SYSTEM PROGRAMMING

Final Term Project – SIC/XE Assembler

Phase 2

Team Members:

1 - Mohamed Alkazzafy Ibrahim	– 37
2 - Mohamed Samir Zaitoon	– 40
3 - Mohamed Hesham Alkhaligy	– 45
4 – Mostafa Mohamed Fares	– 48

MAY 14, 2019

The term project is to implement SIC/XE assembler, written in C/C++, producing code for the absolute loader used in the SIC/XE programming assignments.

In phase 2 of the project, you are going to build on the previous phase and use its output to implement pass 2 of the assembler.

Requirement specification:

1. The assembler is to execute by entering: assemble <source-file-name>
2. The source file for the main program for this phase is to be named

   assemble.cpp
3. The output of the assembler should include (at least):

   a) Object-code file whose format is the same as the one described in the text book in section 2.1.1 and 2.3.5.

   b) A report at the end of pass2. Pass1 and Pass2 errors should be included as part of the assembler report, exhibiting both the erroneous lines of source code and the error.
4. The assembler should support:

   a) EQU and ORG statements.

   b) Simple expression evaluation. A simple expression includes simple (A <op> B) operand arithmetic, where <op> is one of +, -, *, / and no spaces surround the operation, eg. A+B.

Design:

C++ is used in the implementation of the phase-2-project in which several classes are created and utilized in the program (brief summary):

1. Class “pass1”: which holds the main pass-1 algorithm
2. Class “statement”: which contains a given instruction in its respective components (Label-”Optional”, Operation Code, Operand-“Optional”, Comment-“Optional”)
3. Class “parser”: which parses a given string into an instruction if valid and gives error in the case of error detection.
4. Module “Common”: which contains the common tables between pass-1 and pass-2 like “SYMTAB, OPTAB, LITTAB and so on” , common class like “info, symbol, Literal” and common functions
5. Module “pass2” : which holds the main pass-2 algorithm

Main Data Structures:

Appendix:

-info: is a class that contains format type, operation code & operands respectively.

-symbol: is a class holds information of a symbol appear in code

Like address, length, typeAddress (relocatable or absolute) and type (word or byte)

-Literal: is a class holds information of a literal appear in code

Like literal, length, address, value

-X: is a class is a class holds information of a statement like address of of a statement, statement, obcode

Map: a) optab: is used to store info of the mnemonics of a certain assembler hence, is static.

b) directivetab: is used to store the info of the directives of a certain assembler hence, is static.

c) register_tab: is used to store the values of the registers.

d)symtab: is used to store the symbols in the code

Vectors:

a)littab: is used to store literals in the code

b)intermediate: used to store statement of each line in code with its address

Algorithms description:

Pass-2 Algorithm:

- After pass-1 finished, the intermediate block is ready for pass-2
- Firstly ,Pass-2 check if there is an error occurred in pass-1 ,then do nothing and return if true
- If there is no error, get first statement from intermediate
- If OPCODE == “START” then write listing line
- Write header record to object program
- Initialize first text record
- For i = 1 to intermediate. Size do the following:
    - Item <= intermediate[i]
    - If item. OPCODE == ‘END’ then break
    - pc = intermediate[i+1]. address
    - Search OPTAB for OPCODE
    - If found then calculate obcode of this statement
    - Else if OPCODE ==’BYTE’ or ‘WORD’ then convert constant to object code
    - Else if OPCODE == “BASE” then
        - Check if its operand is in symtab
        - enabelBase = true
        - base register = address of this operand
        - else set error
    - else if OPCODE ==” NOBASE”
        - enableBase = false
    - if this statement has obcode then
        - obcode will not fit into current text record or address of this statement is not is the sequence that beginning from the start address of this text record
            - write text record to object program
            - initialize new text record
        - add object code to text record
    - write listing line and error if exist
- write last text record to object program
- write end record to object program
- write last listing line and error if exist
- end pass-2

calculate object code:

-parameter: (X item, string error, string op)

-set object code to item.obcode if there is no error and 0 otherwise

Algorithm:

- get operation code of mnemonic instruction and set it to opcode
- if mnemonic is format 2 then
    - get number of each register from register table
    - if the registers found then assign obcode
    - else set error
- else if format 3 and no. operand >= 1(of a mnemonic)
    - calculate address of operand for different cases and determine that flags (n, I, x)
    - set error if exists
    - if format 4, then make displacement equals address and set flags (b, p, e) to its value
    - else if indirect or immediate, then displacement equals address and set flags (b, p, e) to its value
    - else if displacement has already calculated, then set flags (b, p, e) to its value
    - else calculate relative address and set error if the displacement is an `overflow value`
- else if format 3 and no. operand == 0 (of a mnemonic) then set set flags (n, I, x, b, p, e) to its value and displacement will be equal 0
- if format == 3 then
    - assign obcode
- end of method

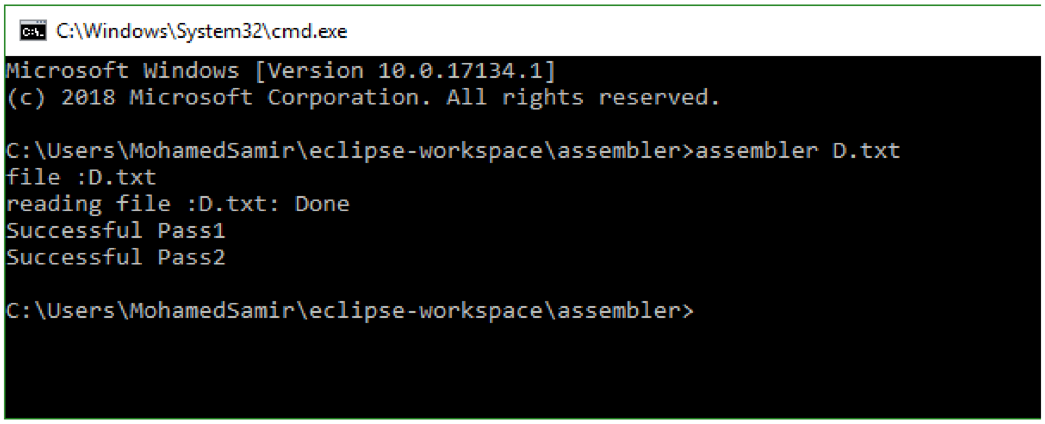
Assumptions:

Component: is either a label, operation code, operand or comment.

