ASSEMBLER

Systems Programming-1
Final-Project (phase-2)

1) Mohamed Nabil (62).

2) Nesma Emad El din (72).

3) Caroline Adel (49).

4) Amr Fathy Al dafrawy (48).

Requirement Specification:

It is required to implement Phase-2 of a (cross) Assembler for (a subset of) SIC/XE machines.

Phase-2 specification requires the following:

1) Input:

Source file name "src.txt" via a (.exe).

2) Process:

The input source file is parsed in order to produce pass-1, after that the symbol table produced in phase-1 is used while forming phase-2 assembled file and object code.

The assembler for phase-2 should support:

1. EQU and ORG statements.

2. 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.

3) Output:

The output of the assembler should include (at least):

1. 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.
2. A report at the end of pass2.
Pass1 and Pass2 errors should be included as
part of the assembler report,
exhibiting both the offending line of source code and the error.

Design:

Pass-2 assembler:

- Source lines are read in sequence.
- The lines are passed to a parser method which supports free formatting.
- The parser method, along with other methods, is used as discussed before to output the pass-1 for the assembler and save the symbol table.

- If pass-1 was performed successfully pass-2 is then executed.
- In pass-2 every line entry is checked for producing the op-code.
- Errors in pass-2 are recorded so as to determine whether the object file will be created or not.

Algorithms Description:

The implementation is encapsulated into one class

assemble.cpp (as specified in the description) this class includes

A. Class Entry():

- This class entity represents the lines which are written to the output file.
- Its member variables are (int loc) to hold address of the current entry, and a set of strings (label, op_code, operand, comment, error, ObjectCode) to describe entry.

B. A set of methods described below:

1) checkByte:

- Parameters:

  A string which is the operand to the BYTE directive.

- **Return type:**

  Returns an integer which is the value of the operand.

- **Functionality:**

  Checks for byte declaration syntax, returns its value if correct syntax, -1 otherwise.

2) checkWord:

- **Parameters:**

  A string which is the operand to the WORD directive.

- **Return type:**

  Returns an integer which is the value of the operand.

- **Functionality:**

  Checks for word declaration syntax, returns its value if correct syntax, -1 otherwise.

3) toDecimal:

- **Parameters:**

  A string which is to be converted.

- **Return type:**

An integer which is the hexadecimal conversion of the string.

- **Functionality:**

  Convert a string into hexadecimal notation.

- **Description:**

  The result is added up via converting every character into its integer notation and then adding the correct weighting according to the decimal place of the digit (16 ^ weight).

4)toInteger:

- **Parameters:**

  A string which is to be converted.

- **Return type:**

  An integer which is the decimal conversion of the string.

- **Functionality:**

  Convert a string into decimal notation.

- **Description:**

  The result is added up via converting every character into its integer notation

and then adding the correct weighting according to the decimal place of the digit (10^ weight).

5) toLower:

- Parameters:

  String to be converted.

- Return type:

  String after conversion.

- Functionality:

  Convert all characters of the input string to lower case.

6) checkOperand:

- Parameters:

  Two strings, the first is an opcode and the second is its corresponding operand.

- Return type:

  A Boolean to denote the result of the check.

- Functionality:

  To check whether the operand is a valid match for the opcode or not.

- **Description:**

  It first checks for single operand instructions, as "tixr" and "clear", if their corresponding operand is a register true value is returned otherwise false.

  Then it checks the format for opcodes that take 2 operands, it checks both are register names and they are separated by a comma, if so it returns true otherwise false.

7) validateOpcode:

- **Parameters:**

  A string which is the opcode to be validated.

- **Return type:**

  An integer value.

- **Functionality:**

  Checks for the opcode in the opcode map, returns an integer value denoting the byte format for the given opcode if found, if the opcode is invalid it returns -1.

8) isDuplicateLabel:

- **Parameters:**

  The string of which existence is to be checked for duplicity.

- **Return type:**

  A Boolean to denote the check.

- **Functionality:**

  Returns true if the label is a duplicate, false if not.

9)oneWord:

- **Parameters:**

  A vector of strings containing parameters of a source code line of length 1, a vector of strings holding comment line and an integer denoting the format.

