGN AND CONVERT 2'S COMPL. OF R0
              JSR  HEX2ASCII       FA56   BD FA 00      CONVERT HEX TO ASCII
              JSR  REMOVE_0        FA59   BD FD 00      ZERO SUPPRESS
              RTS                 FA5C   39
%-------------------------------------------------------------------------------
% SIGN DETECTION
              .ORIGIN  $FA60
%
DET_SIGN      LDX  ALU_POINTER     FA60   DE 02
              TST  X,$00           FA62   6D 00         IS MSB OF R0 = 0 ?  PLUS?
              BPL  L23             FA64   2A 0B
              LDX  OUT_POINTER     FA66   DE 0A
              LDAB #$2D            FA68   C6 2D         '-' CODE
              STAB X,$00           FA6A   E7 00
              JSR  CMPL2S          FA6C   BD FC E0      2'S COMPLEMENT OF R0
              BRA  L24             FA6F   20 06
L23           LDX  OUT_POINTER     FA71   DE 0A
              LDAB #$20            FA73   C6 20         ' ' CODE
              STAB X,$00           FA75   E7 00
L24           RTS                 FA77   39
%-------------------------------------------------------------------------------
% LINE EDITOR ANALYSIS
              .ORIGIN  $FA7E
%
EDITOR_ANA    LDX  PR_POINTER_4    FA7E   DE 22         PUT HEAD ADDRESS OF PROGRAM
              STX  PR_POINTER_3    FA80   DF 20         TO HEAD ADDRESS OF LINE POINTER
              STX  PR_POINTER_1    FA82   DF 10
EDITOR_ANA_1  LDX  PR_POINTER_1    FA84   DE 10         PROGRAM POINTER BY 1 CHARACTER
              LDAB X,$00           FA86   E6 00         STORE DATA POINTED PR_POINT_1 TO ACCB
              INX                 FA88   08           POINTER + 1
              STX  PR_POINTER_1    FA89   DF 10
              CMPB #$20            FA8B   C1 20         SPACE CODE ?
              BNE  L31             FA8D   26 05
              NOP                 FA8F   01           IF SPACE CODE THEN DO NOT PROCESSING
              NOP                 FA90   01
              NOP                 FA91   01
              BRA  EDITOR_ANA_1    FA92   20 F0
L31           CMPB #$3A            FA94   C1 3A         ':' ?
              BNE  L32             FA96   26 03
              JMP  CMP_LINE_NUM    FA98   7E FA B9      IF ':' THEN COMPARE LINE NUMBER
L32           CMPB #$25            FA9B   C1 25         '%' (PROGRAM END) ?
              BEQ  L38             FA9D   27 71         IF '%' THEN MOVE '%' AND INSERT NEW LINE
              CMPB #$30            FA9F   C1 30
              BLT  EDITOR_ANA_1    FAA1   2D E1
              CMPB #$39            FAA3   C1 39
              BGT  EDITOR_ANA_1    FAA5   2E 0D
              JSR  NUM_1           FAA7   BD F9 50
              BRA  EDITOR_ANA_1    FAAA   20 D8
              RTS                 FAAC   39
%
EDITOR_ANA_4  LDX  PR_POINTER_4    FAAD   DE 22
              STX  PR_POINTER_3    FAAF   DF 20
              STX  PR_POINTER_1    FAB1   DF 10
EDITOR_ANA_3  JSR  SHIFT_ALU_PO    FAB3   BD F9 40      AT HEAD OF LINE, POINTER+1 & CLEAR R0
              JMP  EDITOR_ANA_1    FAB6   7E FA 84
%
CMP_LINE_NUM  JSR  CMP16_1         FAB9   BD FA F0      COMPARE LINE NUMBER R1(NEW)-R0
              BNE  L34             FABC   26 11
              JSR  PULL_ALU_PO     FABE   BD FA FB      IF LINE NUM IS SAME, THEN SWAP LINE PROCESS
              JSR  SWAP_LINE       FAC1   BD FB 70
              LDAA LINE_LINGTH_0   FAC4   96 0C         IF NEW LINE LINGTH=0 THEN DO NOT MOVE_1
```

```
             CMPA #$00           FAC6   81 00
             NOP                 FAC8   01
             BEQ  L36            FAC9   27 03
             JSR  MOVE_1         FACB   BD FB 20
L36          RTS                 FACE   39
L34          BCC  L35            FACF   24 0E         R1<R0
             JMP  L40            FAD1   7E FB 40
%
% IF NEW LINE NUMBER < EXIST LINE NUMBER, THEN MOVE EXIST LINE AND INSERT NEW LINE
%
L37          LDX  PR_POINTER_3   FAD4   DE 20
             STX  PR_POINTER_1   FAD6   DF 10
             JSR  INSERT_LINE    FAD8   BD FB 50      INSERT NEW LINE
             JSR  MOV_1          FADB   BD FB 20
             RTS                 FADE   39
%
L35          JSR  PULL_ALU_PO    FADF   BD FA FB
%
             LDX  PR_POINTER_1   FAE2   DE 10
             JSR  GET_LINE_LENG  FAE4   BD FB E0      IF NEW LINE NUMBER > EXIST LINE NUMBER,
             STX  PR_POINTER_1   FAE7   DF 10         THEN ADVANCE POINTER TO HEAD OF NEXT LINE
             STX  PR_POINTER_3   FAE9   DF 20
             JMP  EDITOR_ANA_3   FAEB   7E FA B3
%
             NOP                 FAEE   01
             NOP                 FAEF   01
%
CMP16_1      LDX  ALU_POINTER    FAF0   DE 02
             LDAA X,$03          FAF2   A6 03         PROCESSING FOR LOWER 8 BIT ONLY
             SUBA X,$01          FAF4   A0 01
             NOP                 FAF6   01
             NOP                 FAF7   01
             NOP                 FAF8   01
             NOP                 FAF9   01
             RTS                 FAFA   39
%
PULL_ALU_PO  INX                 FAFB   08
             INX                 FAFC   08
             STX  ALU_POINTER    FAFD   DF 02
             RTS                 FADF   39
%
ADD_LENG_OFS STX  TEMP_0         FB00   DF 06         TEMP_2,3 IS (TEMP_0,1 + DEF_LINE_0,1)
             LDAA TEMP_1         FB02   96 07
             ADDA DEF_LINE_1     FB04   9B 0F
