Name: Priyanshu Behera

Enrollment Number: 21114077

Subbatch: O3

SICXE Assembler

Q1: How to start the assembler?

Ans: Download the zip file, unzip it.

Give the input to the assembler in the file:input.txt. If not created then create a one.

Then type the following command line on the terminal:

“g++ pass2.cpp -o assembler.exe”

And then type the following to get the assembling done:

./assembler.exe

After this it will display

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS F:\21114077_Priyanshu_Behera_SIC-XE-Assembler> g++ pass2.cpp -o assembler.exe
PS F:\21114077_Priyanshu_Behera_SIC-XE-Assembler> ./assembler.exe
****Input file and executable(assembler.out) should be in same folder****

Enter name of input file:
```

Enter the name of the input file. Note the executable created should be in the same folder as the input file.

It will create the necessary output files:

```
intermediate_input.txt
listing_input.txt
object_input.txt
pass1.cpp
pass2.cpp
```

Note:- The various tables generated are stored in tables_input.txt

Given below is the test input(sample input) as described by Sir for which I checked my code:

input.txt

```
SUM START 0
FIRST LDX #0
    LDA #0
    +LDB #TABLE2
    BASE TABLE2
LOOP ADD TABLE,X
    ADD TABLE2,X
    TIX COUNT
    JLT LOOP
    +STA TOTAL
    RSUB
COUNT RESW 1
TABLE RESW 2000
TABLE2 RESW 2000
TOTAL RESW 1
    END FIRST
```

And has the following intermediate file :

intermediate_input.txt

```
Line    Address Label   OPCODE  OPERAND Comment
5   00000   0   SUM START   0
10  00000   0   FIRST   LDX #0
15  00003   0       LDA #0
20  00006   0       +LDB    #TABLE2
25  0000A   0       BASE    TABLE2
30  0000A   0   LOOP    ADD TABLE,X
35  0000D   0       ADD TABLE2,X
40  00010   0       TIX COUNT
45  00013   0       JLT LOOP
50  00016   0       +STA    TOTAL
55  0001A   0       RSUB
60  0001D   0   COUNT   RESW    1
65  00020   0   TABLE   RESW    2000
70  01790   0   TABLE2  RESW    2000
75  02F00   0   TOTAL   RESW    1
80  02F03           END FIRST
```

Below is screenshot for the lisiting-file:

```
Line    Address Label   OPCODE  OPERAND ObjectCode  Comment
10  00003   0       LDA #0  010000
15  00006   0       +LDB    #TABLE2 69101790
20  0000A   0       BASE    TABLE2
25  0000A   0   LOOP    ADD TABLE,X 1BA013
30  0000D   0       ADD TABLE2,X    1BC000
35  00010   0       TIX COUNT   2F200A
40  00013   0       JLT LOOP    3B2FF4
45  00016   0       +STA    TOTAL   0F102F00
50  0001A   0       RSUB        4F0000
55  0001D   0   COUNT   RESW    1
60  00020   0   TABLE   RESW    2000
65  01790   0   TABLE2  RESW    2000
70  02F00   0   TOTAL   RESW    1
75  02F03           END FIRST
```

Here is the object file for the same:

object_input.txt

```
H^SUM   ^000000^002F03
T^000000^1D^05000001000069101790 1BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000
```

Here is the next input sic-xe code:

```
COPY    START   0
        EXTDEF  BUFFER,BUFEND,LENGTH
        EXTREF  RDREC,WRREC
FIRST   STL     RETADR
CLOOP  +JSUB    RDREC
        LDA     LENGTH
        COMP   #0
        JEQ     ENDFIL
       +JSUB    WRREC
        J       CLOOP
ENDFIL  LDA    =C'EOF'
        STA     BUFFER
        LDA    #3
        STA     LENGTH
       +JSUB    WRREC
        J      @RETADR
RETADR  RESW    1
LENGTH  RESW    1
        LTORG
BUFFER  RESB    4096
BUFEND  EQU     *
MAXLEN  EQU     BUFEND-BUFFER
RDREC   CSECT
```

```
.
.        SUBROUTINE TO READ RECORD INTO BUFFER
.
         EXTREF   BUFFER,LENGTH,BUFFEND
         CLEAR    X
         CLEAR    A
         CLEAR    S
         LDT      MAXLEN
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'
MAXLEN   WORD     BUFEND-BUFFER
.....
WRREC    CSECT
```

```
.
.        SUBROUTINE TO WRITE RECORD FROM BUFFER
.
         EXTREF    LENGTH,BUFFER
         CLEAR     X
        +LDT       LENGTH
WLOOP    TD       =X'05'
         JEQ       WLOOP
        +LDCH      BUFFER,X
         WD       =X'05'
         TIXR      T
         JLT       WLOOP
         RSUB
         END       FIRST
```

Below is my intermediate file:

```
Line    Address Label   OPCODE  OPERAND Comment
5   00000   0   COPY    START   0
10              EXTDEF  BUFFER,BUFEND,LENGTH
15              EXTREF  RDREC,WRREC
20  00000   0   FIRST   STL RETADR
25  00003   0   CLOOP   +JSUB   RDREC
30  00007   0       LDA LENGTH
35  0000A   0       COMP    #0
40  0000D   0       JEQ ENDFIL
45  00010   0       +JSUB   WRREC
50  00014   0       J   CLOOP
55  00017   0   ENDFIL  LDA =C'EOF'
60  0001A   0       STA BUFFER
65  0001D   0       LDA #3
70  00020   0       STA LENGTH
75  00023   0       +JSUB   WRREC
80  00027   0       J   @RETADR
85  0002A   0   RETADR  RESW    1
90  0002D   0   LENGTH  RESW    1
95  00030   0       LTORG
100 00030   0   *   =C'EOF'
105 00033   0   BUFFER  RESB    4096
110 01033   0   BUFEND  EQU *
115 01000       MAXLEN  EQU BUFEND-BUFFER
120 00000   0   RDREC   CSECT
125
```

```
125 .
130 .        SUBROUTINE TO READ RECORD INTO BUFFER
135 .
140              EXTREF  BUFFER,LENGTH,BUFFEND
145 00000   0        CLEAR   X
150 00002   0        CLEAR   A
155 00004   0        CLEAR   S
160 00006   0        LDT MAXLEN
165 00009   0    RLOOP   TD  INPUT
170 0000C   0        JEQ RLOOP
175 0000F   0        RD  INPUT
180 00012   0        COMPR   A,S
185 00014   0        JEQ EXIT
190 00017   0        +STCH   BUFFER,X
195 0001B   0        TIXR    T
200 0001D   0        JLT RLOOP
205 00020   0    EXIT    +STX    LENGTH
210 00024   0        RSUB
215 00027   0    INPUT   BYTE    X'F1'
220 00028   0    MAXLEN  WORD    BUFEND-BUFFER
225 .....
230 00000   0    WRREC   CSECT
235
```

```
235 .
240 .        SUBROUTINE TO WRITE RECORD FROM BUFFER
245 .
250              EXTREF  LENGTH,BUFFER
255 00000   0        CLEAR   X
260 00002   0        +LDT    LENGTH
265 00006   0    WLOOP   TD  =X'05'
270 00009   0        JEQ WLOOP
275 0000C   0        +LDCH   BUFFER,X
280 00010   0        WD  =X'05'
285 00013   0        TIXR    T
290 00015   0        JLT WLOOP
295 00018   0        RSUB
300 0001B            END FIRST
305 0001B   0    *   =X'05'
```

Here is the listing-file for the same:

```
Line    Address Label   OPCODE  OPERAND ObjectCode  Comment
5   00000   0   COPY    START   0
10              EXTDEF  BUFFER,BUFEND,LENGTH
15              EXTREF  RDREC,WRREC
20  00000   0   FIRST   STL RETADR  172027
25  00003   0   CLOOP   +JSUB   RDREC   4B100000
30  00007   0       LDA LENGTH  032023
35  0000A   0       COMP    #0  290000
40  0000D   0       JEQ ENDFIL  332007
45  00010   0       +JSUB   WRREC   4B100000
50  00014   0       J   CLOOP   3F2FEC
55  00017   0   ENDFIL  LDA =C'EOF' 032016
60  0001A   0       STA BUFFER  0F2016
65  0001D   0       LDA #3  010003
70  00020   0       STA LENGTH  0F200A
75  00023   0       +JSUB   WRREC   4B100000
80  00027   0       J   @RETADR 3E2000
85  0002A   0   RETADR  RESW    1
90  0002D   0   LENGTH  RESW    1
95  00030   0       LTORG
100 00030   0   *   =C'EOF'     454F46
105 00033   0   BUFFER  RESB    4096
110 01033   0   BUFEND  EQU *
115 01000       MAXLEN  EQU BUFEND-BUFFER
120 00000   0   RDREC   CSECT
125
```

```
130 .       SUBROUTINE TO READ RECORD INTO BUFFER
135 .
140             EXTREF  BUFFER,LENGTH,BUFFEND
145 00000   0       CLEAR   X   B410
150 00002   0       CLEAR   A   B400
155 00004   0       CLEAR   S   B440
160 00006   0       LDT MAXLEN  770000
165 00009   0   RLOOP   TD  INPUT   E3201B
170 0000C   0       JEQ RLOOP   332FFA
175 0000F   0       RD  INPUT   DB2015
180 00012   0       COMPR   A,S A004
185 00014   0       JEQ EXIT    332009
190 00017   0       +STCH   BUFFER,X    57100000
195 0001B   0       TIXR    T   B850
200 0001D   0       JLT RLOOP   3B2FE9
205 00020   0   EXIT    +STX    LENGTH  13100000
210 00024   0       RSUB        4F0000
215 00027   0   INPUT   BYTE    X'F1'   F1
220 00028   0   MAXLEN  WORD    BUFEND-BUFFER   000000
225 .....
230 00000   0   WRREC   CSECT
235 .
```