- **Return type:**

  Void.

- **Functionality:**

  Creates a new Entry instance and adds it up to the source code entry table.

  Updates the current address with the correct value.

10) twoWord:

- Parameters:

  A vector of strings containing parmeters of a source code line of length 2, a vector of strings holding comment line and an integer denoting the format.

- Return type:

  Void.

- Functionality:

  Creates a new Entry instance and adds it up to the source code entry table.

  Updates the current address with the correct value.

11) threeWord:

- Parameters:

  A vector of strings containing parmeters of a source code line of length 3, a vector of strings holding comment line and an integer denoting the format.

- Return type:

  Void.

- Functionality:

Creates a new Entry instance and adds it up to the source code entry table.

If a possible error is encountered it is added.

Updates the current address with the correct value.

12) parse_sic:

- Parameters:

  String (line of the source code) to be parsed.

- Return type:

  Void.

- Functionality:

  Check line syntax, detect errors and convert the input into a form valid for later purpose (writing the list file).

  Handles fixed format parsing.

- Description:

  The input line is first checked for its type if It is a comment it is added as a comment, otherwise the parsing process both checks for syntax and starts separating line

entries, converting them to lower case strings, validating opcodes and whether they match their corresponding operands in order to add the line as a new Entry.

It also checks that appropriate types of code are in their correct positions according to sic machine fixed formatting rules.

13) parse:

- Parameters:

  String (line of the source code) to be parsed.

- Return type:

  Void.

- Functionality:

  Check line syntax, detect errors and convert the input into a form valid for later purpose (writing the list file).

  Handles the free format parsing.

- Description:

  The input line is first checked for its type if It is a comment it is added as a comment, otherwise the parsing process both checks

for syntax and starts separating line entries, converting them to lower case strings, validating opcodes and whether they match their corresponding operands in order to add the line as a new Entry.

Only separates strings upon spaces since no kind of formatting specification is imposed in free formatting.

14) **fillingMap:**

- **Parameters:**

  Void.

- **Return type:**

  Void.

- **Functionality:**

  Filling up the map holding the opcodes and their formats from an external text file.

15) **checkEndLine:**

- **Parameters:**

  String which is the last line in the source code.

- **Return type:**

Void.

- **Functionality:**

  Checks that the end line of the program is valid one.

16) **checkSpace:**

- **Parameters:**

  String to be checked.

- **Return type:**

  Boolean denoting check result.

- **Functionality:**

  Checks for spaces in a string, returns true if at least one character was found, false if not (the line is all spaces).

17) **checkOrg:**

- **Parameters:**

  String which is the operand for 'org' directive.

- **Return type:**

  Int which is the address to bs assigned for the current line.

- **Functionality:**

18) checkEqu:

- Parameters:

  String which is the operand for the 'equ' directive, and a string which is the label for the 'equ' directive.

- Return type:

  Int which corresponds to the address to be assigned to this line.

- Functionality:

19) formate1:

- Parameters:

  String which is an instruction of format 1.

- Return type:

  String holding the corresponding opcode.

- Functionality:

  A member of a set of functions designed to return an opcode for a certain

instruction line according to its format used, format -1 is used in this case.

20) formate2:

- Parameters:

  String which is the instruction, and another string which is the operand to an instruction of format 2.

- Return type:

  A string holding the corresponding opcode.

- Functionality:

  A member of a set of functions designed to return an opcode for a certain instruction line according to its format used, format -2 is used in this case, it reviews the registers -operands- and accordingly calculates the opcode.

21) formate3_4:

- Parameters:

  Integer which holds the index of the current line in the entries table,

  String which is the instruction, and another string which is the operand to an

instruction of format 2, and an integer holding the current address of the line.

- Return type:

  A string holding the corresponding opcode.

- Functionality:

  A member of a set of functions designed to return an opcode for a certain instruction line according to its format used, format -3 || 4 are used in this case, it reviews the instruction opcode, calculates the corresponding "nixbpe" values and evaluates the object code.

22) eval_address:

- Parameters:

  String which is the operand for a certain instruction.

- Return type:

  Returns integer which is address for corresponding operand.