             STAA TEMP_3         FB06   97 09
             LDAA TEMP_0         FB08   96 06
             ADCA DEF_LINE_0     FB0A   99 0E
             STAA TEMP_2         FB0C   97 08
             RTS                 FB0E   39
%
             NOP                 FB0F   01
%
L38          LDX  PR_POINTER_5   FB10   DE 24
             JSR  GET_LINE_LENG  FB12   BD FB E0
             CMPB #$00           FB15   C1 00
             BNE  L39            FB17   26 01
             RTS                 FB19   39            IF LENGTH OF NEW LINE = 0, DO NOT PROCESSING
L39          JMP  L37            FB1A   7E FA D4      ELSE INSERT NEW LINE
%
             NOP                 FB1D   01
             NOP                 FB1E   01
             NOP                 FB1F   01
%
MOVE_1       LDX  PR_POINTER_5   FB20   DE 24         HEAD ADDRESS OF NEW LINE
             STX  TEMP_0         FB22   DF 06
             LDX  PR_POINTER_3   FB24   DE 20         HEAD ADDRESS OF MOVE_1 EXECUTION AREA
             STX  TEMP_2         FB26   DF 08
L41          LDX  TEMP_0         FB28   DE 06
             LDAA X,$00          FB2A   A6 00
             INX                 FB2C   08
             STX  TEMP_0         FB2D   DF 06
             LDX  TEMP_2         FB2F   DE 08
```

```
             STAA X,$00          FB31   A7 00
             INX                 FB33   08
             STX  TEMP_2         FB34   DF 08
             CMPA #$0D           FB36   81 0D
             BNE  L41            FB38   26 EE
             RTS                 FB3A   39
%
             .ORIGIN  $FB40
%
L40          JSR  PULL_ALU_PO    FB40   BD FA FB     RETURN ALU POINTER
             LDX  PR_POINTER_5   FB43   DE 24
             JSR  GET_LINE_LENG  FB45   BD FB E0
             CMPB #$00           FB48   C1 00
             BNE  L29            FB4A   26 01
             RTS                 FB4C   39           IF LENGTH OF NEW LINE = 0, DO NOT PROCESSING
L29          JMP  L37            FB4D   7E FA D4     ELSE INSERT NEW LINE
%
INSERT_LINE  LDX  PR_POINTER_5   FB50   DE 24        HEAD ADDRESS OF NEW LINE
             JSR  GET_LINE_LENG  FB52   BD FB E0
             STAB DEF_LINE_1     FB55   D7 0F
             CLRB                FB57   5F
             STAB DEF_LINE_0     FB58   D7 0E
             LDX  PR_POINTER_1   FB5A   DE 10
             STX  PR_POINTER_2   FB5C   DF 14
             JMP  MOVE_2         FB5E   7E FB 8F
%
%-------------------------------------------------------------------------------
% SWAP LINE
             .ORIGIN  $FB70
%
SWAP_LINE    LDX  PR_POINTER_3   FB70   DE 20        CALCULATE LENGTH OF EXIST LINE
             JSR  GET_LINE_LENG  FB72   BD FB E0
             STAB LINE_LENGTH_1  FB75   D7 0D
             STX  PR_POINTER_2   FB77   DF 14
             LDX  PR_POINTER_5   FB79   DE 24
             JSR  GET_LINE_LENG  FB7B   BD FB E0     CALCULATE LENGTH OF NEW LINE
             STAB LINE_LENGTH_0  FB7E   D7 0C        LENGTH OF NEW LINE
             SUBB LINE_LENGTH_1  FB80   D0 0D        (NEW LINE LENGTH) - (EXIST LINE LENGTH)
             STAB DEF_LINE_1     FB82   D7 0F
             BMI  L44            FB84   2B 03        CONVERT LENGTH DIFFERENCE TO 16BIT
             CLRB                FB86   5F
             BRA  L45            FB87   20 02
L44          LDAB #$FF           FB89   C6 FF
L45          STAB DEF_LINE_0     FB8B   D7 0E
             BCS  MOVE_3         FB8D   25 1E
MOVE_2       LDX  END_ADDRESS    FB8F   DE 12
             STX  TEMP_0         FB91   DF 06        SOURCE ADDRESS TO MOVE
             JSR  ADD_LENG_OFS   FB93   BD FB 00
L46          LDX  TEMP_0         FB96   DE 06
             LDAA X,$00          FB98   A6 00
             DEX                 FB9A   09
             STX  TEMP_0         FB9B   DF 06
             LDX  TEMP_2         FB9D   DE 08        DESTINATION ADDRESS TO MOVE
             STAA X,$00          FB9F   A7 00
             DEX                 FBA1   09
             STX  TEMP_2         FBA2   DF 08
             LDX  TEMP_0         FBA4   DE 06
             INX                 FBA6   08
             CPX  PR_POINTER_2   FBA7   9C 14        WHEN SOURCE ADDRESS = PR_POINTER_2
             BEQ  L47            FBA9   27 1F        THEN STOP TO MOVE
             BRA  L46            FBAB   20 E9
%
MOVE_3       LDX  PR_POINTER_2   FBAD   DE 14
             STX  TEMP_0         FBAF   DF 06
             JSR  ADD_LENG_OFS   FBB1   BD FB 00
L48          LDX  TEMP_0         FBB4   DE 06
             LDAA X,$00          FBB6   A6 00
             INX                 FBB8   08
             STX  TEMP_0         FBB9   DF 06
             LDX  TEMP_2         FBBB   DE 08
             STAA X,$00          FBBD   A7 00
             INX                 FBBF   08
```

```
             STX  TEMP_2          FBC0   DF 08
             LDX  TEMP_0          FBC2   DE 06
             CPX  END_ADDRESS     FBC4   9C 12
             BEQ  L47             FBC6   27 02
             BRA  L48             FBC8   20 EA
L47          RTS                  FBCA   39
%
%-------------------------------------------------------------------------------
GET LINE LENGTH
             .ORIGIN  $FBE0
%
GET_LINE_LENGLDAB #$01            FBE0   C6 01
L50          LDAA X,$00           FBE2   A6 00
             CMPA #$3A            FBE4   81 3A        ':' ?