- Every component must be separated by a white space.
- Every comment should start with a dot character ".".
- Literals (Including LTORG) are available

Sample Runs:

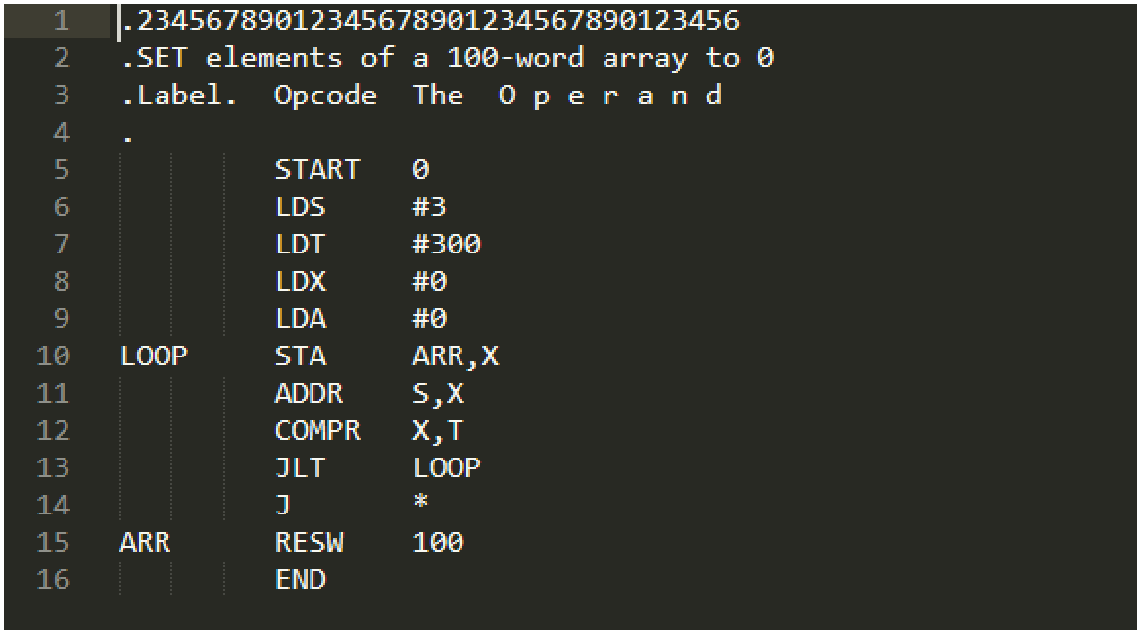
1:



C:\Windows\System32\cmd.exe

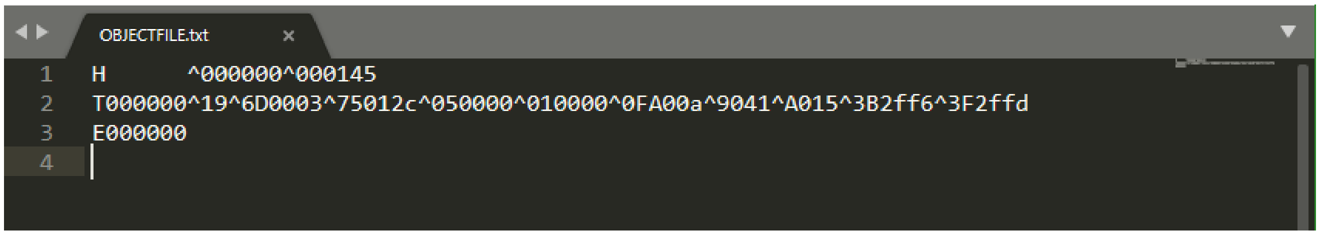
```
Microsoft Windows [Version 10.0.17134.1]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MohamedSamir\eclipse-workspace\assembler>assembler D.txt
file :D.txt
reading file :D.txt: Done
Successful Pass1
Successful Pass2

C:\Users\MohamedSamir\eclipse-workspace\assembler>
```

D.txt:



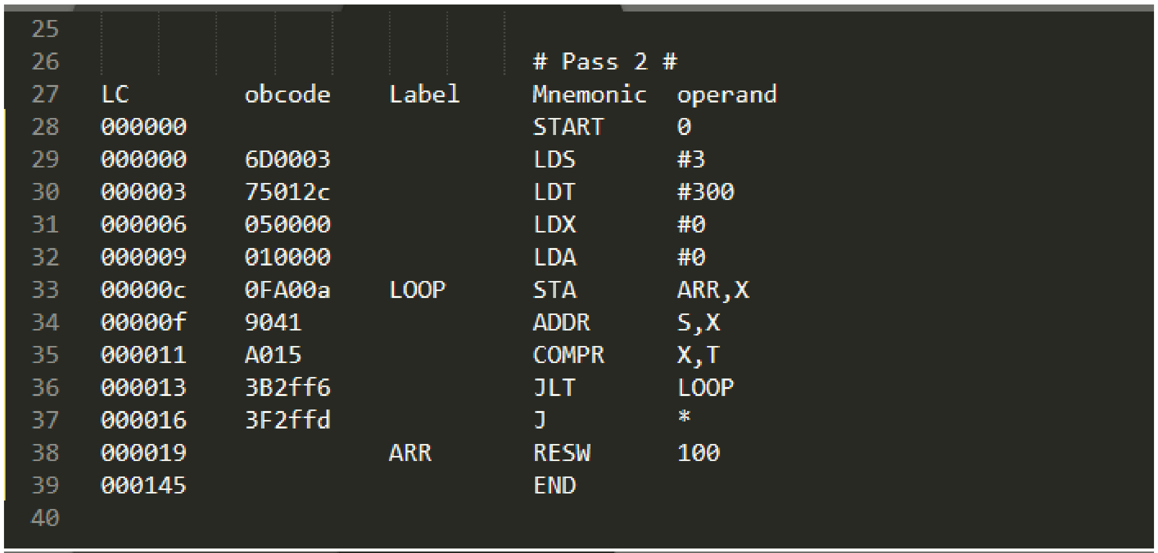
```
.23456789012345678901234567890123456
.SET elements of a 100-word array to 0
.Label.  Opcode  The  O p e r a n d
.
                 START    0
                 LDS      #3
                 LDT      #300
                 LDX      #0
                 LDA      #0
LOOP             STA      ARR,X
                 ADDR     S,X
                 COMPR    X,T
                 JLT      LOOP
                 J        *
ARR              RESW     100
                 END
```

