Roll No									

National Institute of Technology Goa

Programme Name: **B.Tech**

Online End Semester Examinations, May-2021

Course Name: Systems Programming | Course Code: CS251

Date: **13/05/2021** | Time: 9:30 AM-12:30 PM

Duration: **3 Hours** | Max. Marks: **100**

ANSWER ALL QUESTIONS

NOTE: Opcodes for a selected set of instructions are given at the end of this paper. Same codes must be used.

1.
   a. Explain the various instruction formats supported in SIC/XE giving 2 examples for each.
   b. Write short notes on one-pass assembler and a multi-pass assembler.

[6+4=10 marks]

2.
   a. What are the data structures used in a linking loader? Explain.
   b. Compare and contrast a linking loader and a linkage editor.
   c. Write a short note on dynamic linking.

[3+4+3=10 marks]

3.
   a. What is the limitation of a standard 2-pass macro processor when a macro is defined within another macro? How is it addressed?
   b. Briefly explain the important data structures used by a macro processor

[4+6=10 marks]

4. Consider a macro defined as follows:

```
RDBUFF      MACRO  &INDEV, &BUFADR, &RECLTH, &EOR, &MAXLTH
&EORCT      SET    %NITEMS(&EOR)
            CLEAR  X
            CLEAR  A
            IF     (&MAXLTH EQ ' ')
            +LDT   #4096
            ELSE
            +LDT   #&MAXLTH
            ENDIF
$LOOP       TD     =X`&INDEV`
            JEQ    $LOOP
            RD     =X`&INDEV`
&CTR        SET    1
            WHILE  (&CTR LE &EORCT)
            COMP   =X`0000&EOR[&CTR]`
            JEQ    $EXIT
&CTR        SET    &CTR+1
            ENDW
            STCH   &BUFADR,X
            TIXR   T
            JLT    $LOOP
$EXIT       STX    &RECLTH
            MEND
```

Expand the following macro invocation statements for the macro definition given above

a. `RDBUFF      F1, BUF, LENGTH, ,`

b. `RDBUFFF2, BUFFER, LEN, (00, 05, 03, 04)`

[5+5=10 marks]

5. Consider the following assembly language program written for SIC/XE.

```
ARRMP       START       0
            EXTDEF      ALPHA, MAXLEN
            EXTREF      SUMARR, FMAXA
            +JSUB       SUMARR
            +JSUB       FMAXA
            +JSUB       FMINA
            +JSUB       MKZERA
            RSUB
ALPHA       RESW        100
AEND        EQU         *
MAXLEN      EQU         AEND-ALPHA


SUMARR      CSECT
            EXTREF      ALPHA, MAXLEN
            CLEAR       X
            CLEAR       A
            LDS         THREE
            +LDT        MAXLEN
            LDA         #0
            STA         SUM
ADDLP       +ADD        ALPHA,X
            ADDR        S,X
            COMPR       X,T
            JLT         ADDLP
            STA         SUM
            RSUB
THREE       WORD        3
SUM         RESW        1


FMAXA       CSECT
            EXTREF      ALPHA, MAXLEN
            CLEAR       X
            CLEAR       A
            LDS         THREE
            STX         MAXVAL
            +LDT        MAXLEN
MAXLP       +LDA        ALPHA,X
            COMP        MAXVAL
            JLT         INCR
            STA         MAXVAL
INCR        ADDR        S,X
            COMPR       X,T
            JLT         MAXLP
            RSUB
THREE       WORD        3
MAXVAL      RESW        1
```

Transfer/copy the program from the question paper onto your answer script. Against each line of the assembly language program so copied, write the corresponding location address and object code. Write the object programs that are generated at the end of the assembly process.

[20 marks]

6. Consider the three (separately assembled) programs (PROGA, PROGB and PROGC) given below. Each of the programs consists of a single control section. Each program contains a list of items (LISTA, LISTB and LISTC) and the ends of these lists are marked by the labels ENDA, ENDB and ENDC. Each program consists of same set of references to these symbols. Note that all portions of the programs which are not involved in relocation and linking are omitted.

```
0000  PROGA START 0
            EXTDEF LISTA, ENDA
            EXTREF LISTB, ENDB, LISTC, ENDC
            .
            .
            .
0010  REF1  +LDA  LISTB-LISTA
0014  REF2  +LDT  LISTB+7
0018  REF3  LDX   #ENDA-LISTA
001B  REF4  LDS   LISTA
            .
            .
            .
0030  LISTA EQU   *
            .
            .
            .
0047  ENDA  EQU   *
0047  REF5  WORD  LISTC
004A  REF6  WORD  LISTB-3
004D  REF7  WORD  LISTA+LISTB
0050  REF8  WORD  ENDC-LISTC-100
0053  REF9  WORD  LISTA-LISTB-ENDA+ENDB
      END
```

```
0000  PROGB START 0
            EXTDEF LISTB, ENDB
            EXTREF LISTA, ENDA, LISTC, ENDC
            .
            .
            .
0023  REF1  +LDA  LISTB-LISTA
0027  REF2  LDT   LISTB+7
002A  REF3  +LDX  #ENDA-LISTA
002E  REF4  +LDS  LISTA
            .
            .
            .
0038  LISTB EQU   *
            .
            .
            .
0044  ENDB  EQU   *
0044  REF5  WORD  LISTC
0047  REF6  WORD  LISTB-3
004A  REF7  WORD  LISTA+LISTB
004D  REF8  WORD  ENDC-LISTC-100
0050  REF9  WORD  LISTA-LISTB-ENDA+ENDB
      END
```

```
0000  PROGC START 0
            EXTDEF LISTC, ENDC
            EXTREF LISTA, ENDA, LISTB, ENDB
            .
            .
            .
0028  REF1  +LDA  LISTB-LISTA
002C  REF2  +LDT  LISTB+7
0030  REF3  +LDX  #ENDA-LISTA
0034  REF4  +LDS  LISTA
            .
            .
            .
004F  LISTC EQU   *
            .
            .
005D  ENDC  EQU   *
005D  REF5  WORD  LISTC
0060  REF6  WORD  LISTB-3
0063  REF7  WORD  LISTA+LISTB
0066  REF8  WORD  ENDC-LISTC-100
0069  REF9  WORD  LISTA-LISTB-ENDA+ENDB
      END
```

