PISE ASHUTOSH KALIDAS

21114073

O3, B.Tech. CSE (2Y)

SIC-Xe Assembler

Q1. How to run the assembler?

Ans:-

1. Download the zip-file, unzip it, and move into the folder.
2. Run pass2.cpp using `g++ pass2.cpp`
3. Enter the name of the input file (input1.txt or input2.txt)

Q2. Where is the intermediate file generated?

Ans:- A file named intermediate_input.txt will be generated.

```
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
```

```
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 .
```

```
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 .
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'
```

Q3. Where is the listing file generated?

Ans:- A file named listing_input.txt will be generated.

```
Line    Address Label   OPCODE  OPERAND ObjectCode  Comment
5   00000   0   SUM START   0
10  00000   0   FIRST   LDX #0  050000
15  00003   0       LDA #0  010000
20  00006   0       +LDB    #TABLE2 69101790
25  0000A   0       BASE    TABLE2
30  0000A   0   LOOP    ADD TABLE,X 1BA013
35  0000D   0       ADD TABLE2,X    1BC000
40  00010   0       TIX COUNT   2F200A
45  00013   0       JLT LOOP    3B2FF4
50  00016   0       +STA    TOTAL   0F102F00
55  0001A   0       RSUB        4F0000
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
```

```
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 .
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
```

Q4. Where is the final object program generated?

Ans:- A file named object_input.txt will be generated.

```
H^SUM    ^000000^002F03
T^000000^1D^0500000100006910179O1BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000
```

```
H^COPY   ^000000^001033
D^BUFFER00033BUFEND01033LENGTH0002D
R^RDREC WRREC
T^000000^1D^1720274B10000003202329000033200074B1000003F2FEC0320160F2016
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^B410B400B440770000E3201B332FFADB2015A00433200957100000B850
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
```

Q5. How are the errors shown?

Ans:- A file named error_input.txt will be generated. Errors in pass1 and pass2 will be shown separately.

Eg. In the following program, LDA #5000 goes out of bound. Also, in +LDB #TABLE3, TABLE3 is an unrecognized symbol. Both the errors are shown correctly.

```
SUM     START   0
FIRST   LDX     #0
        LDA     #5000
        +LDB    #TABLE3
        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
```

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


**********PASS2**********
Line: 15 Immediate value exceeds format limit
Line 20 : Symbol doesn't exists. Found TABLE3
```

Assembler Design:-

In pass1 following tasks are done:-

1. Addresses are assigned to each instruction.
2. Symbols and labels are entered into symtabs.
3. Literals are entered into lit-tabs.

In pass2 following tasks are done:-

1. Object code is calculated for each instruction.
2. Finally, object program for the entire program is generated.

In tables.cpp, different useful structs are declared, and format and opcode are assigned for each instruction (Instruction set is defined).

```
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 ;
    }
};
struct struct_opcode{
    string opcode;
    int format;
    char exists;
    struct_opcode(){
      opcode="undefined";
      format=0;
      exists='n';
    }
};
```

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

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

OPTAB["COMP"].opcode="28";
OPTAB["COMP"].format=3;
OPTAB["COMP"].exists='y';

OPTAB["COMPF"].opcode="88";
OPTAB["COMPF"].format=3;
OPTAB["COMPF"].exists='y';

OPTAB["COMPR"].opcode="A0";
OPTAB["COMPR"].format=2;
OPTAB["COMPR"].exists='y';

OPTAB["DIV"].opcode="24";
OPTAB["DIV"].format=3;
OPTAB["DIV"].exists='y';
```

In utility.cpp, different useful I/O functions, string-processing functions, and string to int/hex functions are defined, which are used in pass1, pass2 while assembling.

```
void readFirstNonWhiteSpace(string line,int& index,bool& status,string& data,bool readTillEnd=false){
  data = "";
  status = true;
  if(readTillEnd){
    data = line.substr(index,line.length() - index);
    if(data==""){
      status = false;
    }
    return;
  }
  while(index<line.length()&&!checkWhiteSpace(line[index])){//If no whitespace then data
    data += line[index];
    index++;
  }

  if(data==""){
    status = false;
  }

  while(index<line.length()&&checkWhiteSpace(line[index])){//Increase index to pass all whitespace
    index++;
  }
}
```

```
int String_to_decimal(string str)
{
  int value;
  stringstream(str) >> value;
  return value;
}
string getString(char c){
  string s(1,c) ;
  return s ;
}

string intToStringHex(int x,int fill = 5){
  stringstream s;
  s << setfill('0') << setw(fill) << hex << x;
  string temp = s.str();
  temp = temp.substr(temp.length()-fill,fill);
  transform(temp.begin(), temp.end(),temp.begin(),::toupper);
  return temp;
}
```