SIC/XE Assembler

Phase 1

—

Ahmad Abdallah Waheeb

Ahmed Mohamed EL Zeny

Abdelrahman Kamal

Abdelfattah Mohamed

Requirements Specification

Implementing SIC/XE assembler, written in C++, producing code for the absolute loader used in the SIC/XE programming assignments. Implementing the only Pass1 of the assembler. The output of this phase would be used as input for the subsequent phase.

Data Structures:

struct arr:

An object that represents the operations supported by the assembler with 3 properties:

the opCode for the operation code,

length for the operation length and

rgx for the regular expression used to detect this particular operation.

opt_table:

An unordered map that holds all the operations supported by the assembler.

directives:

An unordered map that holds all the directives supported by the assembler.

symtab:

An unordered map that stores all the labels detected by the assembler and their addresses.

Design

- We created multiple functions as each one handles a specific part of the program.
- Support free-formatted assembly language programs. In a free-formatted assembly program, statements are not restricted to begin at a given position in the line. Many consecutive white spaces or tabs should be treated as a single space.

```
void pass_1(string file);
```

/**

- Params: file path.

*/

- It's used to execute logic of pass 1.

```
void print_line_at_file(string line);
```

/**

- Params: read line.

*/

- It's used to print error and warning at output file.

```
void print_sym_table(string label, int pcCounter);
```

/**

- Params: label , pcCounter

*/

- It’s used to print Symbols table at file.

```
void sym_table();
```

/**

*/

- Get label and address from symtab to print it at file.

```
void print_format(string line, int pcCounter,int no_of_line);
```

/**

- Params: label, pcCounter, no_of_line.

*/

- Print read line at output file with a format.

```
bool validate_arr(string operation, string line);
```

/**

- Params: operation, line.

*/

- To validate passed operation.

`bool validate_dir(string operation, string line);`

/**

- Params: operation, line.

*/

- To validate passed directive.

`bool comment_line(string line);`

/**

- Params: line.

*/

- Check if this line is a comment or not.

`bool has_plus(string line);`

/**

- Parsms: line.

*/

- To check if operation format 3 or 4.

`string get_operand(string line);`

/**

- Params: line.

*/

- To get operand from line.

`string has_label(string line);`

/**

- Params: line.

*/

- To get label from line.

`string get_operation(string line);`

/**

- Params: line.

*/

- To get operation from line.

`string get_comment(string line);`

/**

- Params: line.

*/

- To get comment from line.

`string trim(const string &str);`

/**

- Params: &str.

*/

- To get any string without spaces.

```
int getInstructionLength(string operation, string line);
```

/**

- Params: operation, line.

*/

- To get instruction length.

```
int length_of_ins(string operation, string line);
```

/**

- Params: operation, line.

*/

- To get the length of directive and zero if it’s not defined.

```
bool comment_line(string line);
```

/**

- Params: operation, line.

*/

- returns true if the line given is a comment false otherwise.

```
bool start_line(string line);
```

