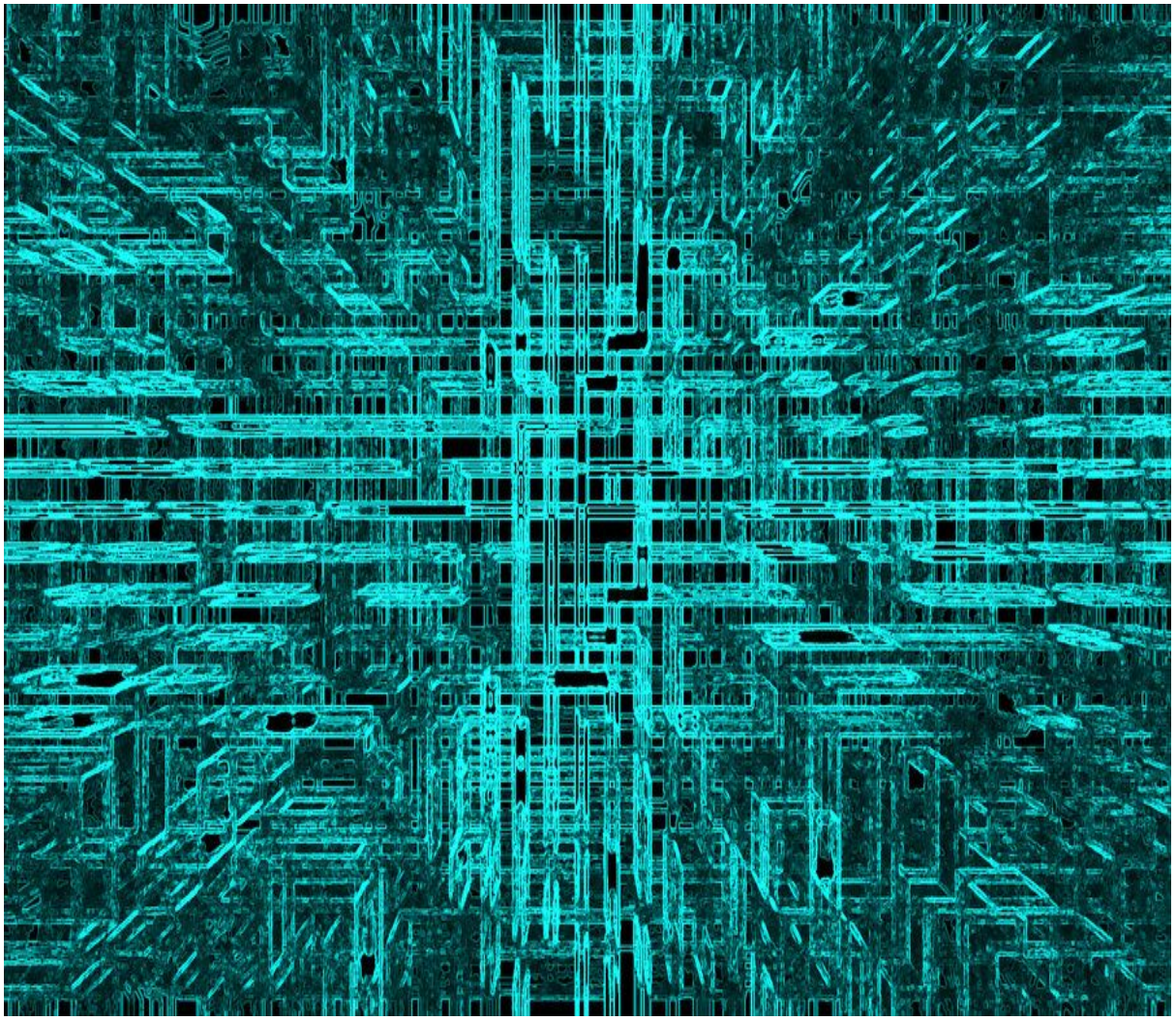
Systems Programming

Final Project

SIC/XE Assembler(1)