OBJECTFILE.txt

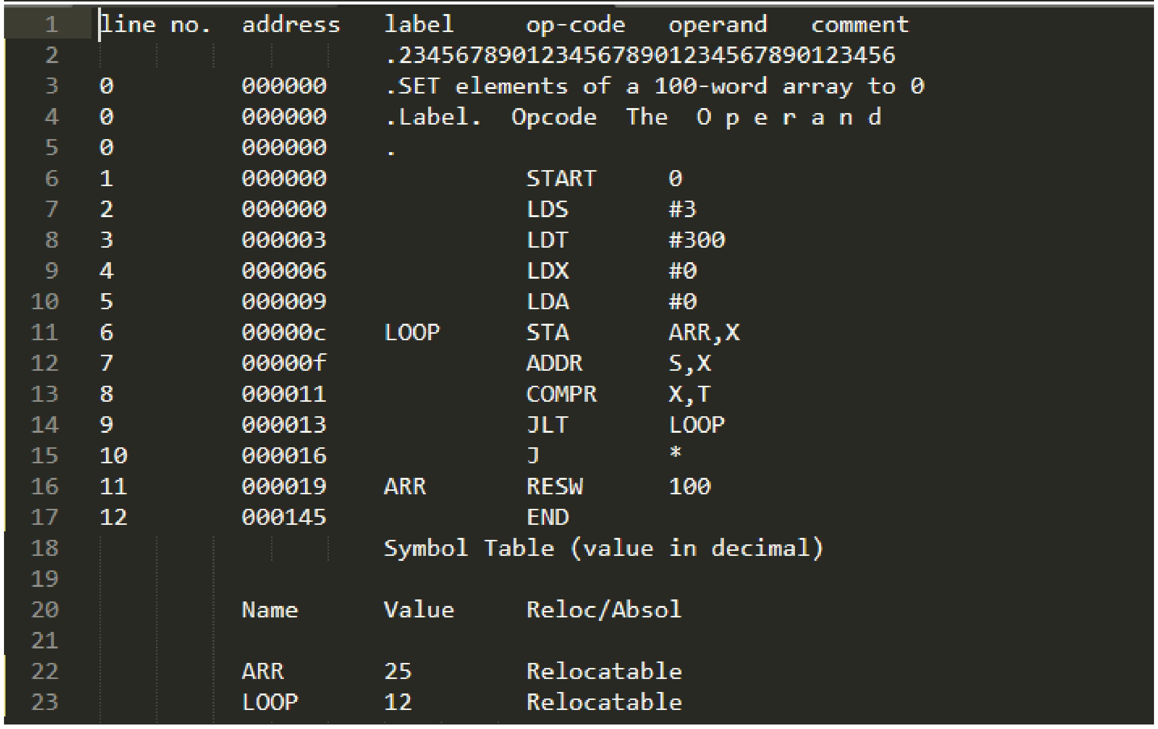


```
H      ^000000^000145
T000000^19^6D0003^75012c^050000^010000^0FA00a^9041^A015^3B2ff6^3F2ffd
E000000
```

LISTFILE.txt:

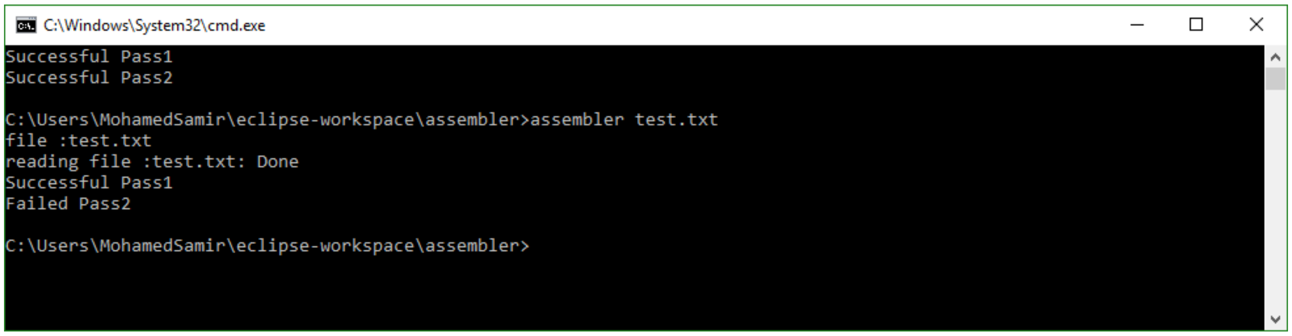


```

                              # Pass 2 #
LC        obcode    Label     Mnemonic  operand
000000                        START     0
000000    6D0003              LDS       #3
000003    75012c              LDT       #300
000006    050000              LDX       #0
000009    010000              LDA       #0
00000c    0FA00a    LOOP      STA       ARR,X
00000f    9041                ADDR      S,X
000011    A015                COMPR     X,T
000013    3B2ff6              JLT       LOOP
000016    3F2ffd              J         *
000019              ARR       RESW      100
000145                        END

```



```
line no.  address   label     op-code   operand   comment
                    .23456789012345678901234567890123456
0         000000    .SET elements of a 100-word array to 0
0         000000    .Label.  Opcode  The  O p e r a n d
0         000000    .
1         000000              START     0
2         000000              LDS       #3
3         000003              LDT       #300
4         000006              LDX       #0
5         000009              LDA       #0
6         00000c    LOOP      STA       ARR,X
7         00000f              ADDR      S,X
8         000011              COMPR     X,T
9         000013              JLT       LOOP
10        000016              J         *
11        000019    ARR       RESW      100
12        000145              END
                    Symbol Table (value in decimal)

          Name      Value     Reloc/Absol

          ARR       25        Relocatable
          LOOP      12        Relocatable
```