/**

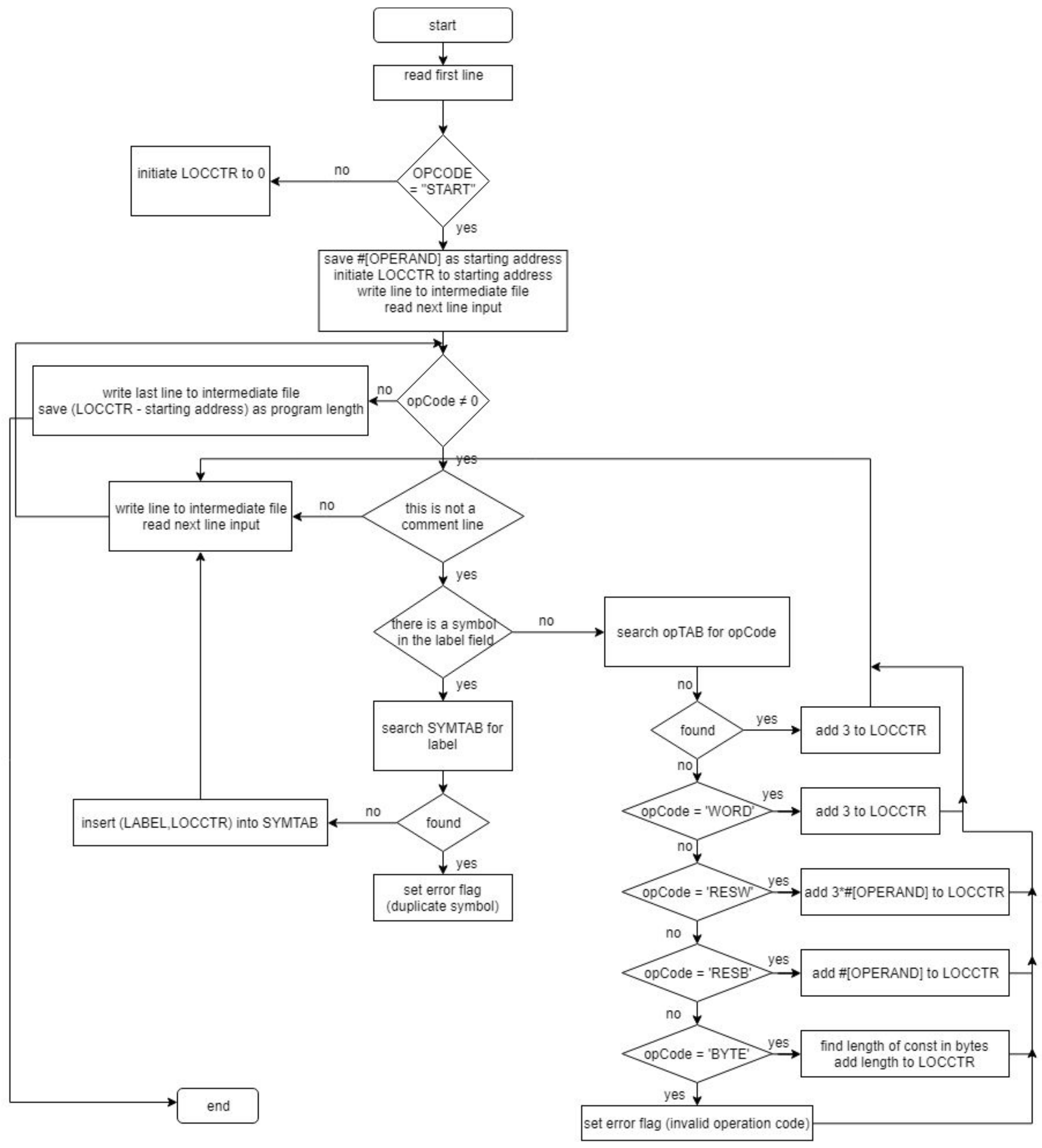
- Params: operation, line.

*/

- returns true if the line has given is a start line, false otherwise.