a. Transfer/copy the programs from the question paper onto your answer script. Against each line of the assembly language program so copied, write the corresponding object code.
b. Write the object programs that are generated at the end of the assembly process.
c. Assume PROGA to be loaded starting from 3000 and PROGC and PROGB are to immediately follow PROGA. Show how the three programs appear in memory after loading and linking.

[15+10+15=40 marks]

**********ALL THE BEST**********

SIC/XE Instruction Set

Op codes in blue are SIC/XE only instructions
Op codes in red are not implemented by the simulator

Notes: P=privileged, C=CC set (<,=,>), F=floating point
See Appendix A of *System Software* by Beck for information on instruction formats and addressing modes.

Mnemonic	Format	Opcode	Effect	Notes
ADD m	3/4	18	A ← (A) + (m..m+2)	
ADDF m	3/4	58	F ← (F) + (m..m+5)	F
ADDR r1,r2	2	90	r2 ← (r2) + (r1)	
AND m	3/4	40	A ← (A) & (m..m+2)	
CLEAR r1	2	4	r1 ← 0	
COMP m	3/4	28	A : (m..m+2)	C
COMPF m	3/4	88	F : (m..m+5)	CF
COMPR r1,r2	2	A0	(r1) : (r2)	C
DIV m	3/4	24	A : (A) / (m..m+2)	
DIVF m	3/4	64	F : (F) / (m..m+5)	F
DIVR r1,r2	2	9C	(r2) ← (r2) / (r1)	
FIX	1	C4	A ← (F) [convert to integer]	
FLOAT	1	C0	F ← (A) [convert to floating]	F
HIO	1	F4	Halt I/O channel number (A)	P
J m	3/4	3C	PC ← m	
JEQ m	3/4	30	PC ← m if CC set to =	
JGT m	3/4	34	PC ← m if CC set to >	
JLT m	3/4	38	PC ← m if CC set to <	
JSUB m	3/4	48	L ← (PC); PC ← m<	
LDA m	3/4	00	A ← (m..m+2)	
LDB m	3/4	68	B ← (m..m+2)	
LDCH m	3/4	50	A [rightmost byte] ← (m)	
LDF m	3/4	70	F ← (m..m+5)	F
LDL m	3/4	08	L ← (m..m+2)	
LDS m	3/4	6C	S ← (m..m+2)	
LDT m	3/4	74	T ← (m..m+2)	
LDX m	3/4	04	X ← (m..m+2)	
LPS m	3/4	D0	Load processor status from information beginning at address m (see Section 6.2.1) 6.2.1)	P
MUL m	3/4	20	A ← (A) * (m..m+2)	
MULF m	3/4	60	F ← (F) * (m..m+5)	
MULR r1,r2	2	98	r2 ← (r2) * (r1)	
NORM	1	C8	F ← (F) [normalized]	F
OR m	3/4	44	A ← (A) | (m..m+2)	
RD m	3/4	D8	A [rightmost byte] ← data from device specified by (m)	P
RMO r1,r2	2	AC	r2 ← (r1)	
RSUB	3/4	4C	PC ← (L)	
SHIFTL r1,n	2	A4	r1 ← (r1); left circular shift n bits. [*for assembled instruction, r2 is n-1*]	
SHIFTR r1,n	2	A8	r1 ← (r1); right shift n bits with vacated bit positions set equal to leftmost bit of (r1) [*for assembled instruction, r2 is n-1*]	
SIO	1	F0	Start I/O channel number (A); address of channel program is given by (S)	P
SSK m	3/4	EC	Protection key for address m ← (A) (see Section 6.2.4)	P
STA m	3/4	0C	m..m+2 ← (A)	
STB m	3/4	78	m..m+2 ← (B)	
STCH m	3/4	54	m ← (A) [rightmost byte]	
STF m	3/4	80	m..m+5 ← (F)	F
STI m	3/4	D4	Interval timer value ← (m..m+2) (see Section 6.2.1)	P

STL m	3/4	14	m..m+2 ← (L)	
STS m	3/4	7C	m..m+2 ← (S)	
STSW m	3/4	E8	m..m+2 ← (SW)	P
STT m	3/4	84	m..m+2 ← (T)	
STX m	3/4	10	m..m+2 ← (X)	
SUB m	3/4	1C	A ← (A) - (m..m+2)	
SUBF m	3/4	5C	F ← (F) - (m..m+5)	F
SUBR r1,r2	2	94	r2 ← (r2) - (r1)	
SVC n	2	B0	Generate SVC interrupt. {*for assembled instruction, r1 is n*]	
TD m	3/4	E0	Test device specified by (m)	PC
TIO	1	F8	Test I/O channel number (A)	PC
TIX m	3/4	2C	X ← (X) + 1; (X) : (m..m+2)	C
TIXR r1	2	B8	X ← (X) + 1; (X) : (r1)	C
WD m	3/4	DC	Device specified by (m) ← (A) [rightmost byte to device specified by m]	P