

Systems Programming

Final Term Project “PHASE I”- SIC/XE Assembler

Team Members:

- *Karim Mohamed Ali* *ID: 32*
- *Mazen Ahmed Zakaria* *ID: 34*
- *Michael Rami Ramsis* *ID: 35*
- *Youssef Abdallah Youssef* *ID: 62*

PROBLEM STATEMENT:

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 1 of the project, it is required to implement Pass1 of the assembler. The output of this phase should be used as input for subsequent phase.

Specifications and FEATURES:

1. A parser that is capable of handling source lines that are instructions, storage declaration, comments, and assembler directives (a directive that is not implemented is ignored with an error message).
2. For instructions, the parser is capable of decoding 2, 3 and 4-byte instructions as follows:
 - a) 2-byte with 1 or 2 symbolic register reference (e.g., TIXR A, ADDR S,A)
 - b) RSUB (ignoring any operand or perhaps issuing a warning)
 - c) 3-byte PC-relative with symbolic operand to include immediate, indirect, and indexed addressing.
 - d) 3-byte absolute with non-symbolic operand to include immediate, indirect, and indexed addressing.
 - e) 4-byte absolute with symbolic or non-symbolic operand to include immediate, indirect, and indexed addressing
3. The parser handles all storage directives (BYTE, WORD, RESW, and RESB).
4. The output of this phase contains:
 - a) The symbol table.
 - b) The source program in a format similar to the listing file described in your text book except that the object code is not generated as shown below.

c) A meaningful error message should be printed below the line in which the error occurred.

5. FREE-FORMATED Assembly language program is supported.

DESIGN:

We designed a class called ListingEntry to hold all the necessary fields to be displayed in the listing file.

Class SymValue to store the address for each label and its length and it can be modified easily for extra fields for pass 2.

Class Parser to parse the instruction into its different fields: label, mnemonic, operand, comment.

Class InstructionValidator to validate the fixed format instruction and it returns the supposed to be correct instruction to compare it with the instruction entered by the user.

MAIN DATA STRUCTURE:

1. Vector of class ListingEntry:

ListingEntry contains all fields required for a row in the listing table.

- Line number
- Address
- Label
- Mnemonic
- Operand
- Comment
- Error flag and Error Message

2. Symbol table (unordered_map)

- Symbol name
- Value

3. Unordered_map of all valid mnemonics, directives and registers.

ALGORITHM DESCRIPTION:

Algorithm pass 1:

```
-----
while (file has lines) begin
    read line
    if (firstWord[0] != '.') then
        split the line into words[]
        if (word[i] start with '.')
            set comment field with this word
        if (words.length == 1) then
            if (word[0] not found in mnemonic map) then
                set error flag (wrong mnemonic)
            else if (! free format)
                check if the line match the stanrard format
                set mnemonic filed
            else
                set mnemonic field
        else if (words.length == 2)
            if (first word match a mnemonic in the map)
                set mnemonic field
                set operand field field with the 2nd word
            else if (second word match a mnemonic)
                set the mnemonic field
                set the label field with the first word
                put label in symtable
                if (label exists)
                    set error flag (duplicate label definition)
            if (! free format)
                check if the line match the standard format
        else if (words.length == 3)
            if (second word match a mnemonic in the map)
                set mnemonic field
                set label field with the firsst word
                put label in symtable
                if (label exists)
                    set error flag (duplicate label definition)
                set operand field with the third word
            else
                set error flag (wrong mnemonic)
            if (! free format)
                check if the line match the stanrard format
        else
            set error flag
    if (mnemonic field == "START") then
        if (operand is not empty) then
            set loc counter with operand
        else
            set loc counter with 0
    else if (mnemonic field == "END")
        set endFlag = true
        if (label field is not empty)
            set error flag (can't have a lable)
    else if (mnemonic == "BASE")
        if there is a label OR there is not an operand
            set error flag
    else if (mnemonic == "NOBASE")
        if there is label OR an operand
            set error flag
    else if (mnemonic == "ORG")
        check that there is not a label and an operand exists
        check that the operand exists in the symTable
```

```

else if (mnemonic == "EQU")
    check that there exist a label and an operand existing in SymTable
else if (mnemonic == "WORD" OR "BYTE" OR "RESW" OR "RESB")
    check there is a label and an operand
    add to loc counter (3 * #words) OR (1 * #bytes)
else if (mnemonic == "RESB")
    check there is not an operand
else
    check the format of the instruction
    add to loc counter 2 or 3 or 4 according to the format
else
    add comment line

end

```

ASSUMPTIONS:

- We assumed that any comment field in the instruction must be preceded by a dot character.