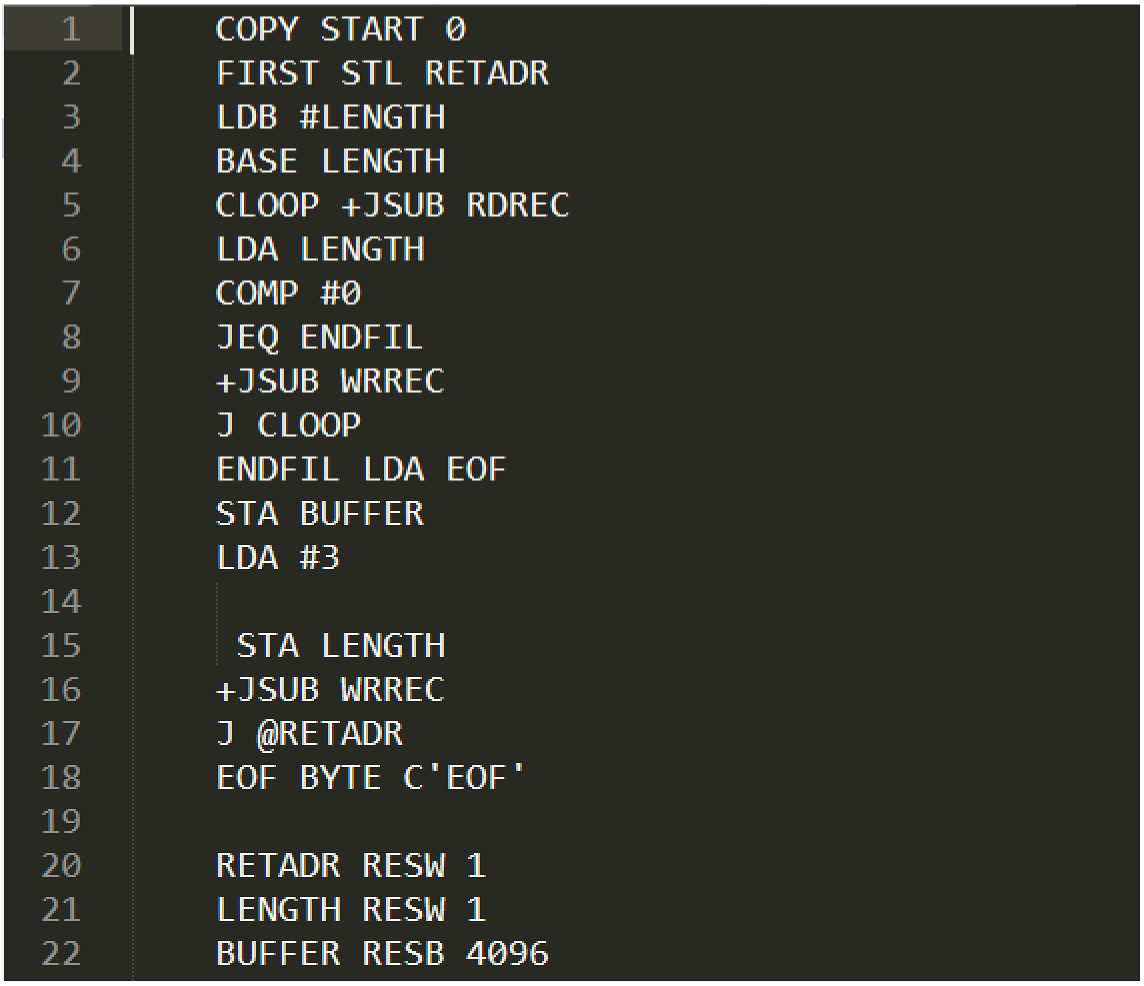
2:



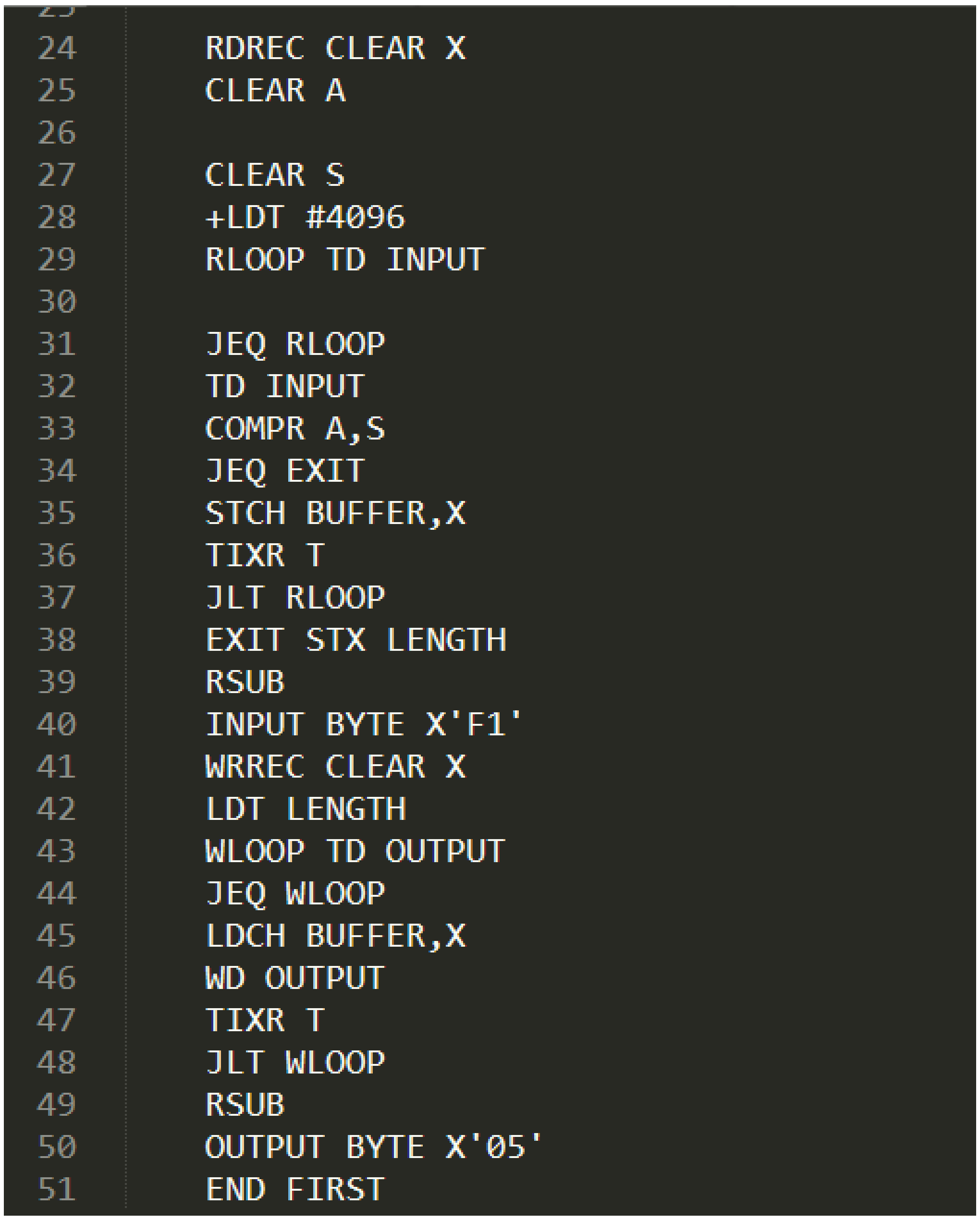
C:\Windows\System32\cmd.exe

```
Successful Pass1
Successful Pass2

C:\Users\MohamedSamir\eclipse-workspace\assembler>assembler test.txt
file :test.txt
reading file :test.txt: Done
Successful Pass1
Failed Pass2

C:\Users\MohamedSamir\eclipse-workspace\assembler>
```

