Mobile phones are banned

M S Ramaiah Institute of Technology

Department of Information Science and Engineering

Term:	31/01/2019 to 22/05/2019	**Course Code:**	IS61
Course:	System Software	**Semester:**	6
CIE:	Test I	**Max Marks:**	30
Date:	11/03/19	**Time:**	9 :30 –10 :30 AM

Portions for Test: Lecture Nos. from 1 to 17 as per lesson plan

Instructions to Candidates: Answer any Two out of Three questions.

Sl No	Questions	Marks	B.L	CO
1a)	Explain the instruction formats of SIC/XE machine.	4	U	CO1
1b)	Suppose that ALPHA is an array of 100 words, write a sequence of instructions for SIC/XE to set all 100 elements of the array to 0.	6	C	CO1
1c)	Given the following SIC program, write the object code. RDREC is at location 203D and WRREC is at location 2062.	5	A	CO2

```
COPY     START   0000        Instruction   Opcode
CLOOP    JSUB    RDREC       CLEAR         B4
         LDA     LENGTH      LDCH          50
         COMP    ZERO        RMO           AC
         JEQ     ENDFIL      LDT           74
         JSUB    WRREC       TD            E0
         J       CLOOP       JEQ           30
ENDFIL   LDA     EOF         RD            D8
         STA     BUFFER,X    COMPR         A0
         LDA     THREE       STCH          54
         STA     LENGTH,X    TIXR          B8
         JSUB    WRREC       JLT           38
         LDL     RETADR      STX           10
EOF      BYTE    C'KLNC'     STL           14
THREE    WORD    3           JSUB          48
ZERO     WORD    0           LDA           00
RETADR   RESW    2           COMP          28
LENGTH   RESW    6           J             30
BUFFER   RESB    4096        STA           0C
FIRST    STL     RETADR      LDL           08
         RSUB                RSUB          4C
```

2a)	Illustrate the role of the flag bits n ,i, x in addressing modes of Format 3 and 4 of SIC/XE instructions.	4	U	CO1
2b)	Generate the Object code for the following SIC/XE program (LDA-00,STA-0C,LDX-04, ADD-18, TIX-2C, JLT-38)	8	A	CO2

```
Q1      START   0
FIRST   +LDA    VAL
        STA     BLOCK
        LDX     #0
LOOP    ADD     BUF
        TIX     #5
        JLT     LOOP
        +STA    VAL
BLOCK   RESW    1
BUF     WORD    5
VAL     WORD    13
        END     FIRST
```

Scanned with CamScanner

2c)	Suppose memory addresses 21, 33 and 2700 (all decimal) of a SIC/XE machine contain the values 2400, 2700 and 33 (all decimal) respectively.If (PC)=0000 and the A register (i.e., the accumulator) contains the value −17 (decimal) when the machine executes the 3-byte instruction whose hexadecimal representation is 0E2021. Show the contents of the memory locations 21, 33 and 2700 after the execution of the instruction. Your answers must be in decimal. Opcode of STA is 0C. Show the working.	3	A	CO1
3a)	Explain the addressing modes used in SIC machine.	2	U	CO1
3b)	Write the object program for the following object code with program name 'TEST'.	5	A	CO2

Address	Object Code	Address	Object Code
1000	141033	1030	000000
1003	482039	2039	041030
1006	001036	203C	001030
1009	281030	203F	E0205D
100C	301015	2042	30203F
100F	482061	2045	D8205D
1012	3C1003	2048	281030
1015	00102A	204B	302057
1018	0C1039	204E	549039
101B	00102D	2051	2C205E
101E	0C1036	2054	38203E
1021	482061	2057	101036
1024	081033	205A	4C0000
1027	400000	205D	F1
102A	454F46	205E	001000
102D	000003		