             BEQ  L51             FBE6   27 0A
L53          CMPA #$0D            FBE8   81 0D
             BEQ  L49             FBEA   27 04        WHEN 'CR', THEN EXIT
             INCB                 FBEC   5C           NUMBER OF CHARACTER COUNTER + 1
             INX                  FBED   08           ADVANCE LINE POINTER TO NEXT POSITION
             BRA  L50             FBEE   20 F2
L49          INX                  FBF0   08
             RTS                  FBF1   39
%
L51          LDAA X,$01           FBF2   A6 01
             CMPA #$0D            FBF4   81 0D        ':' FOLLOWED BY 'CR' ?
             BEQ  L52             FBF6   27 04
             LDAA X,$00           FBF8   A6 00        ELSE RECOVER ACCA
             BRA  L53             FBFA   20 EC
%
L52          LDAB #$00            FBFC   C6 00        IF ':' FOLLOWED BY 'CR' THEN
             RTS                  FBFE   39           CLEAR NUMBER OF CHARACTER COUNTER
%
%-------------------------------------------------------------------------------
% PROGRAM LINE SYNTAX ANALYSIS PART 3

             .ORIGIN  $FC00
%
MAIN_3       CMPB #$24            FC00   C1 24        '$'?
             BNE  L54             FC02   26 06
             JSR  MEM2VAR         FC04   BD FF 20     POINTER VARIABLE PROCESS
             JMP  MAIN_1          FC07   7E F8 64
L54          CMPB #$22            FC0A   C1 22        '"'?
             BNE  L55             FC0E   26 06
             JMP  TEXT_1          FC0E   7E FE 68     COMMENT PROCESS
             JMP  MAIN_1          FC11   7E F8 64
L55          CMPB #$3B            FC14   C1 3B        ';'?
             BNE  L56             FC16   26 06
             JMP  DO_NOT_OUT_2    FC18   7E FF 67     NO OUTPUT
             JMP  MAIN_1          FC1B   7E F8 64
L56          CMPB #$2E            FC1E   C1 2E        '.'?
             BNE  L57             FC20   26 06
             JSR  OUT_PROGRAM_2   FC22   BD FF A0     OUTPUT PROGRAM LIST (MODIFIED)
             RTS                  FC25   39
%
             NOP                  FC26   01
             NOP                  FC27   01
L57          JMP  VAR2MEM         FC28   7E FD CD     GO TO PROCESS OF WRITE MEMORY
%
%-------------------------------------------------------------------------------
             .ORIGIN  $FC30
ANSWER_2     JSR  ANSWER          FC30   BD F9 35
             TST  RUN_FLAG        FC33   7D 00 26     IF RUN_FLAG='1' THEN RETURN MAIN_1
             BPL  L58             FC36   2A 03
             JMP  MAIN_1          FC38   7E F8 64     IF RUN_FLAG='0' THEN GO TO IN_1LINE
L58          RTS                  FC3B   39
%
%-------------------------------------------------------------------------------
LOAD VARIABLES A TO Z INTO R0
             .ORIGIN  $FC40
%
LOAD_VAR     ASLB                 FC40   58           SHIFT THE ASCII CODE BY 1 BIT
             STAB VAR_POINTER+1   FC41   D7 05        TO THE VARIABLE POINTER
```

```
             LDX  VAR_POINTER     FC43   DE 04        LOAD VARIABLE POINTER INTO IX
             LDAA X,$00           FC45   A6 00
             LDAB X,$01           FC47   E6 01
             LDX  ALU_POINTER     FC49   DE 02        ASSIGN VALUE OF VARIABLE TO R0
             STAA X,$00           FC4B   A7 00
             STAB X,$01           FC4D   E7 01
             RTS                  FC4F   39
%
%-------------------------------------------------------------------------------
STORE R0 INTO SELECTED VARIABLES A TO Z
             .ORIGIN  $FC60
%
STORE_VAR    LDX  PRG_POINTER_0   FC60   DE 00
             LDAB X,$00           FC62   E6 00        PARSE THE NEXT CHARACTER AFTER '='
             ASLB                 FC64   58           CONVERT ASCII CODE TO VAR. POINTER
             STAB VAR_POINTER+1   FC65   D7 05
             LDX  ALU_POINTER     FC67   DE 02
             LDAA X,$00           FC69   A6 00
             LDAB X,$01           FC6B   E6 01
             LDX  VAR_POINTER     FC6D   DE 04
             STAA X,$00           FC6F   A7 00        ASSIGN VALUE OF R0 TO SELECTED VARIABLE
             STAB X,$01           FC71   E7 01
             RTS                  FC73   39
%
%-------------------------------------------------------------------------------
PROCESSING CONDITIONAL STATEMENTS
             .ORIGIN  $FC80
%
CONDITION    LDAB X,$00           FC80   E6 00
             CMPB #$3D            FC82   C1 3D        '='?
             BNE  L59             FC84   26 0D
             JSR  CMP16_2         FC86   BD FE B0     IF '=' THE COMPARE R1-R0
             BEQ  L60             FC89   27 07        IF R1-R0=0 THEN RETURN TO MAIN_1
L61          LDX  PR_POINTER_0    FC8B   DE 00        IF R1-R0 NOT EQUAL 0 THEN
             JSR  GET_LINE_LENG   FC8D   BD FB E0     MOVE POINTER TO THE NEXT 'CR' CHARACTER
             STX  PR_POINTER_0    FC90   DF 00
L60          RTS                  FC92   39
%
L59          CMPB #$3C            FC93   C1 3C        '<'?