Algorithms Description

Here is a flowchart of the algorithm we used

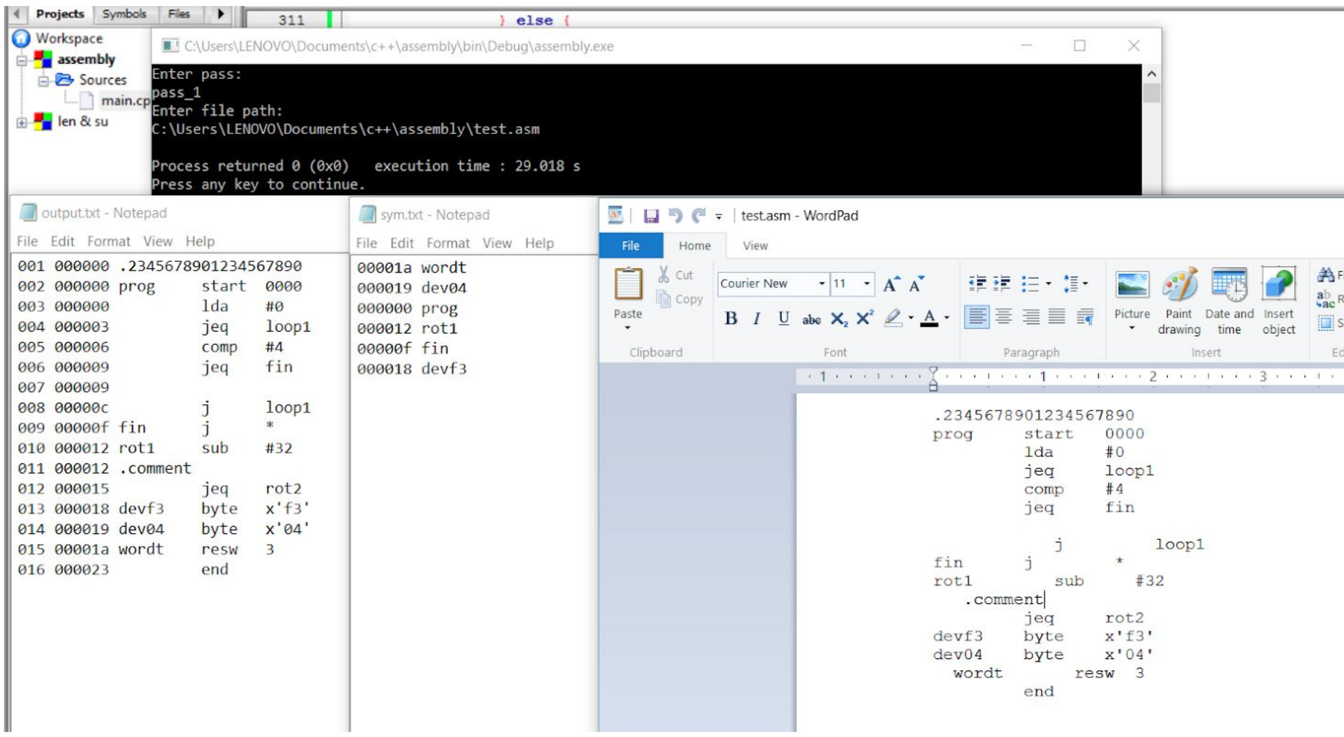

start
read first line
OPCODE = "START"
no
initiate LOCCTR to 0
yes
save #[OPERAND] as starting address
initiate LOCCTR to starting address
write line to intermediate file
read next line input
opCode ≠ 0
no
write last line to intermediate file
save (LOCCTR - starting address) as program length
yes
this is not a comment line
no
write line to intermediate file
read next line input
yes
there is a symbol in the label field
no
search opTAB for opCode
yes
search SYMTAB for label
found
no
insert (LABEL,LOCCTR) into SYMTAB
yes
set error flag (duplicate symbol)
found
yes
add 3 to LOCCTR
opCode = 'WORD'
add 3 to LOCCTR
opCode = 'RESW'
add 3*#[OPERAND] to LOCCTR
opCode = 'RESB'
add #[OPERAND] to LOCCTR
opCode = 'BYTE'
find length of const in bytes
add length to LOCCTR
set error flag (invalid operation code)
end

Assumptions

We assumed that any comment must be preceded by a dot.

The start operand has to be decimal.

Sample Runs:



```
Enter pass:
pass_1
Enter file path:
C:\Users\LENOVO\Documents\c++\assembly\test.asm

Process returned 0 (0x0)   execution time : 29.018 s
Press any key to continue.
```

output.txt:

```
001 000000 .2345678901234567890
002 000000 prog     start   0000
003 000000          lda     #0
004 000003          jeq     loop1
005 000006          comp    #4
006 000009          jeq     fin
007 000009
008 00000c          j       loop1
009 00000f fin      j       *
010 000012 rot1     sub     #32
011 000012 .comment
012 000015          jeq     rot2
013 000018 devf3    byte    x'f3'
014 000019 dev04    byte    x'04'
015 00001a wordt    resw    3
016 000023          end
```