Name: Basem Mohamed Gaber ID: 4826
Name: Ziad Sherif El-Saeed ID: 4640
Name: Ahmed Medhat Mohamed ID: 4612
Name: Ahmed Mos'ad Mahmoud ID: 4604
Name: Mahmoud Fouad Desouki ID: 4656

-Requirements Specification:

building a parser that deals with source lines that are instructions, storage declaration, comments, and assembler directives.
the parser should be able to decode 2,3 and 4-byte instructions as follows:

- 2 bytes with 1,2 symbolic register reference.
- RSUB (it ignores any operand)
- 3 byte PC -relative with symbolic operand to include immediate, indirect and indexed addressing.
- 3 byte absolute with non-symbolic operand to include immediate, indirect and indexed addressing.
- 4 byte absolute with symbolic or non-symbolic operand to include immediate, indirect and indexed addressing.

The parser should handle ALL storage directives.
The output of this phase should contain:

- The symbol table.
- The source program in a correct format.
- a meaningful error message to be printed below the line in which the error occurred.

-Design:

The source code is divided into 3 classes :

a) Main
b) Pass1
c) Opcodes

*The main class contains the initiation of the data structures and controls the reading and writing of source codes and list files respectively.
*The Pass1 class contains the processing of the assembly code itself including the control of the instructions to produce an output. All error handling take place in the Pass1 class as well.

*The Opcodes class contains the optbl hashmap that holds the opcodes resembling each and every instruction in SIC/XE alongside the frmttbl hashmap which holds the possible format/s for these instructions.

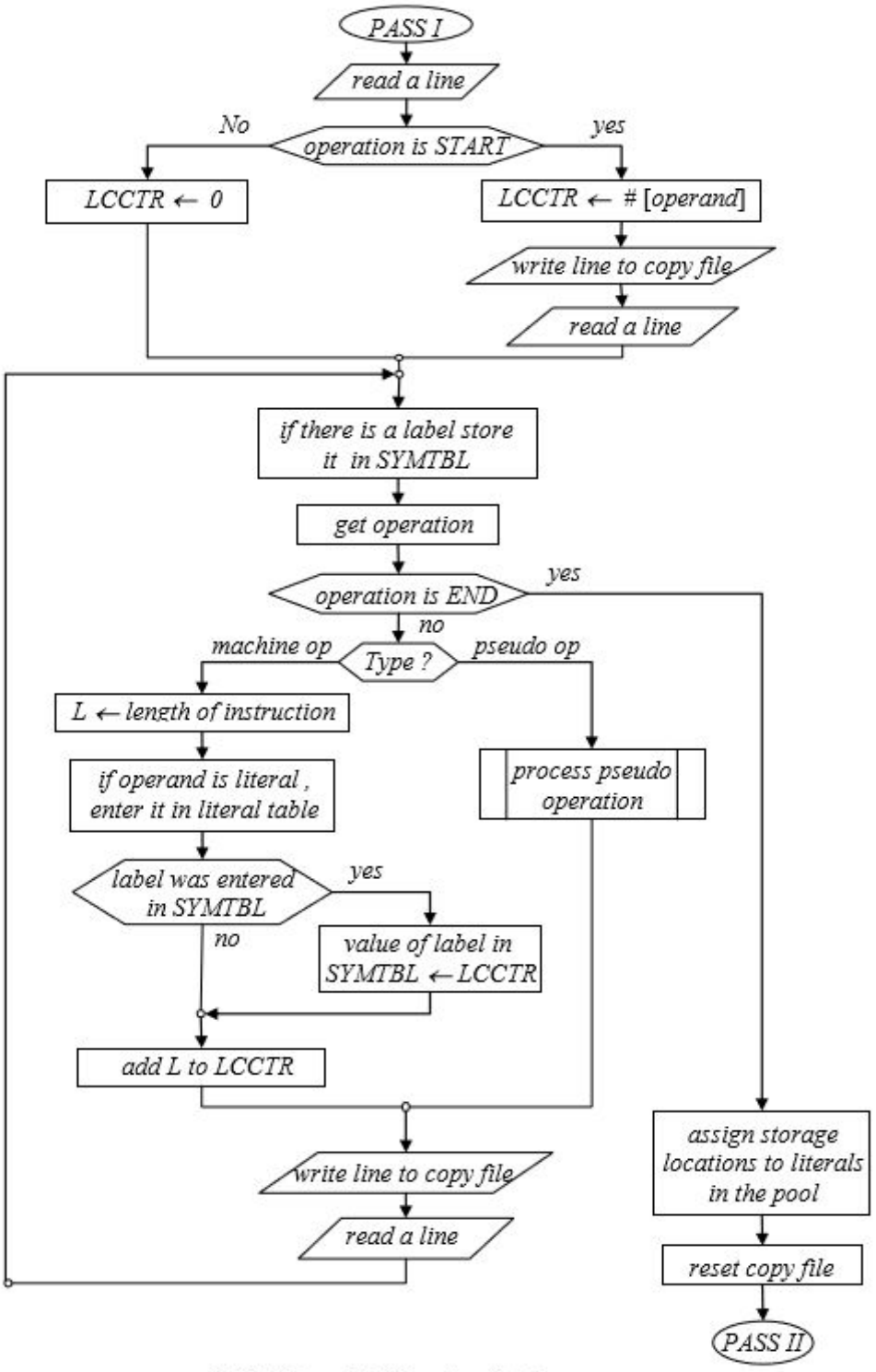
The program starts from the main class which then passes control to the Pass1 class which processes the assembly code line by line and keeps track of the LOCCTR and notifies on the instance of occurence of any errors.