             BNE  L60             FC95   26 FB
             JMP CMP16_3          FC97   7E FE D0     IF '<' THEN COMPARE R1-R0
             BCS  L60             FC9A   25 F6        IF R1-R0<0, THE RETURN TO MAIN_1
             BRA  L61             FC9C   20 ED        ELSE MOVE POINTER TO THE NEXT 'CR'
%
%-------------------------------------------------------------------------------
% PROCESSING GOTO STATEMENTS
             .ORIGIN  $FCA0
%
PROC_GOTO    LDX  PR_POINTER_4    FCA0   DE 22
             STX  PR_POINTER_1    FCA2   DF 10
             STX  PR_POINTER_3    FCA4   DF 20
L62          JSR  SHIFT_ALU_PO    FCA6   BD F9 40     ADVANCE POINTER AT THE END OF LINE
L65          LDX  PR_POINTER_1    FCA9   DE 10
             LDAB X,$00           FCAB   E6 00        INPUT 1 CHARACTER
             INX                  FCAD   08
             STX  PR_POINTER_1    FCAE   DF 10
             COMP #$20            FCB0   C1 20        SPACE CODE?
             BEQ  L65             FCB2   27 F5        IF SPACE CODE THEN DO NOTHING
             CMPB #$3A            FCB4   C1 3A        ':'?
             BEQ  L63             FCB6   27 07        IF ':' THEN COMPARE LINE NUMBER
             JSR  NUM_1           FCB8   BD F9 50     ELSE GOTO PROCESSING NUMBERS
             BRA  L65             FCBB   20 EC        AFTER 1CHA. PROCESS, INPUT NEXT 1CHA.
%
             NOP                  FCBD   01
             NOP                  FCBE   01
%
L63          JSR  CMP16_1         FCBF   BD FA F0     R1-R0 COMPARE LINE NUMBER
             BEQ  L64             FCC2   27 0E
             JSR  PULL_ALU_PO     FCC4   BD FA FB
             LDX  PR_POINTER_1    FCC7   DE 10        RESTORE ALU POINTER
             JSR  GET_LINE_LENG   FCC9   BD FB E0     IF LINE NUMBERS DO NOT MATCH,
             STX  PR_POINTER_1    FCCC   DF 10        MOVE POINTER TO NEXT LINE HEAD
```

```
             STX  PR_POINTER_3    FCCE   DF 20
             BRA  L62             FCD0   20 D4
%
L64          JSR  PULL_ALU_PO     FCD2   BD FA FB      IF LINE NUMBER MATCH,
             JSR  PULL_ALU_PO     FCD5   BD FA FB      RETURN ALU POINTER
             JSR  PULL_ALU_PO     FCD8   BD FA FB
             LDX  PR_POINTER_3    FCDB   DE 20         ASSIGN DETECTED HEAD ADDRESS
             STX  PR_POINTER_0    FCDD   DF 00         TO EXECUTION POINTER_0
             RTS                  FCDF   39            RETURN TO MAIN
%
%-------------------------------------------------------------------------------
% CONVERT 2'S COMPLEMENT
%
CMPL2S       LDX  ALU_POINTER     FCE0   DE 02
             DEX                  FCE2   09            ADVANCE POINTER
             DEX                  FCE3   09
             LDAA X,$03           FCE4   A6 03         LOAD R1 LOWER
             COMA                 FCE4   43            INVERSE
             STAA X,$01           FCE7   A7 01         STORE R0 LOWER
             LDAA X,$02           FCE9   A6 02         LOAD R1 UPPER
             COMA                 FCEB   43            INVERSE
             STAA X,$00           FCEC   A7 00         STORE R1 LOWER
             DEX                  FCEE   09            ADVANCE POINTER
             DEX                  FCEF   09
             STX  ALU_POINTER     FCF0   DF 02
             LDAA #$01            FCF2   86 01
             STAA X,$01           FCF4   A7 01         LOAD #$0001 TO R0
             CLR  X,$00           FCF6   6F 00
             JSR  ADD16           FCF8   BD F9 A0      STORE (R1 + R0) INTO R0
             RTS                  FCFB   39
%
%-------------------------------------------------------------------------------
% ZERO SUPPRESS
             .ORIGIN  $FD00
%
REMOVE_ZERO  LDX  OUT_POINTER     FD00   DE 0A
L66          LDAB X,$01           FD02   E6 01
             CMPB #$30            FD04   C1 30         IS THE NEXT CHARACTER ZERO?
             BNE  L03             FD06   26 0D
             LDAB X,$00           FD08   E6 00         SEND CURRENT POSITION CHARACTER
             STAB X,$01           FD0A   E7 01         TO THE NEXT POSITION
             LDAB #$20            FD0C   C6 20         SET CURRENT POSITION CHARACTER
             STAB X,$00           FD0E   E7 00         TO A SPACE CODE
             INX                  FD10   08
             CPX  OUT_POINTER_1   FD11   9C 28         IS THE LAST CHARACTER ?