Test.txt:

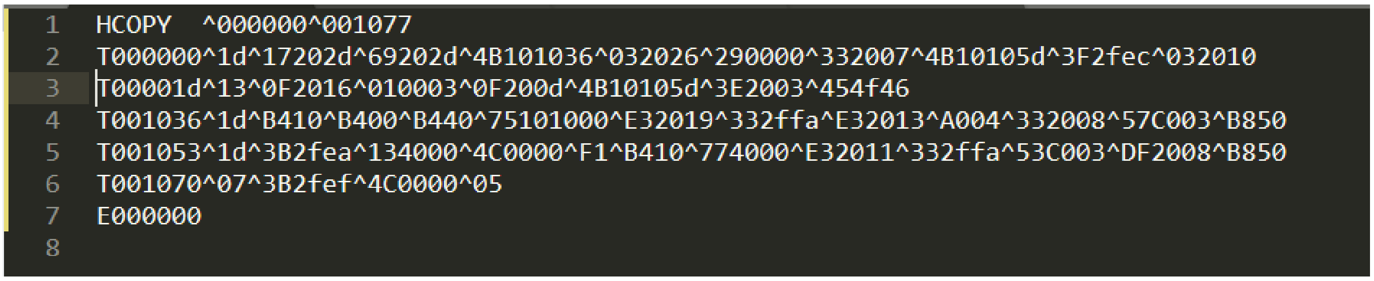


```
COPY START 0
FIRST STL RETADR
LDB #LENGTH
BASE LENGTH
CLOOP +JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
+JSUB WRREC
J CLOOP
ENDFIL LDA EOF
STA BUFFER
LDA #3

 STA LENGTH
+JSUB WRREC
J @RETADR
EOF BYTE C'EOF'

RETADR RESW 1
LENGTH RESW 1
BUFFER RESB 4096
```



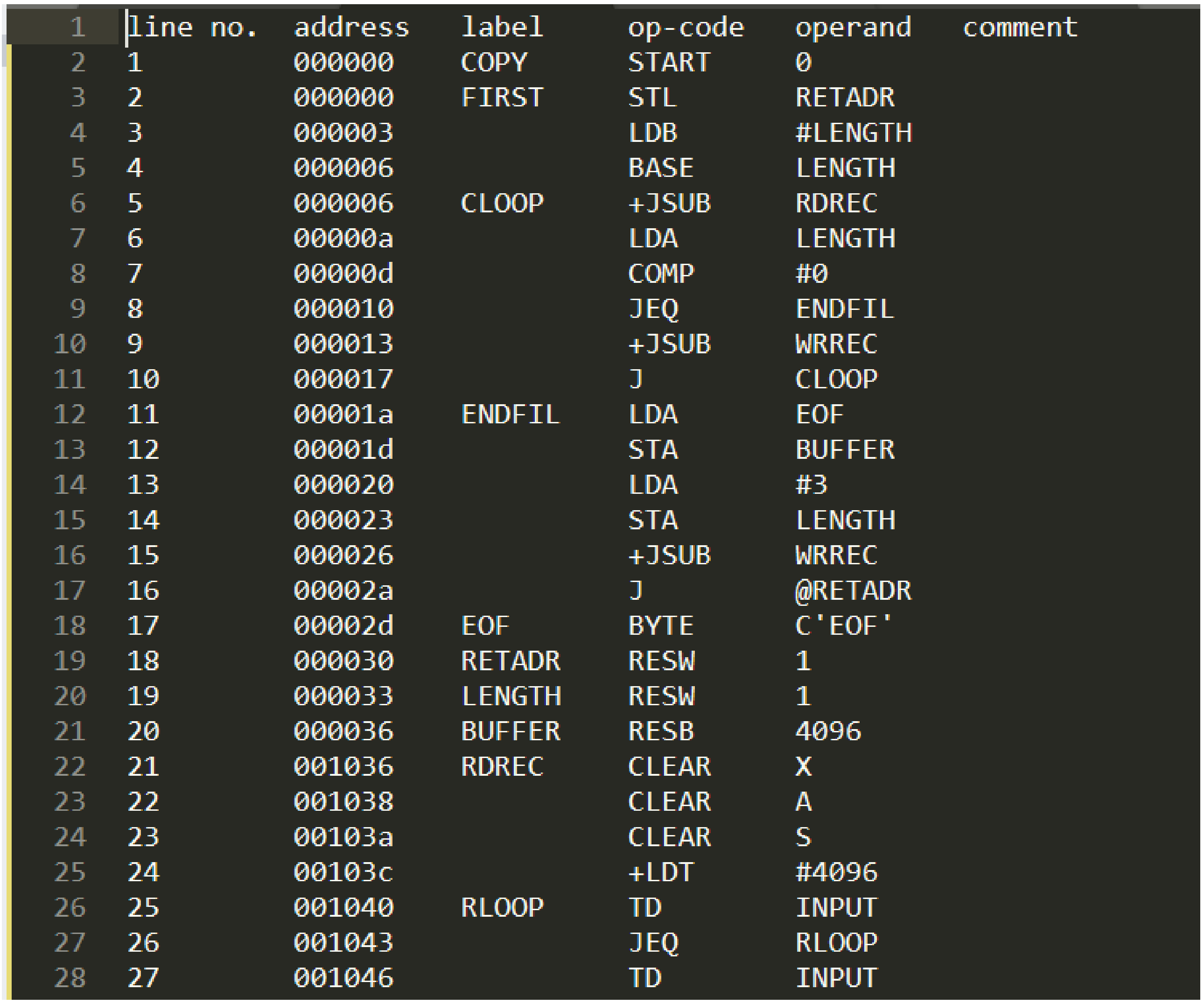
```
RDREC CLEAR X
CLEAR A

CLEAR S
+LDT #4096
RLOOP TD INPUT

JEQ RLOOP
TD INPUT
COMPR A,S
JEQ EXIT
STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT STX LENGTH
RSUB
INPUT BYTE X'F1'
WRREC CLEAR X
LDT LENGTH
WLOOP TD OUTPUT
JEQ WLOOP
LDCH BUFFER,X
WD OUTPUT
TIXR T
JLT WLOOP
RSUB
OUTPUT BYTE X'05'
END FIRST
```