- Functionality:

  Evaluates the address for all formats, it Is used to get the TA for a certain

instruction whether from the symbol table or otherwise.

23) ObjectCode:

- Parameters:

  Void.

- Return type:

  Void .

- Functionality:

  Method responsible for assigning objectcodes for all source code lines in order to be printed out In the main method in pass-2. This is only when pass-1 is error free.

24) buildObjectFile:

- Parameters:

  Void.

- Return type:

  Void.

- Functionality:

  Method responsible for building up the object file if and only if pass-2 is error free.

25) modifyLocation:

- Parameters:

- Return type:
- Functionality:

Main Data Structures:

1) Maps:

A map data structure is used:

- For the symbol table:

  It is used to insert labels in the symbol table and to easily be able to retrieve them or check for their duplicity in constant time.

- For saving up SIC/XE machine appendix:

  A two dimensional map is used to save up SIC/XE machine instruction set and their corresponding format.

- For saving up SIC/XE machine instructions op-codes.

Assumptions:

1) Errors for pass-1 are produced as follows:

- In case of an invalid operand
  - "****Error: Invalid Operand".
- If line length is exceeded above limit
  - "****Error: Invalid length of the line".
- Invalid line spaces
  - "****Error: invalid spaces in this line".
- Invalid beginning
  - "****Error: invalid start of the program".
- Invalid op code
  - "****Error: Invalid OpCode".
- Duplicate symbols
  - "****Error: Duplicate Symbol".
- Invalid entry
  - "****Error: Invalid Entry".
- Invalid end program
  - "****Error: invalid end of the program".

2) Errors for pass-2 are produces as follows:

3) Free format is used , not fixed format.

4) Operands and Labels cannot include space characters.

Sample Runs:

First sample:

Input:

```
.23456789012345678901234
COPY     START   0
FIRST    STL     RETADR
         LDB     #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
         RD      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
```

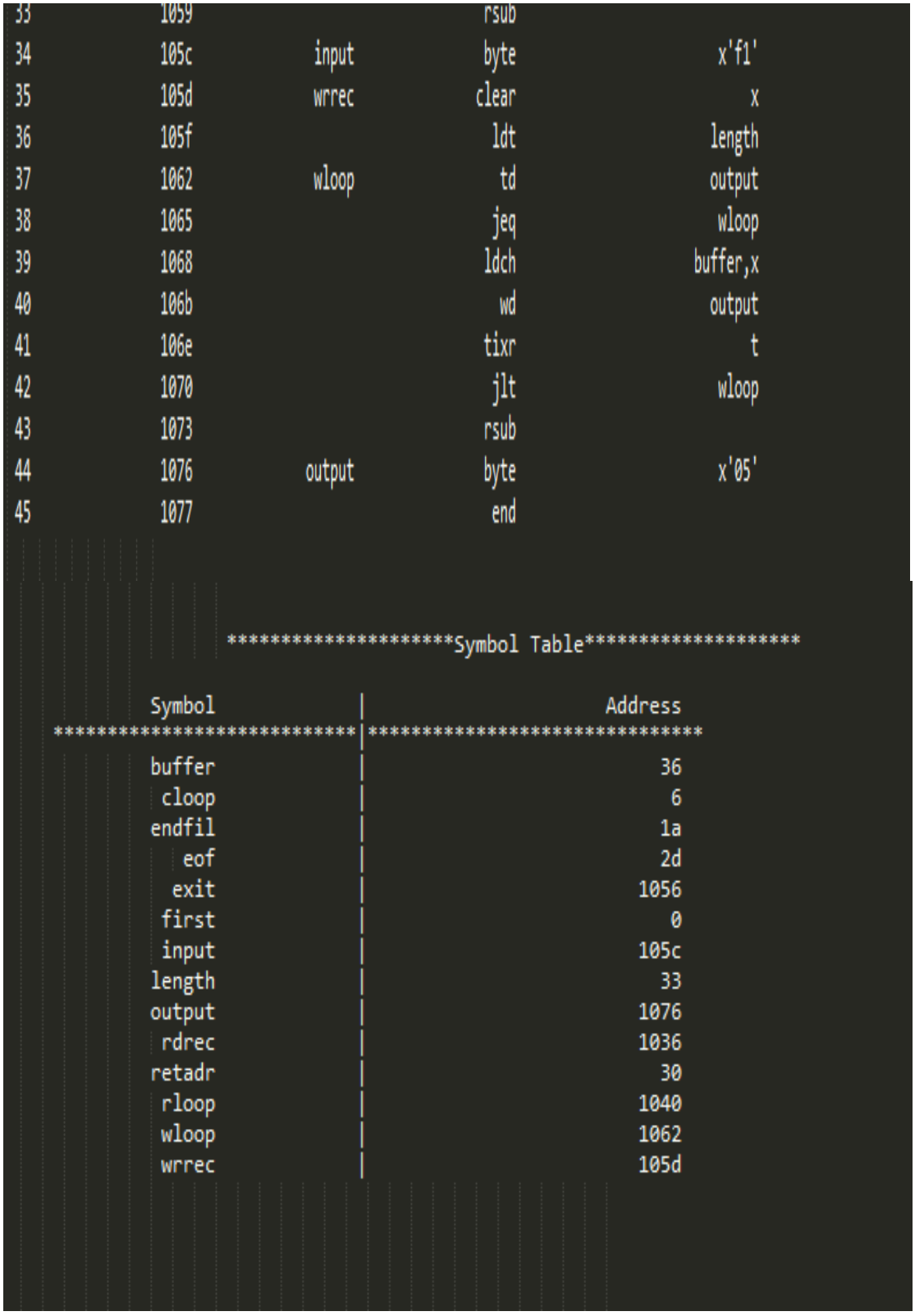
```
         JLT     WLOOP
         RSUB
OUTPUT   BYTE    X'05'
         END
```