             BNE  L66             FD13   26 ED
             RTS                  FD15   39
%
%-------------------------------------------------------------------------------
% SUBTRACTION 16BIT 2'S COMPLEMENT
%      SUBSTITUTE R1-R0 INTO R0
%      POINTER POSITION IS R0 BEFOR & AFTER
             .ORIGIN  $FD20
%
SUB_16       LDX  ALU_POINTER     FD20   DE 02
             LDAA X,$03           FD22   A6 03
             SUBA X,$01           FD24   A0 01
             STAA X,$03           FD26   A7 03
             LDAA X,$02           FD28   A6 02
             SBCA X,$00           FD2A   A2 00
             STAA X,$02           FD2C   A7 02
             INX                  FD2E   08
             INX                  FD2F   08
             STX  ALU_POINTER     FD30   DF 02
             RTS                  FD32   39
%
%-------------------------------------------------------------------------------
% MULTIPLICATION 16BIT UNSIGNED
             .ORIGIN  $FD3C
%
MPY16_1      LDX  ALU_POINTER     FD3C   DE 02
             LDAA X,$02           FD3E   A6 02         R1 UPPER
```

```
             STAA TEMP_4H        FD40   97 2A
             LDAA X,$03          FD42   A6 03        R1 LOWER
             STAA TEMP_4L        FD44   97 2B
             CLR  X,$02          FD46   6F 02        CLEAR R1
             CLR  X,$03          FD48   6F 03
             LDAA #$10           FD4A   86 10
             STAA TEMP_6         FD4C   97 2C        INITIALIZE LOOP COUNTER
L68          ROR  TEMP_4H        FD4E   76 00 2A     SHIFT RIGHT 16BIT TEMP_4
             ROR  TEMP_4L        FD51   76 00 2B
             BCC  L67            FD54   24 07
             JSR  ADD16          FD56   BD F9 A0     16BIT ADD
             DEX                 FD59   09           DO NOT MOVE POINTER
             DEX                 FD5A   09
             STX  ALU_POINTER    FD5B   DF 02
L67          ASL  X,$01          FD5D   68 01        SHIFT LEFT 16BIT (R0 LOWER)
             ROL  X,$00          FD5F   69 00                     (R0 UPPER)
             DEC  TEMP_6         FD61   7A 00 2C     LOOP COUNTER -1
             BNE  L68            FD64   26 E8        REPEAT SHIFT&ADD UNTIL COUNT.=0
             INX                 FD66   08
             INX                 FD67   08           STEP BACK POINTER
             STX  ALU_POINTER    FD68   DF 02
             RTS                 FD6A   39
%
%-------------------------------------------------------------------------------
% MULTIPLICATION 16BIT 2'S COMPLIMENT
             .ORIGIN  $FD70
%
MPY_COMPL16  LDX  ALU_POINTER    FD70   DE 02
             LDAA X,$00          FD72   A6 00
             STAA TEMP_8H        FD74   97 2E        SAVING R0 TO TEMP_8
             LDAA X,$01          FD76   A6 01
             STAA TEMP_8L        FD78   97 2F
             CLRB                FD7A   5F           CLEAR SIGN DETECTION REGISTER
             TST  X,$02          FD7B   6D 02        IS R1 POSITIVE ?
             BPL  L69            FD7D   2A 11
             INX                 FD7F   08
             INX                 FD80   08
             STX  ALU_POINTER    FD81   DF 02        MOVE POINTER TO R1
             JSR  CMPL2S         FD83   BD FC E0     IF NEGATIVE THEN CONVERT 2'S CMP.
             LDAB #$01           FD86   C6 01        COPY 2'S COMP. OF Y TO R1
             LDAA X,$00          FD88   A6 00
             STAA X,$02          FD8A   A7 02
             LDAA X,$01          FD8C   A6 01
             STAA X,$03          FD8E   A7 03
%
L69          LDAA TEMP_8H        FD90   96 2E        MOVE TEMP_8 TO R0
             STAA X,$00          FD92   A7 00
             LDAA TEMP_8L        FD94   96 2F
             STAA X,$01          FD96   A7 01
             TST  X,$00          FD98   6D 00        IS R0 POSITIVE ?
             BPL  L70            FD9A   2A 11
             JSR  CMPL2S         FD9C   BD FC E0     IF NEGATIVE THEN CONVERT 2'S CMP.
             ADDB #$01           FD9F   CB 01
             LDAA X,$00          FDA1   A6 00        COPY 2'S COMP. OF X TO R0
             STAA X,$02          FDA3   A7 02
             LDAA X,$01          FDA5   A6 01
             STAA X,$03          FDA7   A7 03
             INX                 FDA9   08           RESET POINTER AT R0
             INX                 FDAA   08
             STX  ALU_POINTER    FDAB   DF 02
L70          RTS                 FDAD   39
%
MPY_COMPL16_3LSRB                FDAE   54
             BCC  L71            FDAF   24 03        IF SIGN OF RESULT IS POSITIVE, THEN EXIT
             JSR  CMPL2S         FDB1   BD FC E0     IF SIGN OF RESULT IS NEGATIVE, CONV. 2'CMP
L71          RTS                 FDB4   39
%
MPY_COMPL16_2JSR  MPY_16_1       FDB5   BD FD 3C     NO NEED TO CONVERT
             RTS                 FDB8   39           EQUIVALENT TO UNSIGNED MPY
%
%-------------------------------------------------------------------------------
% POINTER VARIABLE PROCESS
```

```
            .ORIGIN  $FDC0
%
POINTER_VAR LDX  ALU_POINTER    FDC0   DE 02
            LDAA X,$02          FDC2   A6 02        LOAD UPPER OF VALUE
            LDAB X,$03          FDC4   E6 03        LOAD LOWER OF VALUE
            LDX  X,$00          FDC6   EE 00        LOAD ADDRESS TO IX
            STAA X,$00          FDC8   A7 00        ASSIGN UPPER
            STAB X,$01          FDCA   E7 01        ASSIGN LOWER
            RTS                 FDCC   39
%
%-------------------------------------------------------------------------------
% WRITE VALUE DIRECTLY TO MEMORY
%
VAR2MEM     CMPB #$23           FDCD   C1 23        '#' ?
            BNE  L72            FDCF   26 06
            JSR  POINTER_VAR    FDD1   BD FD C0
            JMP  MAIN_1         FDD4   7E F8 64
L72         CMPB #$5C           FDD7   C1 5C        '¥' ?
            BNE  L74            FDD9   26 06
            JSR  OUT1CHA2       FDD8   BD FD F1     OUTPUT CR CODE
            JMP  MAIN_1         FDDE   7E F8 64
L74         CMPB #$41           FDE1   C1 41        'A' TO 'Z' ?