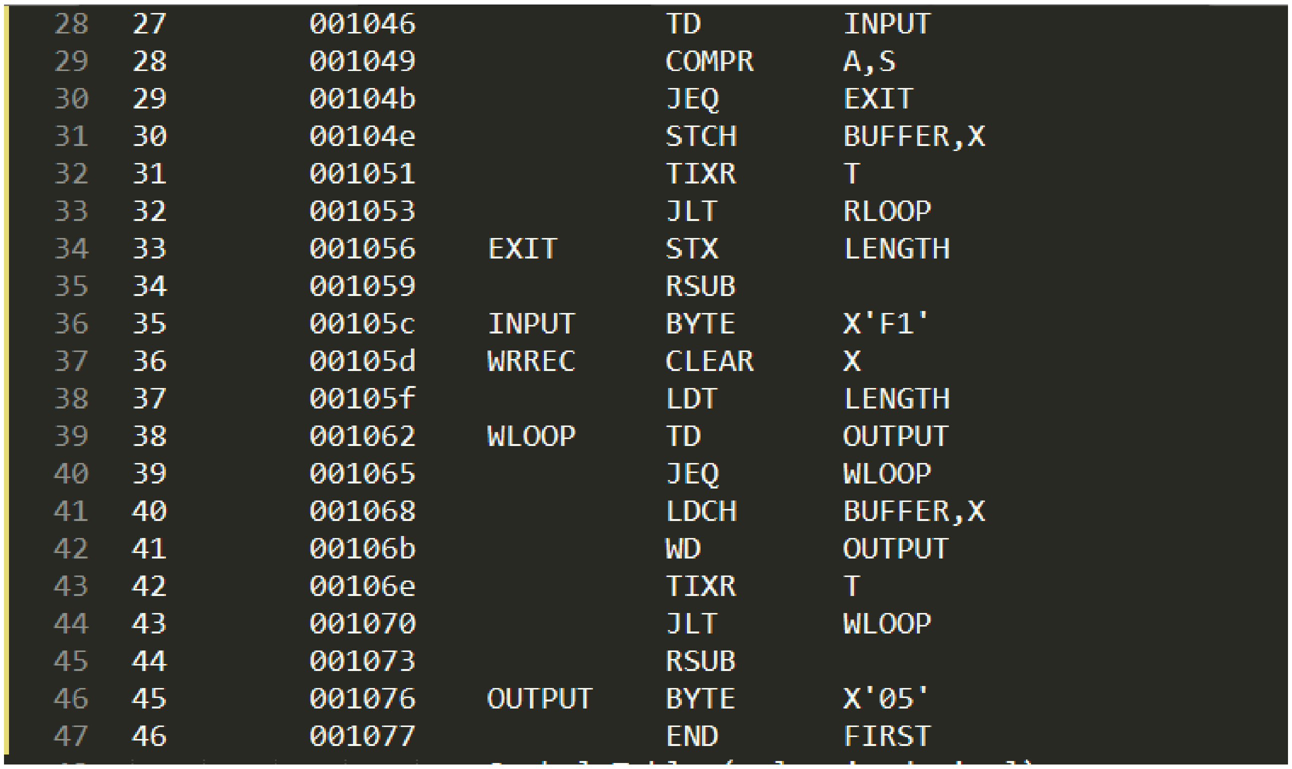
OBJECTFILE.txt:



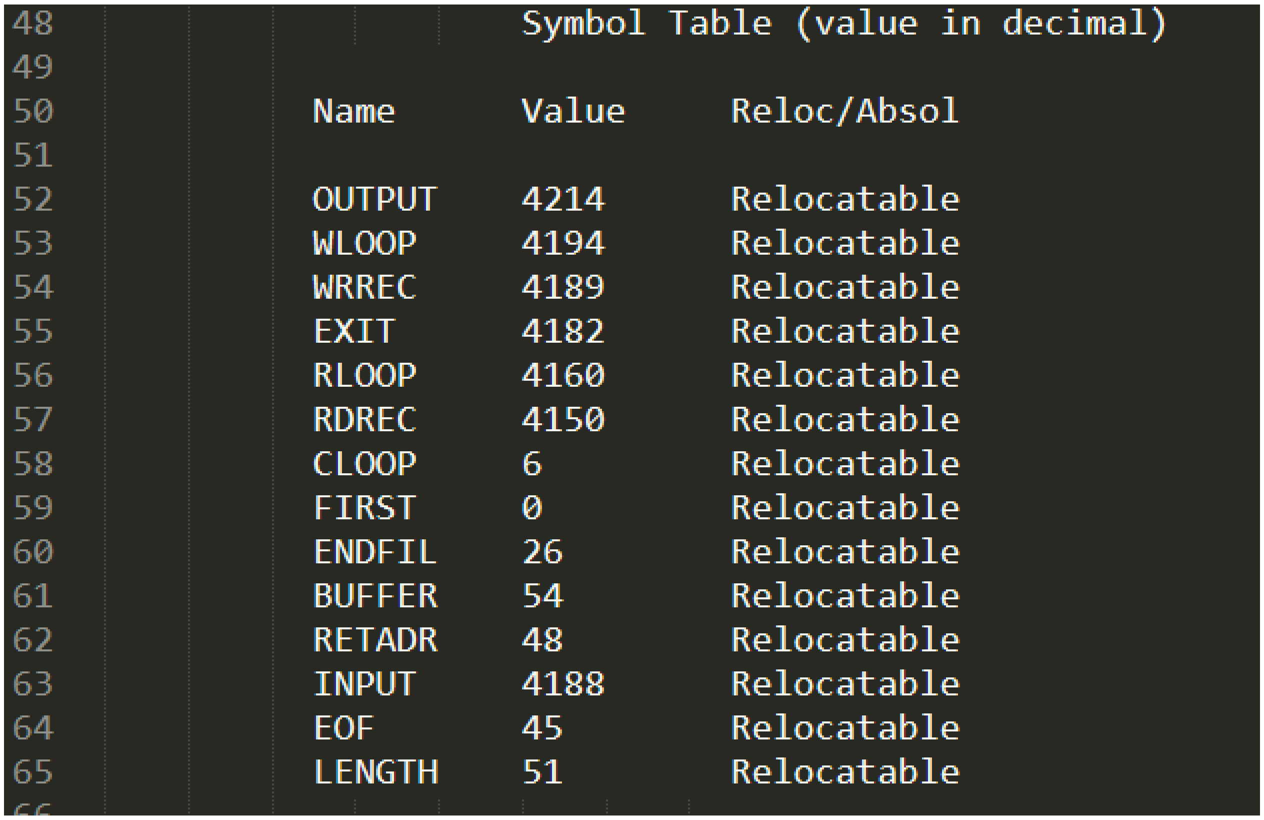
```
HCOPY  ^000000^001077
T000000^1d^17202d^69202d^4B101036^032026^290000^332007^4B10105d^3F2fec^032010
T00001d^13^0F2016^010003^0F200d^4B10105d^3E2003^454f46
T001036^1d^B410^B400^B440^75101000^E32019^332ffa^E32013^A004^332008^57C003^B850
T001053^1d^3B2fea^134000^4C0000^F1^B410^774000^E32011^332ffa^53C003^DF2008^B850
T001070^07^3B2fef^4C0000^05
E000000
```