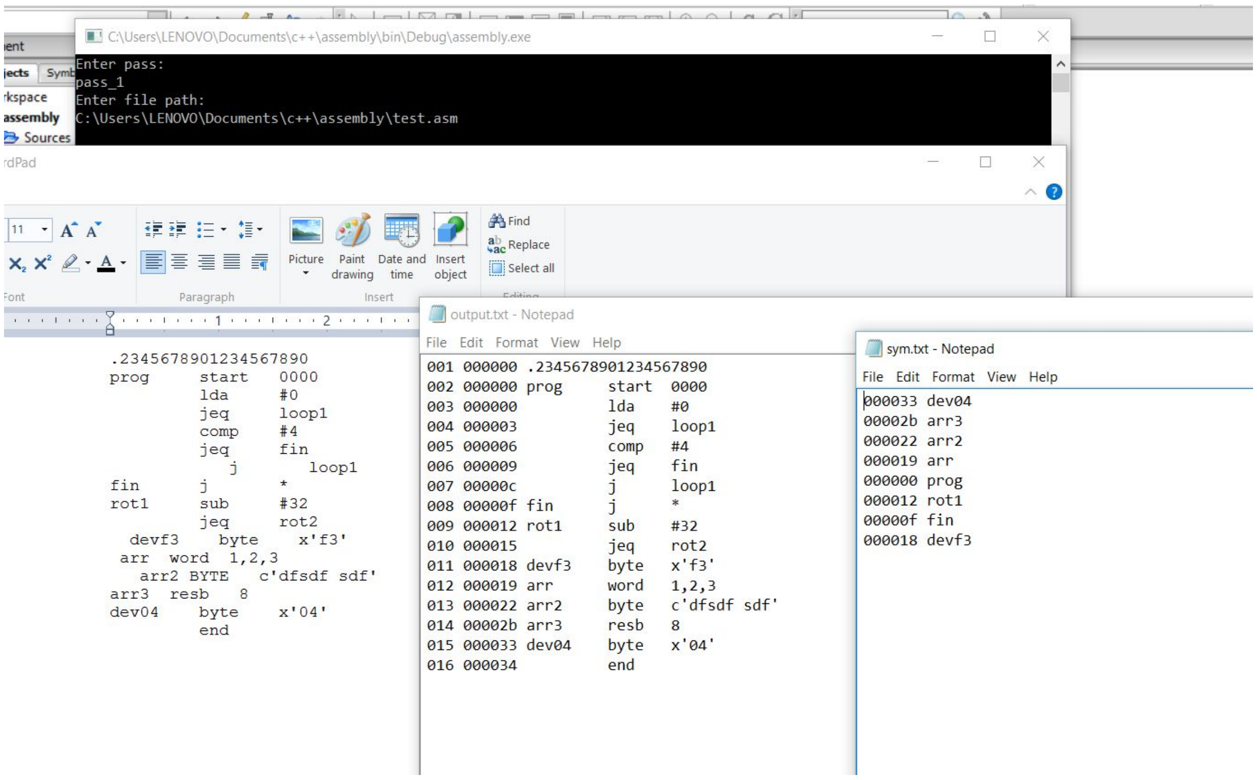
sym.txt:

```
00001a wordt
000019 dev04
000000 prog
000012 rot1
00000f fin
000018 devf3
```

test.asm:

```
.2345678901234567890
prog      start    0000
          lda      #0
          jeq      loop1
          comp     #4
          jeq      fin

             j          loop1
fin       j         *
rot1         sub      #32
   .comment
          jeq      rot2
devf3     byte     x'f3'
dev04     byte     x'04'
  wordt        resw   3
          end
```



```
Enter pass:
pass_1
Enter file path:
C:\Users\LENOVO\Documents\c++\assembly\test.asm
```

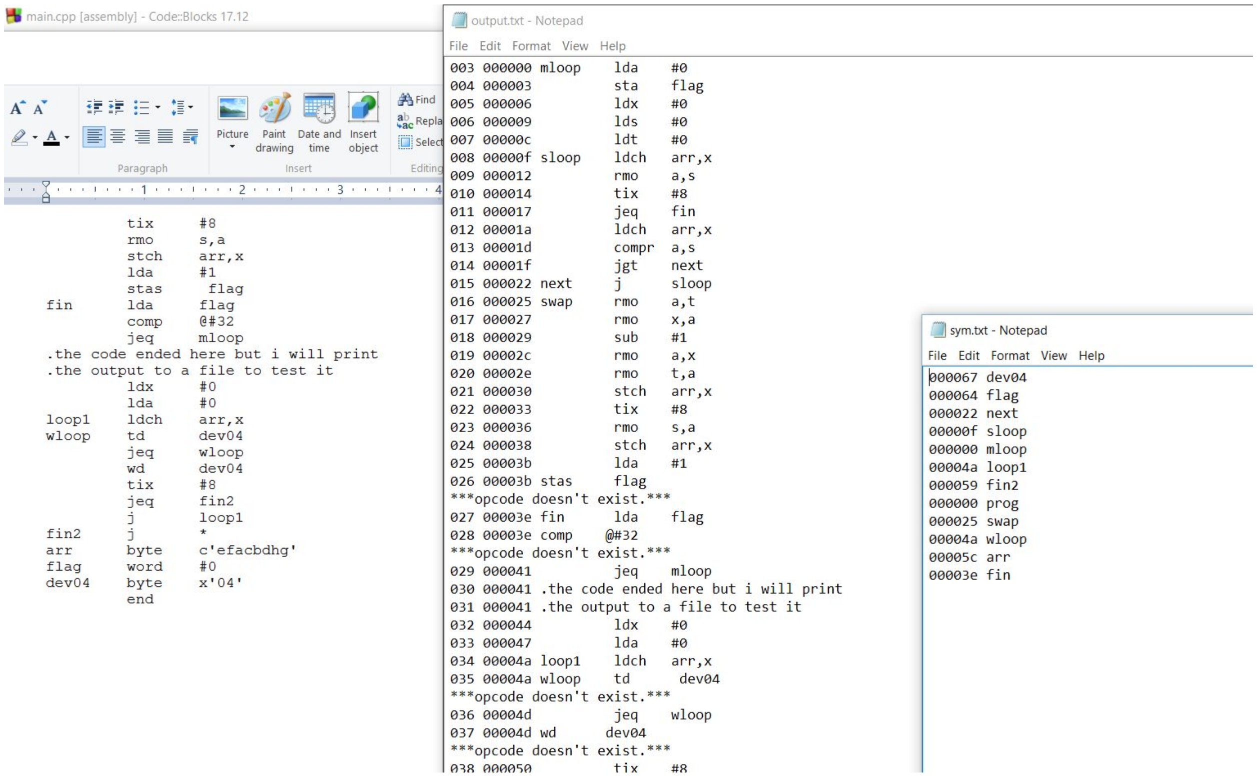
```
.23456789012345678901234567890
prog        start   0000
            lda     #0
            jeq     loop1
            comp    #4
            jeq     fin
                j       loop1
fin         j       *
rot1        sub     #32
            jeq     rot2
  devf3       byte     x'f3'
 arr  word   1,2,3
   arr2 BYTE    c'dfsdf sdf'
arr3   resb     8
dev04       byte    x'04'
            end
```