            BCS  L73            FDE3   25 06
            JSR  LOAD_VAR       FDE5   BD FC 40     GO TO VARIABLE PROCESS
            JMP  MAIN_1         FDE8   7E F8 64
L73         JSR  NUM_1          FDEB   BD F9 50     GO TO NUMERAL PROCESS
            JMP  MAIN_1         FDEE   7E F8 64
%
%-------------------------------------------------------------------------------
% OUTPUT 'CR' CODE
%      (WHEN '¥', THEN OUTPUT 'CR' CODE ONLY)
            .ORIGIN  $FDF1
%
OUT1CHA2    LDAB #$0D           FDF1   C6 0D        SET 'CR' CODE
OUT1CHA3    LDAA ACIA_STATUS    FDF3   B6 A0 00     OUTPUT ONE CHARACTER
            LSRA                FDF6   44
            LSRA                FDF7   44
            BCC  OUT1CHA2       FDF8   24 F9
            STAB ACIA_DATA      FDFA   F7 A0 01
            RTS                 FDFD   39
%
%-------------------------------------------------------------------------------
% DIVISION 16BIT UNSIGNED
            .ORIGIN  $FE00
%
DIV16_1     LDX  ALU_POINTER    FE00   DE 02
            LDAA X,$02          FE02   A6 02        R1 UPPER (DIVIDEND)
            STAA TEMP_4H        FE04   97 2A
            LDAA X,$03          FE06   A6 03        R1 LOWER (DIVIDEND)
            STAA TEMP_4L        FE08   97 2B
            LDAA X,$00          FE0A   A6 00        R0 UPPER (DIVISOR)
            STAA TEMP_2         FE0C   97 08
            LDAA X,$01          FE0E   A6 01        R0 LOWER (DIVISOR)
            STAA TEMP_3         FE10   97 09
            CLRA                FE12   4F
            STAA TEMP_6         FE13   97 2C        CLEAR QUOTIENT REGISTER
            STAA TEMP_7         FE15   97 2D
            STAA TEMP_0         FE17   97 06        CLEAR DIVIDEND REGISTER
            STAA TEMP_1         FE19   97 07
            LDAA #$10           FE1B   86 10
            STAA TEMP_8H        FE1D   97 2E        INITIALIZE LOOP COUNTER
%
L77         ASL  TEMP_4L        FE1F   78 00 2B     SHIFT DIVIDEND(32BIT) BY 1BIT LEFT
            ROL  TEMP_4H        FE22   79 00 2A
            ROL  TEMP_1         FE25   79 00 07
            ROL  TEMP_0         FE28   79 00 06
%
            JSR  SUB16_1        FE2B   BD F9 C0     SUBTRACTION
            BCS  L75            FE2E   25 0B
            ASL  TEMP_7         FE30   78 00 2D     IF SUBTRACTION SUCCESS,
            ROL  TEMP_6         FE33   79 00 2C     SHIFT QUOTIENT BY 1BIT LEFT
            INC  TEMP_7         FE36   7C 00 2D     QUOTIENT + 1
```

```
             BRA  L76            FE39   20 09
%
L75          JSR  ADD16_1        FE3B   BD F9 D0     IF SUBT. NOT SUCCESS, ADD DIVISOR
             ASL  TEMP_7         FE3E   78 00 2D     SHIFT QUOTIENT BY 1BIT LEFT
             ROL  TEMP_6         FE41   79 00 2C
L76          DEC  TEMP_8H        FE44   7A 00 2E
             BNE  L77            FE47   26 D6
%
             LDAA TEMP_6         FE49   96 2C        STORE RESULT QUOTIENT TO R0
             STAA X,$00          FE4B   A7 00
             LDAA TEMP_7         FE4D   96 2D
             STAA X,$01          FE4F   A7 01
             LDAA TEMP_0         FE51   96 06        STORE RESULT REMAINDER TO R1
             STAA X,$02          FE53   A7 02
             LDAA TEMP_1         FE55   96 07
             STAA X,$03          FE57   A7 03
             RTS                 FE59   39
%
%-------------------------------------------------------------------------------
% DIVISION 16BIT 2'S COMPLIMENT
%
DIV_COMPL16_2JSR  MPY_COMPL16    FE5A   BD FD 70     CONVERT DIVIDEND & DIVISOR TO 2'S COMP.
             JSR  DIV16_1        FE5D   BD FE 00     DIVISION 16BIT UNSIGNED
             JSR  MPY_COMPL16_3  FE60   BD FD AE     CONVERT QUOTIENT TO 2' COMP.
             RTS                 FE63   39
%
%-------------------------------------------------------------------------------
% COMMENT STATEMENT PROCESS
             .ORIGIN  $FE68
%
TEXT_1       LDX  OUT_POINTER    FE68   DE 0A
             STX  TEMP_0         FE6A   DF 06
L79          LDX  PR_POINTER_0   FE6C   DE 00        NEW LINE POINTER
             LDAA X,$00          FE6E   A6 00        COPY NEW LINE TO OUTPUT 1 CHARACTER
             INX                 FE70   08
             STX  PR_POINTER_0   FE71   DF 00
             LDX  TEMP_0         FE73   DE 06
             STAA X,$00          FE75   A7 00
             INX                 FE77   08
             STX  TEMP_0         FE78   DF 06
             CMPA #$3B           FE7A   81 3B        IF ';' CODE DETECTED, DO NOT OUTPUT 1LINE
             BEQ  DO_NOT_OUT     FE7C   27 1A
             BRA  L81            FE7E   20 0F
%
L78          JSR  OUT_1LINE      FE80   BD F9 E0     OUTPUT ONE LINE
             JSR  INIT_ALU_PO    FE83   BD F8 1A
             TST  RUN_FLAG       FE86   7D 00 26
             BPL  L80            FE89   2A 03
             JMP  MAIN_1         FE8B   7E F8 64
L80          RTS                 FE8E   39
%
L81          CMPA #$0D           FE8F   81 0D        IF 'CR' CODE DETECTED, STOP COPY,
             BEQ  L78            FE91   27 ED        AND OUTPUT ONE LINE
             BRA  L79            FE93   20 D7
%
             NOP                 FE95   01
             NOP                 FE96   01
             NOP                 FE97   01
%
DO_NOT_OUT   JMP  DO_NOT_OUT_2   FE98   7E FF 67
%
%-------------------------------------------------------------------------------
% COMPARE 16BIT
             .ORIGIN  $FEB0
%
CMP16_2      LDX  ALU_POINTER    FEB0   DE 02
             LDAA X,$03          FEB2   A6 03
             SUBA X,$01          FEB4   A0 01
             BNE  L82            FEB6   26 05        IF LOWER 8BIT ARE NOT ZERO, BRANCH
             LDAA X,$02          FEB8   A6 02
             SBCA X,$00          FEBA   A2 00
             RTS                 FEBC   39
```

```
%
L82           LDAA X,$02          FEBD   A6 02
              SBCA X,$00          FEBF   A2 00
              BNE  L83            FEC1   26 02        IF ALSO UPPER 8BIT ARE NOT ZERO, EXIT
              LDAB #$01           FEC3   C6 01        IF UPPER 8BIT ARE ZERO, CLEAR ZERO FLAG
L83           RTS                 FEC5   39
%
%-------------------------------------------------------------------------------
% COMPARE 16BIT (MEASURES WHEN THE SIGN IS DIFFERENT)
%      REVERSE THE RESULTS OF A COMPARISON USING CARRY.