SAMPLE RUNS:

A. Standard Format:

1.

```

.2345678901234567890
prog      start  0000
.
BGN       LDA    #0
RLOOP     TD     DEVF3
          JEQ     RLOOP
          RD     DEVF3
          STCH    CHAR
          COMP    #4
          JEQ     EXIT
          LDA     SIZE
          ADD     #1
          STA     SIZE
          LDA     #0
          J       RLOOP
EXIT      LDA     SIZE
          J       *

.
SIZE      WORD    0
CHAR      RESB    1
DEVF3     BYTE    X'F3'
end

```

2.

line no.	Address	Label	Mnemonic	Operands	Comment
1		.2345678901234567890			
2	0000	PROG	START	0000	
3		.			
4	0000	BGN	LDA	#0	
5	0003	RLOOP	TD	DEVF3	
6	0006		JEQ	RLOOP	
7	0009		RD	DEVF3	
8	000c		STCH	CHAR	
9	000f		COMP	#4	
10	0012		JEQ	EXIT	
11	0015		LDA	SIZE	
12	0018		ADD	#1	
13	001b		STA	SIZE	
14	001e		LDA	#0	
15	0021		J	RLOOP	
16	0024	EXIT	LDA	SIZE	
17	0027		J	*	
18		.			
19	002a	SIZE	WORD	0	
20	002d	CHAR	RESB	1	
21	002e	DEVF3	BYTE	X'F3'	
22	002f		END		
*** P A S S 1 E N D E D S U C C E S S F U L L Y ***					
S Y M B O L T A B L E					

NAME		VALUE			

CHAR		002d			
EXIT		0024			
SIZE		002a			
RLOOP		0003			
BGN		0000			
DEVF3		002e			
PROG		0000			

.2345678901234567890			line no.	Address	Label	Mnemonic	Operands	Comment
prog	start	0000	1		.2345678901234567890			
.			2	0000	PROG	START	0000	
BGN	LDA	#0	3		.			
	LDS	#0	4	0000	BGN	LDA	#0	
	LDX	#0	5	0003		LDS	#0	
RLOOP	TD	DEVF3	6	0006		LDX	#0	
	JEQ	RLOOP	7	0009	RLOOP	TD	DEVF3	
	RD	DEVF3	8	000c		JEQ	RLOOP	
	STCH	STRING,X	9	000f		RD	DEVF3	
	STCH	CHAR	10	0012		STCH	STRING,X	
	RMO	A,S	11	0015		STCH	CHAR	
	LDA	LEN	12	0018		RMO	A,S	
	ADD	#1	13	001a		LDA	LEN	
	STA	LEN	14	001d		ADD	#1	
	LDX	LEN	15	0020		STA	LEN	
	LDCH	CHAR	16	0023		LDX	LEN	
	COMP	#4	17	0026		LDCH	CHAR	
	JEQ	ENDREAD	18	0029		COMP	#4	
	J	RLOOP	19	002c		JEQ	ENDREAD	
ENDREAD	LDA	LEN	20	002f		J	RLOOP	
	SUB	#2	21	0032	ENDREAD	LDA	LEN	
	STA	LEN	22	0035		SUB	#2	
	LDX	LEN	23	0038		STA	LEN	
	LDCH	STRING,X	24	003b		LDX	LEN	
	WD	DEV05	25	003e	WLOOP	LDA	#0	
	RMO	X,A	26	0041		TD	DEV05	
	COMP	#0	27	0044		JEQ	WLOOP	
	SUB	#1	28	0047		LDCH	STRING,X	
	RMO	A,X	29	004a		WD	DEV05	
	JEQ	EXIT	30	004d		RMO	X,A	
	J	WLOOP	31	004f		COMP	#0	
EXIT	J	*	32	0052		SUB	#1	
.			33	0055		RMO	A,X	
DEVF3	BYTE	X'F3'	34	0057		JEQ	EXIT	
DEV05	BYTE	X'05'	35	005a		J	WLOOP	
STRING	RESB	100	36	005d	EXIT	J	*	
LEN	WORD	0						
CHAR	RESB	1						
	END	BGN						

Free Format:

1.

```
TERMPROJ START 3A0  
.THIS IS A COMMENT LINE  
LBL1 BYTE C'ABCDEF'  
LBL2 RESB 4  
LBL2 RESW 1  
TOP LDA ZERO  
LDX #INDEX  
END
```

line no.	Address	Label	Mnemonic	Operands	Comment
1	03a0	TERMPROJ	START	3A0	
2					.THIS IS A COMMENT LINE
3	03a0	LBL1	BYTE	C'ABCDEF'	
4	03a6	LBL2	RESB	4	
5	03aa	LBL2	RESW	1	
		*** ERROR: duplicate label definition ***			
6	03ad	TOP	LDA	ZERO	
7	03b0		LDX	#INDEX	
8	03b3		END		
*** I N C O M P L E T E A S S E M B L Y ***					
S Y M B O L T A B L E					

NAME		VALUE			

LBL2		03a6			
LBL1		03a0			
TOP		03ad			
TERMPROJ		0000			

2.

```
prog start 0000  
.  
bgn LDX #0  
LOOP LDCH BLANK  
STCH STRING,X  
TIX #100  
JLT LOOP  
J *  
.  
STRING RESB 100  
BLANK BYTE C'*'  
end bgn
```

line no.	Address	Label	Mnemonic	Operands	Comment
1	0000	PROG	START	0000	
2					.
3	0000	BGN	LDX	#0	
4	0003	LOOP	LDCH	BLANK	
5	0006		STCH	STRING,X	
6	0009		TIX	#100	
7	000c		JLT	LOOP	
8	000f		J	*	
9					.
10	0012	STRING	RESB	100	
11	0076	BLANK	BYTE	C'*	
12	0077		END	BGN	
*** P A S S 1 E N D E D S U C C E S S F U L L Y ***					
S Y M B O L T A B L E					

NAME		VALUE			

BLANK		0076			
STRING		0012			
LOOP		0003			
BGN		0000			
PROG		0000			