Output:

PASS 1

.....

LineNo	Adress	Label	Op-code	Operand
0				
0			.23456789012345678901 23	
1	0	copy	start	0
2	0	first	stl	retadr
3	3		ldb	#length
4	6	cloop	+jsub	rdrec
5	a		lda	length
6	d		comp	#0
7	10		jeq	endfil
8	13		+jsub	wrrec
9	17		j	cloop
10	1a	endfil	lda	eof
11	1d		sta	buffer
12	20		lda	#3
13	23		sta	length
14	26		+jsub	wrrec
15	2a		j	@retadr
16	2d	eof	byte	C'EOF'
17	30	retadr	resw	1
18	33	length	resw	1
19	36	buffer	resb	4096
20	1036	rdrec	clear	x
21	1038		clear	a
22	103a		clear	s
23	103c		+ldt	#4096
24	1040	rloop	td	input
25	1043		jeq	rloop
26	1046		rd	input
27	1049		compr	a,s
28	104b		jeq	exit
29	104e		stch	buffer,x
30	1051		tixr	t
31	1053		jlt	rloop
32	1056	exit	stx	length



33	1059		rsub	
34	105c	input	byte	x'f1'
35	105d	wrrec	clear	x
36	105f		ldt	length
37	1062	wloop	td	output
38	1065		jeq	wloop
39	1068		ldch	buffer,x
40	106b		wd	output
41	106e		tixr	t
42	1070		jlt	wloop
43	1073		rsub	
44	1076	output	byte	x'05'
45	1077		end	

**********Symbol Table**********

Symbol	Address
buffer	36
cloop	6
endfil	1a
eof	2d
exit	1056
first	0
input	105c
length	33
output	1076
rdrec	1036
retadr	30
rloop	1040
wloop	1062
wrrec	105d

PASS 2

.....