LISTFILE.txt:

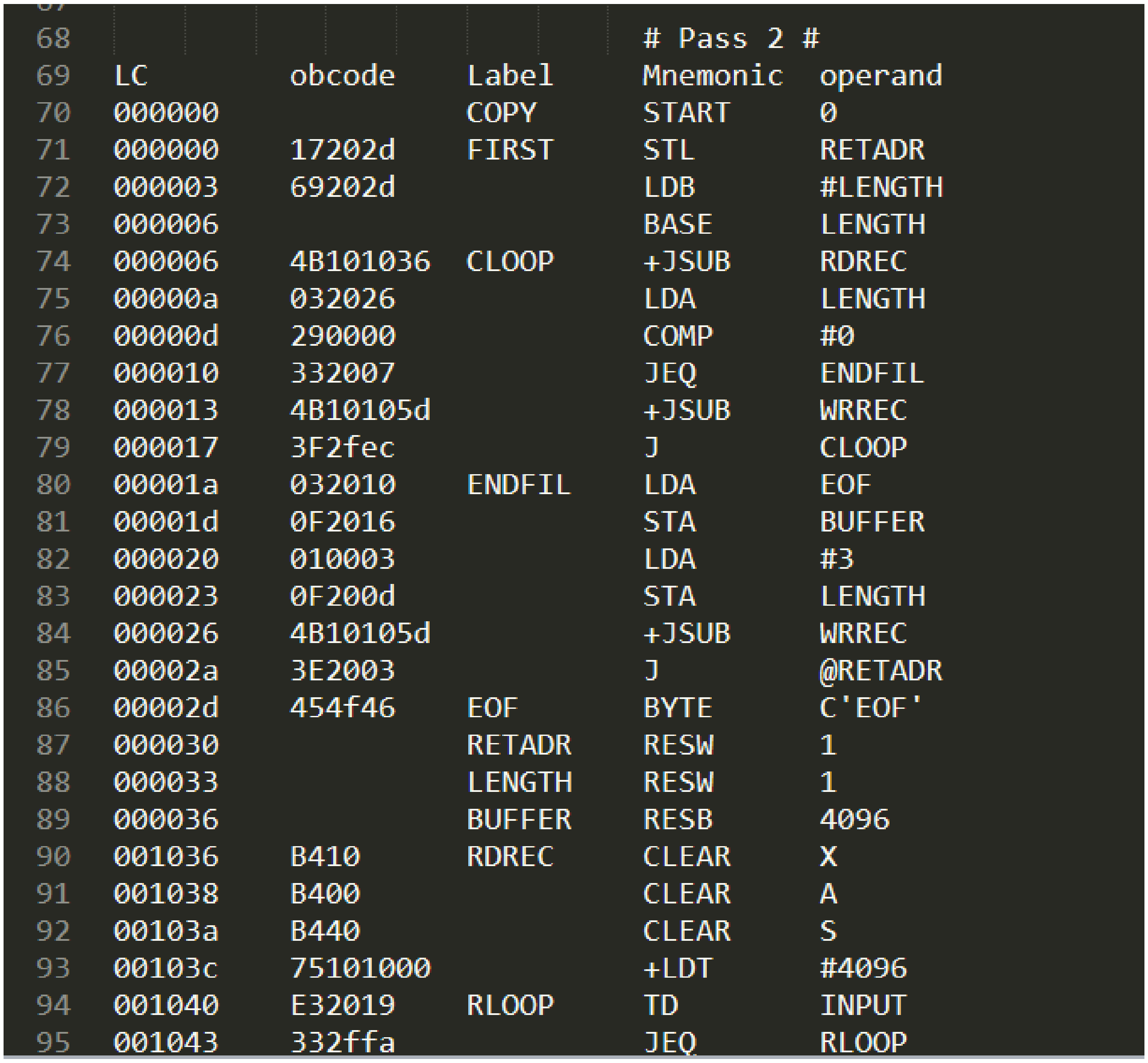

```
line no.  address   label     op-code   operand   comment
1         000000    COPY      START     0
2         000000    FIRST     STL       RETADR
3         000003              LDB       #LENGTH
4         000006              BASE      LENGTH
5         000006    CLOOP     +JSUB     RDREC
6         00000a              LDA       LENGTH
7         00000d              COMP      #0
8         000010              JEQ       ENDFIL
9         000013              +JSUB     WRREC
10        000017              J         CLOOP
11        00001a    ENDFIL    LDA       EOF
12        00001d              STA       BUFFER
13        000020              LDA       #3
14        000023              STA       LENGTH
15        000026              +JSUB     WRREC
16        00002a              J         @RETADR
17        00002d    EOF       BYTE      C'EOF'
18        000030    RETADR    RESW      1
19        000033    LENGTH    RESW      1
20        000036    BUFFER    RESB      4096
21        001036    RDREC     CLEAR     X
22        001038              CLEAR     A
23        00103a              CLEAR     S
24        00103c              +LDT      #4096
25        001040    RLOOP     TD        INPUT
26        001043              JEQ       RLOOP
27        001046              TD        INPUT
```