The Pass1 class outputs using a Buffered Writer into a txt file that contains the location of every instruction, specific indication of errors, and a sorted dump of the symbol table. The output of phase 1 is the list file that is to be used as input for phase 2.

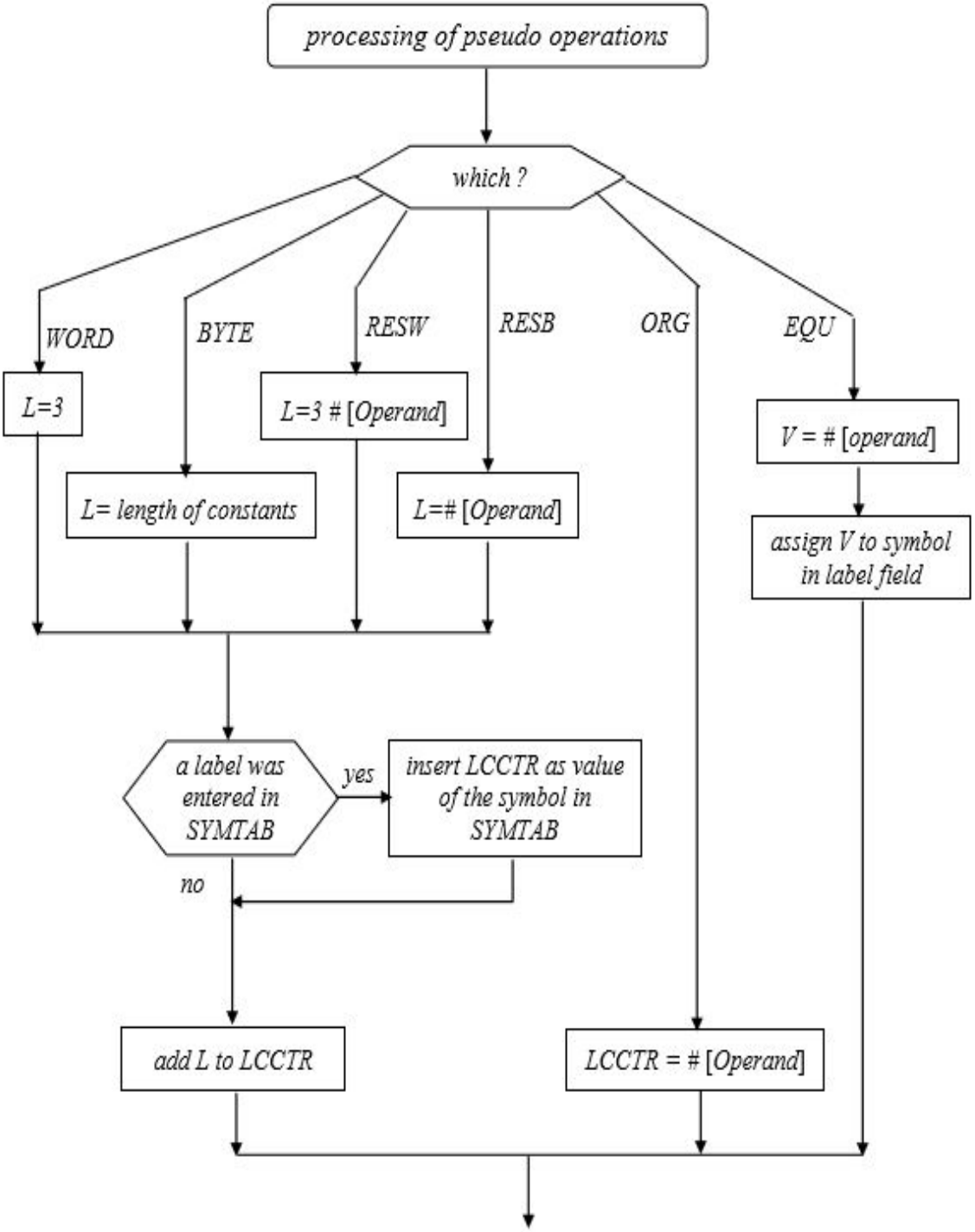
-Main data structures:

- Arrays
- Hashing

-Algorithms description:



PASS I : Define Symbols



Pass I: Continued

Pass 1:

```
begin
    initialize SYMTAB
    read input line
    if opcode = 'START'
        then begin
                starting_address = #[operand]
                LOCCTR = starting_address
                write line to copy file
                read next line
             end
        else LOCCTR = 0

    while opcode ≠ 'END' do
      begin
        if line is an instruction then                    // processing of instruction
          begin if there is a symbol in label field then
                      insert [symbol, LOCCTR] into SYMTAB
                L = length of instruction
                LOCCTR = LOCCTR + L
                if there is a literal in operand field then insert literal into LITTAB
          end
        else                                              // processing of directives
          if      opcode = 'ORG' then LOCCTR = #[operand]
          elseif  opcode = 'EQU' then
                  begin V = #[operand]
                        insert [symbol, V] into SYMTAB
                  end
          else
            begin
              if there is a symbol in label field then
                        store [symbol, LOCCTR] in SYMTBL
              if      opcode= 'WORD' then L = 3
              elseif  opcode = 'BYTE' then L = length of constant in bytes
              elseif  opcode = 'RESW' then L = 3 * #[operand]
              elseif  opcode = 'RESB' then L = #[operand]
              LOCCTR = LOCCTR + L
            end
        write line to copy file
        read next line
      end while

    assign storage to literals in the pool , if any
    reset copy file
    program length = LOCCTR – starting address
end
```

-Sample run:

By reading ‘a_example’ and decoding it.