LineNo	Adress	Label	Op-code	Operand	ObjectCode
0					
0			.2345678901234567890123		
1	0	copy	start	0	
2	0	first	stl	retadr	17202d
3	3		ldb	#length	69202d
4	6	cloop	+jsub	rdrec	4b101036
5	a		lda	length	032026
6	d		comp	#0	290000
7	10		jeq	endfil	332007
8	13		+jsub	wrrec	4b10105d
9	17		j	cloop	3f2fec
10	1a	endfil	lda	eof	032010
11	1d		sta	buffer	0f2016
12	20		lda	#3	010003
13	23		sta	length	0f200d
14	26		+jsub	wrrec	4b10105d
15	2a		j	@retadr	3e2003
16	2d	eof	byte	C'EOF'	454f46
17	30	retadr	resw	1	
18	33	length	resw	1	
19	36	buffer	resb	4096	
20	1036	rdrec	clear	x	b410
21	1038		clear	a	b400
22	103a		clear	s	b440
23	103c		+ldt	#4096	75101000
24	1040	rloop	td	input	e32019
25	1043		jeq	rloop	332ffa
26	1046		rd	input	db2013
27	1049		compr	a,s	a004
28	104b		jeq	exit	332008
29	104e		stch	buffer,x	57c003
30	1051		tixr	t	b850
31	1053		jlt	rloop	3b2fea
32	1056	exit	stx	length	134000
33	1059		rsub		4f0000
34	105c	input	byte	x'f1'	f1
35	105d	wrrec	clear	x	b410
36	105f		ldt	length	774000
37	1062	wloop	td	output	e32011
38	1065		jeq	wloop	332ffa
39	1068		ldch	buffer,x	53c003
40	106b		wd	output	df2008
41	106e		tixr	t	b850
42	1070		jlt	wloop	3b2fef
43	1073		rsub		4f0000
44	1076	output	byte	x'05'	05
45	1077		end		

Object file:

```
Hcopy000000001077
T0000001d17202d69202d4b10103603202629000003320074b10105d3f2fec032010
T00001d190f20160100030f200d4b10105d3e2003454f46b410b400b440
T00103c1a75101000e32019332ffadb2013a00433200857c003b8503b2fea
T001056181340004f0000f1b410774000e32011332ffa53c003df2008
T00106e09b8503b2fef4f000005
M00000705
M00001405
M00002705
E000000
```

Second sample:

Input:

```
.2345678901234567890123456
BUBBLE   START   0
         LDT     #1
LOOP1    LDA     IIND
         COMP    LEN
         JEQ     EXIT
         LDA     #0
         STA     JIND
LOOP2    LDX     JIND
         LDA     LEN
         SUB     #1
         SUB     IIND
         COMPR   X,A
         JLT     CON
         LDA     IIND
         ADD     #1
         STA     IIND
         J       LOOP1
CON      LDCH    STR,X
         STCH    CH1
         LDA     JIND
         ADD     #1
         STA     JIND
         LDX     JIND
         LDCH    STR,X
         STCH    CH2
         LDCH    CH1
         RMO     A,S
         LDCH    CH2
         COMPR   S,A
         JGT     SWAPIT
         J       LOOP2
SWAPIT   LDX     JIND
         LDB     STR
         LDCH    CH1
         STCH    STR,X
         LDB     STR
         SUBR    T,X
         LDCH    CH2
         STCH    STR,X
         LDB     STR
         J       LOOP2
EXIT     J       *
IIND     WORD    0
JIND     WORD    0
STR      BYTE    C'54321'
LEN      WORD    5
CH1      BYTE    C' '
CH2      BYTE    C' '
         END
```

Output:

PASS 1

....

LineNo	Adress	Label	Op-code	Operand
0				
0			.23456789012345678901234 5	
1	0	bubble	start	0
2	0		ldt	#1
3	3	loop1	lda	iind
4	6		comp	len
5	9		jeq	exit
6	c		lda	#0
7	f		sta	jind
8	12	loop2	ldx	jind
9	15		lda	len
10	18		sub	#1
11	1b		sub	iind
12	1e		compr	x,a
13	20		jlt	con
14	23		lda	iind
15	26		add	#1
16	29		sta	iind
17	2c		j	loop1
18	2f	con	ldch	str,x
19	32		stch	ch1
20	35		lda	jind
21	38		add	#1
22	3b		sta	jind
23	3e		ldx	jind
24	41		ldch	str,x
25	44		stch	ch2
26	47		ldch	ch1
27	4a		rmo	a,s
28	4c		ldch	ch2
29	4f		compr	s,a
30	51		jgt	swapit
31	54		j	loop2
32	57	swapit	ldx	jind

```
33                5a                          ldb                       str
34                5d                         ldch                       ch1
35                60                         stch                    str,x
36                63                          ldb                       str
37                66                         subr                       t,x
38                68                         ldch                       ch2
39                6b                         stch                    str,x
40                6e                          ldb                       str
41                71                            j                     loop2
42                74                exit        j                         *
43                77                iind     word                         0
44                7a                jind     word                         0
45                7d                 str     byte                 C'54321'
46                82                 len     word                         5
47                85                 ch1     byte                     C' '
48                86                 ch2     byte                     C' '
49                87                          end
```