              .ORIGIN  $FED0
%
CMP16_3       LDX  ALU_POINTER    FED0   DE 02
              TST  X,$02          FED2   6D 02
              BPL  L84            FED4   2A 02
              LDAB #$01           FED6   C6 01
L84           TST  X,$00          FFD8   6D 00
              BPL  L85            FEDA   2A 02
              ADDB #$01           FEDC   CB 01
L85           LSRB                FEDE   54
              BCS  L86            FEDF   25 0B
              JSR  CMP16_2        FEE1   BD FE B0
              BCS  L87            FEE4   25 03
L88           JMP  CONDITION_L03  FEE6   7E FC 8B
L87           JMP  CONDITION_L02  FEE9   7E FC 92
L86           JSR  CMP16_2        FEEC   BD FE B0
              BCC  L87            FEEF   24 F8
              BRA  L88            FEF1   20 F3
%
%-------------------------------------------------------------------------------
% READ VALUE DIRECTLY FROM MEMORY
              .ORIGIN  $FF20
%
MEM2VAR       LDX  PR_POINTER_0   FF20   DE 00
              LDAB X,$00          FF22   E6 00        LOAD THE NEXT CHARACTER OF '$' TO ACCB
              ASLB                FF24   58           CONVERT ASCII CODE TO POINTER
              STAB VAR_POINTER+1  FF25   D7 05
              LDX  ALU_POINTER    FF27   DE 02
              LDX  X,$00          FF29   EE 00
              LDAA X,$00          FF2B   A6 00
              LDAB X,$01          FF2D   E6 01
              LDX  VAR_POINTER    FF2F   DE 04
              STAA X,$00          FF31   A7 00
              STAB X,$01          FF33   E7 01
              RTS                 FF35   39
%
%-------------------------------------------------------------------------------
% INITIALIZE ALU POINTER
              .ORIGIN  $FF40
%
INIT_ALU_PO_2 LDX  #$00C0         FF40   CE 00 C0     INITIALIZE ALU POINTER
              STX  ALU_POINTER    FF43   DF 02
              CLR  X,$00          FF45   6F 00        CLEAR R0
              CLR  X,$01          FF47   6F 01
              RTS                 FF49   39
%
%-------------------------------------------------------------------------------
% GOTO PROCESS
              .ORIGIN  $FF50
%
PROC_GOTO_2   TST  RUN_FLAG       FF50   7D 00 26
              BPL  L89            FF53   2A 03        IF RUN_FLAG='0', THEN EXIT
              JMP  PROC_GOTO      FF55   7E FC A0     ELSE GOTO PROCESS
L89           JMP  MAIN_1         FF58   7E F8 64
%
%-------------------------------------------------------------------------------
% EDITOR ANALYSIS
              .ORIGIN  $FF5B
%
EDITOR_ANA_5  LDX  ALU_POINTER    FF5B   DE 02
              LDAA X,$01          FF5D   A6 01
              BEQ  L90            FF5F   27 03        IF LINE NUMBER R0=0, THEN EXIT
```

```
             JMP  EDITOR_ANA_2    FF61   7E F9 0D      ELSE GO TO EDITOR PROCESS
L90          JMP  MAIN_1          FF64   7E F8 64
%
%-------------------------------------------------------------------------------
% OUTPUT PROCESS
             .ORIGIN  $FF67
%
DO_NOT_OUT_2 LDX  PR_POINTER_0    FF67   DE 00
             LDAB X,$00           FF69   E6 00         LOAD THE NEXT 1 CHARACTER OF ';'
             INX                  FF6B   08            ADVACE POINTER
             STX  PR_POINTER_0    FF6C   DF 00
             CMPB #$3B            FF6E   C1 3B         IS THE NEXT CHARACTER ';' ?
             BEQ  L91             FF70   27 0C         IF ';' THEN OUTPUT WITH DO NOT MOVE TO
%                                                      NEW LINE
L93          JSR  INIT_ALU_PO     FF72   BD F8 1A      ELSE NORMAL PROCESS WITHOUT OUTPUT
             TST  RUN_FLAG        FF75   7D 00 26
             BPL  L92             FF78   2A 03
             JMP  MAIN_1          FF7A   7E F8 64
L92          RTS                  FF7D   39
%
L91          INX                  FF7E   08            ADVANCE POINTER
             STX  PR_POINTER_0    FF7F   DF 00
             JSR  CMPL16_TO_ASCII FF81   BD FA 53      CONVERT 2'S COMPLIMENT TO ASCII CODE
             JSR  OUT_1LINE_2     FF84   BD FF 89      ONE LINE OUTPUT WITHOUT MOVE NEW LINE
             BRA L03              FF87   20 E9
%
OUT_LINE_2   LDX  OUT_POINTER     FF89   DE 0A         ONE LINE OUTPUT WITHOUT MOVE NEW LINE
L95          LDAB X,$00           FF8B   E6 00
             INX                  FF8D   08
             CMPB #$0D            FF8E   C1 0D         IF 'CR'DETECTED, EXIT BEFORE OUTPUT
             BEQ  L94             FF90   27 05
             JSR  OUT_1CHA        FF92   BD F9 90      OUTPUT ONE CHARACTER
             BRA  L95             FF95   20 F4
L94          RTS                  FF97   39
%
%-------------------------------------------------------------------------------
% OUTPUT PROGRAM LIST WHTH ONE LINE AND PAUSE
%      'CR' ; OUTPUT EVERYTHING AFTER THAT OTHERWISE OUTPUT THE FOLLOWING LINE
             .ORIGIN  $FFA0
%
OUT_PROGRAM_2LDX  PR_POINTER_4    FFA0   DE 22
L97          LDAB X,$00           FFA2   E6 00
             INX                  FFA4   08
             CMPB #$0D            FFA5   C1 0D
             BEQ  L96             FFA7   27 08
             JSR  OUT_1CHA        FFA9   BD F9 90
             CMPB #$25            FFAC   C1 25
             BNE  L97             FFAE   26 F2
             RTS                  FFB0   39
%
L96          JSR  OUT_1CHA        FFB1   BD F9 90
             JSR  IN_1CHA         FFB4   BD F9 80
             CMPA #$0D            FFB7   81 0D
             BEQ  L98             FFB9   27 02
             BRA  L97             FFBB   20 E5
%
L98          JMP  L20             FFBD   7E F9 F2
%
%-------------------------------------------------------------------------------
% IN ONE LINE WITH BACK SPACE ('BS') PROCESS
             .ORIGIN  $FFC0
%
IN_1LINE2    LDX  PR_POINTER_5    FFC0   DE 24
             STX  PR_POINTER_0    FFC2   DF 00
L25          JSR  IN_1CHA         FFC4   BD F9 80
             TAB                  FFC7   16
             JSR  OUT_1CHA        FFC8   BD F9 90
             CMPB #$08            FFCB   C1 08         'BS' CODE ?