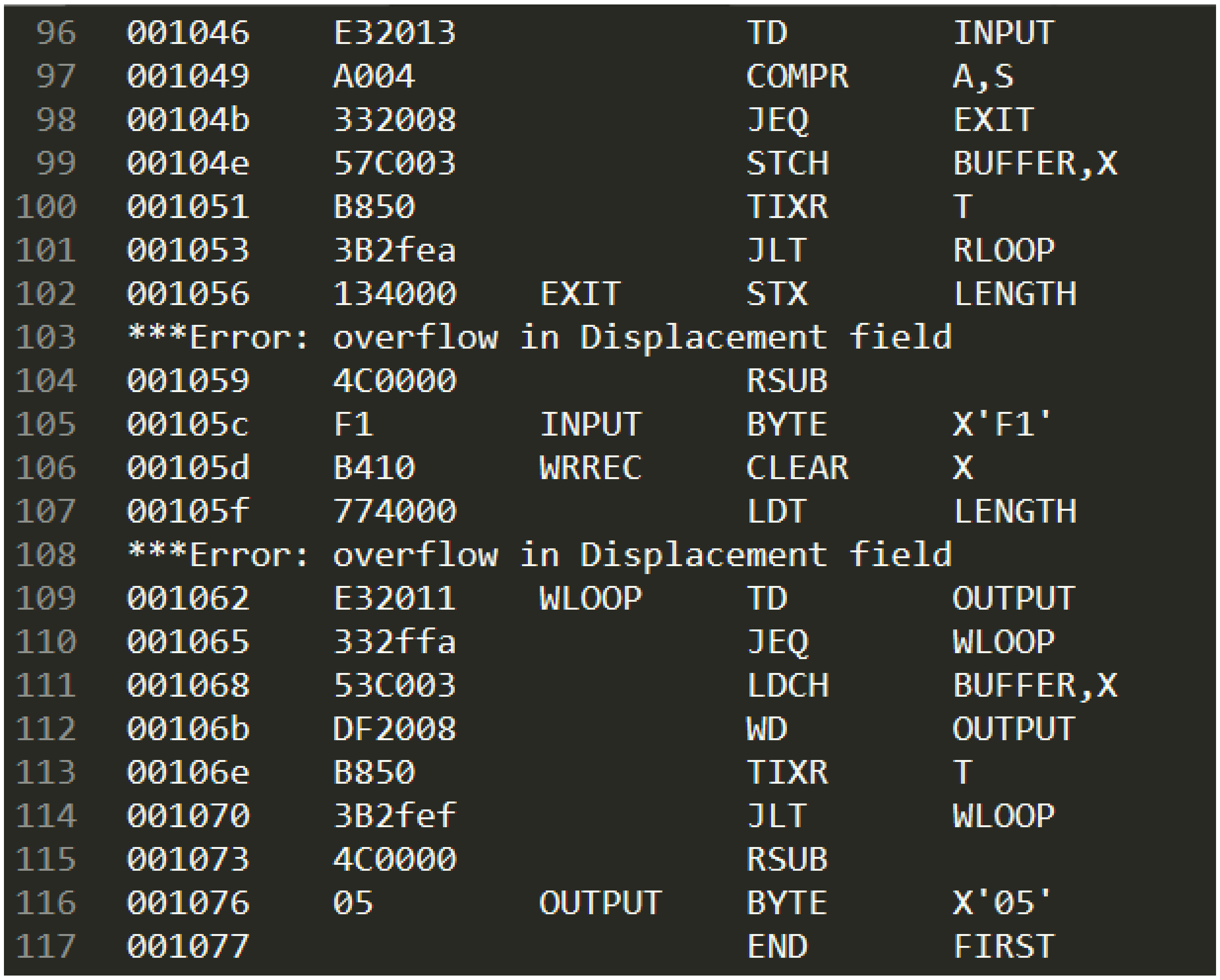
```
27      001046              TD        INPUT
28      001049              COMPR     A,S
29      00104b              JEQ       EXIT
30      00104e              STCH      BUFFER,X
31      001051              TIXR      T
32      001053              JLT       RLOOP
33      001056    EXIT      STX       LENGTH
34      001059              RSUB
35      00105c    INPUT     BYTE      X'F1'
36      00105d    WRREC     CLEAR     X
37      00105f              LDT       LENGTH
38      001062    WLOOP     TD        OUTPUT
39      001065              JEQ       WLOOP
40      001068              LDCH      BUFFER,X
41      00106b              WD        OUTPUT
42      00106e              TIXR      T
43      001070              JLT       WLOOP
44      001073              RSUB
45      001076    OUTPUT    BYTE      X'05'
46      001077              END       FIRST
```



```
                    Symbol Table (value in decimal)

           Name         Value       Reloc/Absol

           OUTPUT       4214        Relocatable
           WLOOP        4194        Relocatable
           WRREC        4189        Relocatable
           EXIT         4182        Relocatable
           RLOOP        4160        Relocatable
           RDREC        4150        Relocatable
           CLOOP        6           Relocatable
           FIRST        0           Relocatable
           ENDFIL       26          Relocatable
           BUFFER       54          Relocatable
           RETADR       48          Relocatable
           INPUT        4188        Relocatable
           EOF          45          Relocatable
           LENGTH       51          Relocatable
```



```
                                        # Pass 2 #
LC        obcode    Label     Mnemonic  operand
000000              COPY      START     0
000000    17202d    FIRST     STL       RETADR
000003    69202d              LDB       #LENGTH
000006                        BASE      LENGTH
000006    4B101036  CLOOP     +JSUB     RDREC
00000a    032026              LDA       LENGTH
00000d    290000              COMP      #0
000010    332007              JEQ       ENDFIL
000013    4B10105d            +JSUB     WRREC
000017    3F2fec              J         CLOOP
00001a    032010    ENDFIL    LDA       EOF
00001d    0F2016              STA       BUFFER
000020    010003              LDA       #3
000023    0F200d              STA       LENGTH
000026    4B10105d            +JSUB     WRREC
00002a    3E2003              J         @RETADR
00002d    454f46    EOF       BYTE      C'EOF'
000030              RETADR    RESW      1
000033              LENGTH    RESW      1
000036              BUFFER    RESB      4096
001036    B410      RDREC     CLEAR     X
001038    B400                CLEAR     A
00103a    B440                CLEAR     S
00103c    75101000            +LDT      #4096
001040    E32019    RLOOP     TD        INPUT
001043    332ffa              JEQ       RLOOP
```



```
001046     E32013                   TD        INPUT
001049     A004                     COMPR     A,S
00104b     332008                   JEQ       EXIT
00104e     57C003                   STCH      BUFFER,X
001051     B850                     TIXR      T
001053     3B2fea                   JLT       RLOOP
001056     134000     EXIT          STX       LENGTH
***Error: overflow in Displacement field
001059     4C0000                   RSUB
00105c     F1         INPUT         BYTE      X'F1'
00105d     B410       WRREC         CLEAR     X
00105f     774000                   LDT       LENGTH
***Error: overflow in Displacement field
001062     E32011     WLOOP         TD        OUTPUT
001065     332ffa                   JEQ       WLOOP
001068     53C003                   LDCH      BUFFER,X
00106b     DF2008                   WD        OUTPUT
00106e     B850                     TIXR      T
001070     3B2fef                   JLT       WLOOP
001073     4C0000                   RSUB
001076     05         OUTPUT        BYTE      X'05'
001077                              END       FIRST
```