```
235 .
240 .       SUBROUTINE TO WRITE RECORD FROM BUFFER
245 .
250             EXTREF  LENGTH,BUFFER
255 00000   0       CLEAR   X   B410
260 00002   0       +LDT    LENGTH  77100000
265 00006   0   WLOOP   TD  =X'05'  E32012
270 00009   0       JEQ WLOOP   332FFA
275 0000C   0       +LDCH   BUFFER,X    53100000
280 00010   0       WD  =X'05'  DF2008
285 00013   0       TIXR    T   B850
290 00015   0       JLT WLOOP   3B2FEE
295 00018   0       RSUB        4F0000
300 0001B           END FIRST
305 0001B   0*  =X'05'      05
```

Here is the output for object-file:

```
H^COPY  ^000000^001033
D^BUFFER00033BUFEND01033LENGTH0002D
R^RDREC WRREC
T^000000^1D^1720274B1000000320232900003320074B1000003F2FEC0320160F2016
T^00001D^0D^0100030F200A4B1000003E2000
T^000030^03^454F46
M^000004^05+RDREC
M^000011^05+WRREC
M^000024^05+WRREC
E^000000

*******************object program for RDREC **************

H^RDREC ^000000^00002B
R^BUFFERLENGTHBUFFEN
T^000000^1D^B410B400B440770000E3201B332FFADB2015A0043320095710000B850
T^00001D^0E^3B2FE9131000004F0000F1000000
M^000018^05+BUFFER
M^000021^05+LENGTH
E
```

```
*******************object program for WRREC **************

H^WRREC ^000000^00001B
R^LENGTHBUFFER
T^000000^1C^B41077100000E32012332FFA53100000DF2008B8503B2FEE4F000005
M^000003^05+LENGTH
M^00000D^05+BUFFER
E
```

The program will take the input from the input.txt file as specified above and write both the intermediate file to the intermediate_input.txt. and listing file onto listing_input.txt and produces the corresponding object-program in object_input.txt.

Any errors generated will be written onto the error_input.txt.

Here are the errors that I got for the following:

```
FIRST LDX #0
      LDA #50000
      +LDB #TABLE2
      BASE TABLE2
LOOP  ADD TABLE,X
      ADD TABLE2,X
      TIX COUNT
```

And since it has a very large value which can't be accommodated in format 3, it displayed the following:

```
************PASS1************


************PASS2************
Line: 15 Immediate value exceeds format limit
```

```
SUM START 0
FIRST LDX #0
    LDA #0
    +LDB #TABLE2
    BASE TABLE2
LOOP ADD TABLE,X
    ADD TABLE2,X
    TIX COUNTER
    JLT LOOP
    +STA TOTAL
    RSUB
COUNT RESW 1
TABLE RESW 2000
TABLE2 RESW 2000
TOTAL RESW 1
    END FIRST
```

Since there is no counter defined in the SIC-XE code it gave the following error:

```
************PASS1************


************PASS2************
Line 40 : Symbol doesn't exists. Found COUNTER
```

Assembler Design:

pass1.cpp:

1. In pass1 ,parsing is done, then we are calculating the addresses for all labels and setting the attributes for a given control-section.

2. In our code we are entering the symbols of corresponding control sections into the symbol table here. The intermediate file is also being written here.

3. Literals are allocated space in pass1 only and then they are pushed to LITTAB.

pass2.cpp:

1.. The intermediate file is read here and then the object codes are being created.

2. Various errors corresponding to if symbols not found, if the displacement filed is very large to be accommodated for format 3,etc are writtern here.

3. The object program or the various records are written here.

tables.cpp:

1.Here we define the various structures required for storing the necessary information.

```
struct struct_csect{
    string name ;
    string LOCCTR ;
    int section_number ;
    int length ;
    map<string,struct_extdef> EXTDEF_TAB ;
    map<string,struct_extref> EXTREF_TAB ;
    struct_csect(){
        name="DEFAULT" ;
        LOCCTR="0" ;
        section_number=0 ;
        length=0 ;
    }
};
```

2.Also the opcode table is stored here.

```
void load_OPTAB(){
  OPTAB["ADD"].opcode="18";
  OPTAB["ADD"].format=3;
  OPTAB["ADD"].exists='y';

  OPTAB["ADDF"].opcode="58";
  OPTAB["ADDF"].format=3;
  OPTAB["ADDF"].exists='y';

  OPTAB["ADDR"].opcode="90";
  OPTAB["ADDR"].format=2;
  OPTAB["ADDR"].exists='y';

  OPTAB["AND"].opcode="40";
  OPTAB["AND"].format=3;
  OPTAB["AND"].exists='y';

  OPTAB["CLEAR"].opcode="B4";
  OPTAB["CLEAR"].format=2;
  OPTAB["CLEAR"].exists='y';
```

Utility.cpp:

Here various helper function such as checking white space,hexadecimal string to int, string to decimal are implemented here which helps reuse and modularity of code.

```
int String_to_decimal(string str)
{
  int value;
  stringstream(str) >> value;
  return value;
}
```

```
int stringHexToInt(string x){
  return stoul(x,nullptr,16);
}
```

```
bool checkWhiteSpace(char x){
  if(x==' ' || x=='\t'){
    return true;
  }
  return false;
}
```