             BEQ  L26             FFCD   27 0A
             STAB X,$00           FFCF   E7 00
             INX                  FFD1   08
L27          STX  PR_POINTER_0    FFD2   DF 00
```

```
             CMPB #$0D           FFD4   C1 0D
             BNE  L25            FFD6   26 EC
             RTS                 FFD8   39
%
L26          CPX  #$0200         FFD9   8C 02 00      IS POINTER AT INITIAL POSITION?
             BEQ  L27            FFDC   27 F4
             DEX                 FFDE   09            POINTER -1
             BRA  L27            FFDF   20 F1
%
%-------------------------------------------------------------------------------
% GREETING MESSAGE
%            .ORIGIN  $FEF8
%
                                 FEF8   43 4F 4D 50 55 54 41 54    'COMPUTAT'
                                 FF00   49 4F 4E 20 49 4E 54 45    'ION INTE'
                                 FF08   52 50 52 45 54 45 52 20    'RPRETER '
                                 FF10   31 20 20 59 41 4D 41 44    '1  YAMAD'
                                 FF18   41 20 32 30 30 39 0D       'A 2009 '
%
%-------------------------------------------------------------------------------
%
% END OF PROGRAM
%
%
%-------------------------------------------------------------------------------
% Appendix
%            CI-1 language specification
%
% 1. Execution
%      The prompt is '>'
%      Execution can be done by typing a single line directly or by executing a command '!'.
%
% 2. Programming line editor
%      Program line recognition, start a line with a line number 1 to 255 followed by':'.
%      Inserting a program line, if the same line number does not exist in the program,
%      a new line will be inserted.
%      Replace a program line, if the same line number exists, that line will be replaced.
%      Deleting a program line, end a line with just the line number.
%
% 3. Line composition
%      A line consists of a integer value, a variable, a function, and a space.
%      The line ends with a 'CR' code. The evaluation value is stored in the arithmetic stack R0.
%      With spaces, the value of R0 is pushed to R1 and R2 in that order.
%      The evaluation of the operator moves back one place on the arithmetic stack.
%
% 4. Execution the program
%      >!'CR'
%
% 5. View the programing list
%      >.'CR'
%
% 6. Numerical value
%      Integer -32768 to 32767, but negative numbers are written as 0A- if the absolute is A.
%
% 7. Variables
%      There are 26,  from A to Z.
%
% 8. Substitution
%      The substitution operator = is substituted from left to right.
%      Example, 20=A    Substitute 20 into A.
%
% 9. RPN arithmetic function
%      Addition,'+', Subtraction,'-', Multiplication,'*', Division,'/'.
%      RPN arithmetic is a space-separated number or variable followed by an operator.
%      Example, A 5+=B   Add A and 5 and substitute in B.   (A+B)*(C-D) is A B+ C D-*.
%
% 10. Control function
%      Condition branch,'?', Jump,'&', Write memory,'#', Read memory,'$', Text,'"'
%
% 11. Display output of register R0 or variable value
%      If the end of the line is only 'CR', print the line evaluation value and break the line.
%      Display a variable, example A'CR'.
```

```
%      If put ‘;’ at the end of the line, the evaluation value is not displayed.
%      If put ‘;;’, the evaluation value is displayed and not break the line.
%
% 12. Conditional branch
%      A conditional statement is ‘?’ after two terms.
%      After the ‘?’ followed by a condition ‘=’ or ‘>’.
%      After that is the statement when the condition is satisfied.
%      Example, 100: A B?= 200&        If A and B are equal, jump to line number 200.
%      Example, 120: A B?< C 1+=C 130&  If A is less than B, add +1 to C and jump to line 130.
%
% 13. Jump statement
%      Putting ‘&’ after a number or variable. Value of R0 is the jump destination line number.
%      Example,  100: 200&   Jump to line number 200.
%      Example,  100: A&     Jump to line number A
%
% 14. Write directly to memory
%      Function ‘#’, R0 is a memory address. R1 is the value to be written.
%      Example, A B#    Write value of variable A to memory address B.
%
% 15. Read directly from memory
%      Function ‘$’,
%      Example, A$B    Data of memory address A substitute into variable B.
%      Example, 16383$B   Data of memory address 3FFFh substitute into variable B upper 8bit,
%                           and data of memory address 4000h substitute into variable B lower 8bit
%
% 16. Text statement
%      Text statement is available in direct execution and in a program.
%      If there is a ‘"’ in a line, the rest of the line is recognized as text.
%      Example,  10: “START     ‘START’ output as a text.
%      Example,  20: “END;    ‘END’ will not be output as a text, but as a comment sentence.
%
%-------------------------------------------------------------------------------------------
% End of appendix
```