System Programming

ASSEMBLER(phase 1)

Mohammed kamal abd elrhman 59
Mohamed El Maghraby 55
Ahmed Hesham 10
Rita samir 26

Introduction

Implementing the pass1 of the SIC/XE assembler in C++

Specifications:

1. Parsing SIC/XE code in formatts 2,3 & 4.
2. Handling directives BYTE,WORD,RESW,RESB,ORG & EQU.
3. Generate the symbol table.
4. Generate the source file of code with assigned addresses.

Implementation details:

- **Design :**
  The main design consist of 4 modules parsing ,validation, executing the pass1 algorithm & printing the output.
  - The parsing module extract the components of each line.
  - The validating module validate the components of every line.
  - Then executing the algorithm and printing the output.
- **Main data structures** :
  - Arrays & Lists : contain the components of every line sequenchelly.
  - Maps : made one to one mapping between the operation code & their hexadecimal values & formats
  - Created data structures LIne & Rows : carry the values of the line components.

- **Algorithm :**

  As it mentioned in the lectures.

Pass 1:

Algorithm Pass 1

```
begin
   read first input line
   if OPCODE = 'START' then
       begin
           save #[OPERAND] as starting address
           initialize LOCCTR to starting address
           write line to intermediate file
           read next input line
       end {if START}
   else
       initialize LOCCTR to 0
   while OPCODE ≠ 'END' do
       begin
           if this is not a comment line then
               begin
                   if there is a symbol in the LABEL field then
                       begin
                           search SYMTAB for LABEL
                           if found then
                               set error flag (duplicate symbol)
                           else
                               insert (LABEL,LOCCTR) into SYMTAB
                       end {if symbol}
                   search OPTAB for OPCODE
                   if found then
                       add 3 {instruction length} to LOCCTR
                   else if OPCODE = 'WORD' then
                       add 3 to LOCCTR
                   else if OPCODE = 'RESW' then
                       add 3 * #[OPERAND] to LOCCTR
                   else if OPCODE = 'RESB' then
                       add #[OPERAND] to LOCCTR
                   else if OPCODE = 'BYTE' then
                       begin
                           find length of constant in bytes
                           add length to LOCCTR
                       end {if BYTE}
                   else
                       set error flag (invalid operation code)
               end {if not a comment}
           write line to intermediate file
           read next input line
       end {while not END}
   write last line to intermediate file
   save (LOCCTR - starting address) as program length
end {Pass 1}
```

- **Assumptions :**
  - Add comments in free format preceded with “.”
  - Don’t enter literals & expressions.

Samples Run :

```
line    Address     Label    Opcode  Operands   Comment
0       1000        prbn08   start   1000
1       1000                 lda     zero
2       1003                 sta     index
3       1006        loop     lda     zero
4       1009                 ldx     index
5       100c                 sta     alpha,x
6       100f                 lda     index
7       1012                 add     three
8       1015                 sta     index
9       1018                 comp    k300
10      101b                 jlt     loop
11      101e        alpha    resw    100
12      114a        k300     word    300
13      114d        zero     word    0
14      1150        three    word    3
15      1153        index    resw    1
16      1156                 end     prbn08
```

```
---------------------------------
|          symbol Table         |
---------------------------------
|Label            |   Address   |
---------------------------------
|alpha            |      101e   |
---------------------------------
|index            |      1153   |
---------------------------------
|k300             |      114a   |
---------------------------------
|loop             |      1006   |
---------------------------------
|prbn08           |      1000   |
---------------------------------
|three            |      1150   |
---------------------------------
```

sample 2

```
line   Address     Label    Opcode  Operands   Comment
0      1000        prog     start   1000
1      1000        loop     addr    s,p
          ****Error:syntax error  this should be two registers in the way r1,r2
2      1000        add      add     add
          ****Error:syntax error  opcode in false position
3      1000        loop     addr
          ****Error:syntax error  un correct way to type operand in that position
4      1000        three    equ     3
5      1004        +ed      sta     length
          ****Error:un correct format
6      1004                 sub     max
7      1007        lop      byte    c'pdo'
8                  cloop    org     1068
9      1068        length   resb    256
10     1168        max      resw    3
11     1171                 end


---------------------------------
|          symbol Table         |
---------------------------------
|Label            |   Address   |
---------------------------------
|cloop            |      1068   |
---------------------------------
|length           |      1068   |
---------------------------------
|lop              |      1007   |
---------------------------------
|max              |      1168   |
---------------------------------
|prog             |      1000   |
---------------------------------
|three            |         3   |
---------------------------------
```

Sample 3:

```
line    Address      Label     Opcode   Operands    Comment
0       1000         prob1     start    1000
1       1000                   ldx      #0
2       1003                   ldt      #48
3       1006                   lds      #2
4       1009         loop      ldch     beta,x
5       100c                   subr     t,a
6       100e                   tixr     s
7       1010                   jlt      loop
8       1013         beta      byte     c'36'
9       1015                   end      prob1


---------------------------------
|           symbol Table        |
---------------------------------
|Label            |   Address   |
---------------------------------
|beta             |      1013   |
---------------------------------
|loop             |      1009   |
---------------------------------
|prob1            |      1000   |
---------------------------------
```