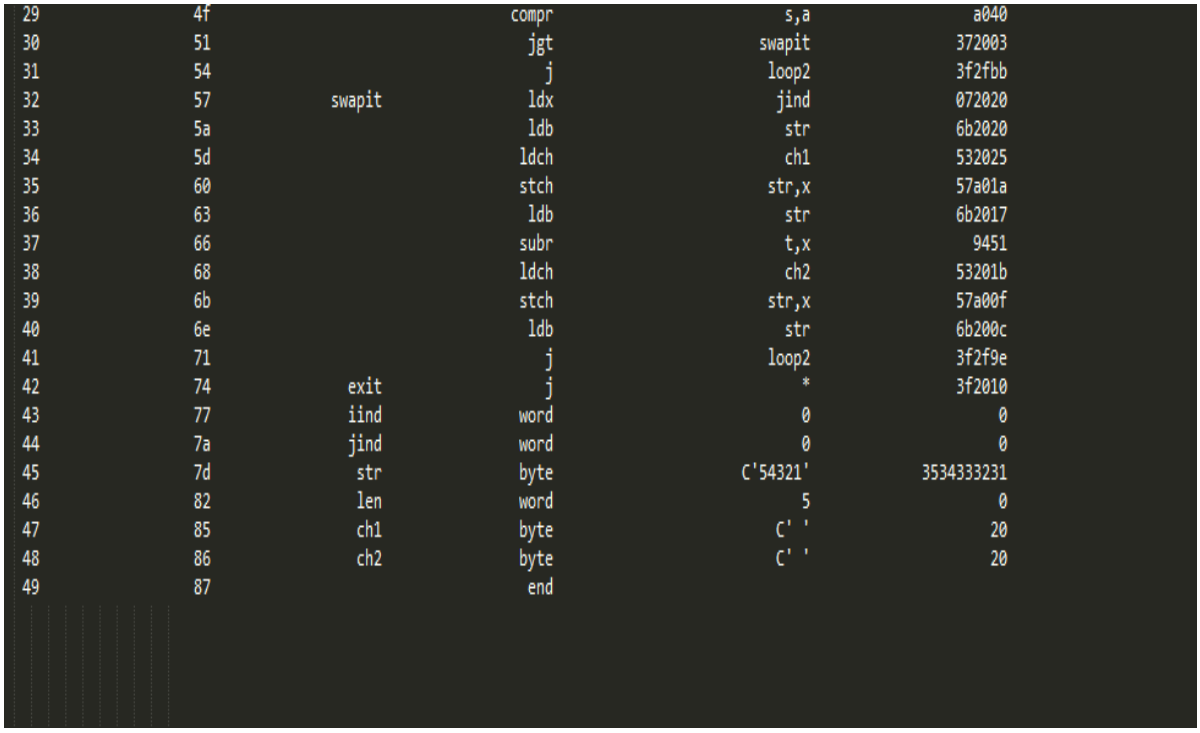
**********Symbol Table**********

Symbol	Address
ch1	85
ch2	86
con	2f
exit	74
iind	77
jind	7a
len	82
loop1	3
loop2	12
str	7d
swapit	57

PASS 2

.....