‘a_example’:

```
            Start   0
prog        LDX     t#0
prog        LDT      #1
            tLDA      #0
            STA      CURRENT
PASS         LDCH    STRING,X
            RMO       A,S
            LDA      #STRINGgggggggggggggg
            ADDR     X,A
            STA      P1
            ADDR     T,X
            LDCH     STRING,X
            COMP     EOF2
            JEQ      DONE
            COMPR    A,S
            JLT      LOOP
            J        PASS
LOOP        LDA      #STRING
            ADDR     X,A
            STA      P2
            JSUB     SWAP
            J        PASS
DONE        SUBR     T,X
            STX      TURNS
            LDX      CURRENT
            TIX      TURNS
            STX      CURRENT
            LDX      #0
            JLT      PASS
            J        *
.
SWAP        LDCH     @P1
            STCH     TEMP
            LDCH     @P2
            STCH     @P1
            LDCH     TEMP
            STCH     @P2
            RSUB
.
read        td       indev
            jeq      read
            rd       indev
            rsub
.
WRITE       TD       OUTDEV
            JEQ      WRITE
            WD       OUTDEV
            RSUB
.
P1          RESW     1
P2          RESW     1
TEMP        RESB     1
I           RESB     1
J           RESB     1
STRING      BYTE     C'53198247*'
EOF         WORD     #42
TURNS       RESW     1
CURRENT     RESW     1
indev       byte     x'F3'
OUTDEV      BYTE     X'05'
```

and writing the output in the file ‘b_example’:

```
000000           Start   0
000000 prog      LDX     t#0
error [19] : '16th and 17th characters of instruction must be blank, operation ends at 16, operation starts at 19 '
000003 prog      LDT     #1
error [04] : 'duplicate label definition:'  prog 'is already defined'
000006           tLDA     #0
error [18] : 'operand field can not contain spaces in the middle'
error [21] : '1st character of operand can not be blank'
error [08] : 'Unrecognized operation code'
000009           STA     CURRENT
00000C PASS       LDCH   STRING,X
error [20] : '1st character of operation can not be blank'
00000F           RMO      A,S
error [18] : 'operand field can not contain spaces in the middle'
error [21] : '1st character of operand can not be blank'
000011           LDA     #STRINGgggggggggggggg
error [22] : 'operand must end at 35th character'
error [10] : 'Immediate operand is not a number'
000014           ADDR    X,A
000016           STA     P1
000019           ADDR    T,X
00001B           LDCH    STRING,X
00001E           COMP    EOF2
000021           JEQ     DONE
000024           COMPR   A,S
000026           JLT     LOOP
000029           J       PASS
00002C LOOP      LDA     #STRING
error [10] : 'Immediate operand is not a number'
00002F           ADDR    X,A
000031           STA     P2
000034           JSUB    SWAP
000037           J       PASS
00003A DONE      SUBR    T,X
00003C           STX     TURNS
00003F           LDX     CURRENT
000042           TIX     TURNS
000045           STX     CURRENT
000048           LDX     #0
00004B           JLT     PASS
00004E           J       *
      .
000051 SWAP      LDCH    @P1
000054           STCH    TEMP
000057           LDCH    @P2
00005A           STCH    @P1
00005D           LDCH    TEMP
000060           STCH    @P2
000063           RSUB
      .
000066 read      td      indev
000069           jeq     read
00006C           rd      indev
00006F           rsub
error [21] : '1st character of operand can not be blank'
error [06] : 'RSUB operation can not have an operand'
      .
000072 WRITE     TD      OUTDEV
000075           JEQ     WRITE
000078           WD      OUTDEV
00007B           RSUB
      .
00007E P1        RESW    1
000081 P2        RESW    1
000084 TEMP      RESB    1
000085 I         RESB    1
000086 J         RESB    1
error [30] : 'Using mnemonics as labels is not allowed'
000087 STRING    BYTE    C'53198247*'
000090 EOF       WORD    #42
000093 TURNS     RESW    1
000096 CURRENT   RESW    1
000099 indev     byte    x'F3'
00009A OUTDEV    BYTE    X'05'
error [13] : 'missing END statement' '

End of first pass
value          name
------------------
  0         prog
 12         pass
 44         loop
 58         done
 81         swap
102         read
114         write
126         p1
129         p2
132         temp
133         i
135         string
144         eof
147         turns
150         current
153         indev
154         outdev

Incomplete Assembly
```