output.txt - Notepad

```
001 000000 .2345678901234567890
002 000000 prog     start  0000
003 000000          lda    #0
004 000003          jeq    loop1
005 000006          comp   #4
006 000009          jeq    fin
007 00000c          j      loop1
008 00000f fin      j      *
009 000012 rot1     sub    #32
010 000015          jeq    rot2
011 000018 devf3    byte   x'f3'
012 000019 arr      word   1,2,3
013 000022 arr2     byte   c'dfsdf sdf'
014 00002b arr3     resb   8
015 000033 dev04    byte   x'04'
016 000034          end
```

sym.txt - Notepad

```
000033 dev04
00002b arr3
000022 arr2
000019 arr
000000 prog
000012 rot1
00000f fin
000018 devf3
```



```
            tix     #8
            rmo     s,a
            stch    arr,x
            lda     #1
            stas     flag
fin         lda     flag
            comp    @#32
            jeq     mloop
.the code ended here but i will print
.the output to a file to test it
            ldx     #0
            lda     #0
loop1       ldch    arr,x
wloop       td      dev04
            jeq     wloop
            wd      dev04
            tix     #8
            jeq     fin2
            j       loop1
fin2        j       *
arr         byte    c'efacbdhg'
flag        word    #0
dev04       byte    x'04'
            end
```

output.txt - Notepad

```
003 000000 mloop    lda    #0
004 000003          sta    flag
005 000006          ldx    #0
006 000009          lds    #0
007 00000c          ldt    #0
008 00000f sloop    ldch   arr,x
009 000012          rmo    a,s
010 000014          tix    #8
011 000017          jeq    fin
012 00001a          ldch   arr,x
013 00001d          compr  a,s
014 00001f          jgt    next
015 000022 next     j      sloop
016 000025 swap     rmo    a,t
017 000027          rmo    x,a
018 000029          sub    #1
019 00002c          rmo    a,x
020 00002e          rmo    t,a
021 000030          stch   arr,x
022 000033          tix    #8
023 000036          rmo    s,a
024 000038          stch   arr,x
025 00003b          lda    #1
026 00003b stas     flag
***opcode doesn't exist.***
027 00003e fin      lda    flag
028 00003e comp    @#32
***opcode doesn't exist.***
029 000041          jeq    mloop
030 000041 .the code ended here but i will print
031 000041 .the output to a file to test it
032 000044          ldx    #0
033 000047          lda    #0
034 00004a loop1    ldch   arr,x
035 00004a wloop    td      dev04
***opcode doesn't exist.***
036 00004d          jeq    wloop
037 00004d wd      dev04
***opcode doesn't exist.***
038 000050          tix    #8
```

sym.txt - Notepad

```
000067 dev04
000064 flag
000022 next
00000f sloop
000000 mloop
00004a loop1
000059 fin2
000000 prog
000025 swap
00004a wloop
00005c arr
00003e fin
```