LineNo	Adress	Label	Op-code	Operand	ObjectCode
0					
0			.2345678901234567890123 45		
1	0	bubble	start	0	
2	0		ldt	#1	750001
3	3	loop1	lda	iind	032071
4	6		comp	len	2b2079
5	9		jeq	exit	332068
6	c		lda	#0	010000
7	f		sta	jind	0f2068
8	12	loop2	ldx	jind	072065
9	15		lda	len	03206a
10	18		sub	#1	1d0001
11	1b		sub	iind	1f2059
12	1e		compr	x,a	a010
13	20		jlt	con	3b200c
14	23		lda	iind	032051
15	26		add	#1	190001
16	29		sta	iind	0f204b
17	2c		j	loop1	3f2fd4
18	2f	con	ldch	str,x	53a04b
19	32		stch	ch1	572050
20	35		lda	jind	032042
21	38		add	#1	190001
22	3b		sta	jind	0f203c
23	3e		ldx	jind	072039
24	41		ldch	str,x	53a039
25	44		stch	ch2	57203f
26	47		ldch	ch1	53203b
27	4a		rmo	a,s	ac04
28	4c		ldch	ch2	532037
29	4f		compr	s,a	a040



29	4f		compr	s,a	a040
30	51		jgt	swapit	372003
31	54		j	loop2	3f2fbb
32	57	swapit	ldx	jind	072020
33	5a		ldb	str	6b2020
34	5d		ldch	ch1	532025
35	60		stch	str,x	57a01a
36	63		ldb	str	6b2017
37	66		subr	t,x	9451
38	68		ldch	ch2	53201b
39	6b		stch	str,x	57a00f
40	6e		ldb	str	6b200c
41	71		j	loop2	3f2f9e
42	74	exit	j	*	3f2010
43	77	iind	word	0	0
44	7a	jind	word	0	0
45	7d	str	byte	C'54321'	3534333231
46	82	len	word	5	0
47	85	ch1	byte	C' '	20
48	86	ch2	byte	C' '	20
49	87		end		

Object file:

```
Hbubble000000000087
T0000001b7500010320712b207933206801000000f206807206503206a1d0001
T00001b1a1f2059a0103b200c032051190001 0f204b3f2fd453a04b572050
T0000351a0320421900010f203c07203953a03957203f53203bac04532037
T00004f19a0403720033f2fbb0720206b202053202557a01a6b20179451
T0000681553201b57a00f6b200c3f2f9e3f20100035343332310
T000085022020
E000000
```

Third sample:

Input:

```
.23456789012345678901 23
PROB2    START   1000
         LDX     INITL
LOOP     LDS     ZERO
         STS     ARRAY,X
         TIX     TEST
         JLT     LOOP
         LDA     NOTFOUND
         LDS     ARRAY-ZERO
         LDT     ARRAY+ZERO
ARRAY    RESW    100
ZERO     WORD    0
INITL    WORD    0
TEST     WORD    100
         END
```

Output:

PASS 1

....

LineNo	Adress	Label	Op-code	Operand
0				
0			.2345678901234567890123	
1	1000	prob2	start	1000
2	1000		ldx	initl
3	1003	loop	lds	zero
4	1006		sts	array,x
5	1009		tix	test
6	100c		jlt	loop
7	100f		lda	notfound
8	1012		lds	array-zero
9	1015		ldt	array+zero
10	1018	array	resw	100
11	1144	zero	word	0
12	1147	initl	word	0
13	114a	test	word	100
14	114d		end	

********************Symbol Table********************

Symbol	Address
array	1018
initl	1147
loop	1003
test	114a
zero	1144

```
                                    PASS 2
                                   .........


ineNo           Adress          Label          Op-code                 Operand          ObjectCode


  0
  0                                      .234567890123456789012345
  1             1000            prob2           start                   1000
  2             1000                            ldx                    initl             072144
  3             1003             loop           lds                     zero              6f213e
  4             1006                            sts                   array,x             7fa00f
  5             1009                            tix                     test              2f213e
  6             100c                            jlt                     loop              3b2ff4
  7             100f                            lda                   notfound
                                ****Error: undefined symbol
  8             1012                            lds                array-zero
                                ****Error: negative address
  9             1015                            ldt                array+zero
                                ****Error: invalid expression
 10             1018            array           resw                    100
 11             1144             zero           word                      0                 0
 12             1147            initl           word                      0                 0
 13             114a             test           word                    100                 0
 14             114d                            end
```

```
                    **********pass2 errors**********

line number : 7 ,error: ****Error: undefined symbol
line number : 8 ,error: ****Error: negative address
line number : 9 ,error: ****Error: invalid expression
```

Object file:

There is no object file because there are errors in pass2.