3c)	Generate the object code and the symbol table for the following SIC/XE program. Use the opcodes given in question 1c.	8	A	CO2

```
FIRST  CLEAR  X
       CLEAR  A
       LDCH =X'04'
       RMO  A,S
       +LDT  #4096
       USE CDATA
       LTORG
       USE
LOOP   TD =X'F3'
       JEQ LOOP
       RD =X'F3'
       COMPR A,S
       JEQ EXIT
       STCH BUF,X
       TIXR T
       JLT LOOP
EXIT   STX RECL
       USE    CDATA
       LTORG
RECL   RESW 1
BUF    RESB 4096
```

Scanned with CamScanner

M S Ramaiah Institute of Technology

Department of Information Science and Engineering

Term:	31/01/2019 to 22/05/2019	**Course Code:**	IS61
Course:	System Software	**Semester:**	6
CIE:	Test I	**Max Marks:**	30
Date:	11/03/19	**Time:**	9 :30 –10 :30 AM

SCHEME AND SOLUTIONS

Sl No	Questions	Marks
1a)	Explain the instruction formats of SIC/XE machine.	4
	Four instruction formats	1mx4

Format 1 (1 byte):

8
op

Format 2 (2 bytes):

8	4	4
op	r1	r2

Format 3 (3 bytes):

6	1	1	1	1	1	1	12
op	n	i	x	b	p	e	disp

Format 4 (4 bytes):

6	1	1	1	1	1	1	20
op	n	i	x	b	p	e	address

1b)	Suppose that ALPHA is an array of 100 words, write a sequence of instructions for SIC/XE to set all 100 elements of the array to 0.			6
		LDS	#3	6m
		LDT	#300	
		LDX	#0	
		LDA	#0	
	STLP	STA	ALPHA,X	
		ADDR	S,X	
		COMPR	X,T	
		JLT	STLP	
	ALPHA	RESW	100	

Scanned with CamScanner

1c) Given the following SIC program, write the object code. RDREC is at location 203D and WRREC is at location 2062. **5**

0000	COPY	START	0000	
0000	CLOOP	JSUB	RDREC	48203D
0003		LDA	LENGTH	000034
0006		COMP	ZERO	28002B
0009		JEQ	ENDFIL	300012
000C		JSUB	WRREC	482062
000F		J	CLOOP	300000
0012	ENDFIL	LDA	EOF	000024
0015		STA	BUFFER,X	0C8046
0018		LDA	THREE	000028
001B		STA	LENGTH,X	0C8034
001E		JSUB	WRREC	482062
0021		LDL	RETADR	08002E
0024	EOF	BYTE	C'KLNC'	4B4C4E43
0028	THREE	WORD	3	000003
002B	ZERO	WORD	0	000000
002E	RETADR	RESW	2	
0034	LENGTH	RESW	6	
0046	BUFFER	RESB	4096	
1046	FIRST	STL	RETADR	14002E
1049		RSUB		4C0000

2a) Illustrate the role of the flag bits n.i.x in addressing modes of Format 3 and 4 of SIC/XE instructions. **4**

- Simple n=0, i=0 (SIC) or n=1, i=1
- Immediate n=0, i=1 TA=Value
- Indirect n=1, i=0 TA=(Operand)
- Index x=1 $TA_{new}=TA_{old}+(X)$
-
- TA=(B)+disp+(X)
- Index+PC relative x=1, b=0, p=1 TA=(PC)+disp+(X)
- Index+Direct x=1, b=0, p=0

2b) Generate the Object code for the following SIC/XE program (LDA-00,STA-0C,LDX-04, ADD-18, TIX-2C, JLT-38) **8**

0000	Q1	START	0000	
0000	FIRST	+LDA	VAL	0310001D
0004		STA	BLOCK	0F2010
0007		LDX	#0	050000
000A	LOOP	ADD	BUF	1B200D
000D		TIX	#5	2D0005
0010		JLT	LOOP	3B2FF7
0013		+STA	VAL	0F10001D
0017	BLOCK	RESW	1	
001A	BUF	WORD	5	000005
001D	VAL	WORD	13	00000D
0020		END	FIRST	

Scanned with CamScanner

2c)	Suppose memory addresses 21, 33 and 2700 (all decimal) of a SIC/XE machine contain the values 2400, 2700 and 33 (all decimal) respectively.If (PC)=0000 and the A register (i.e., the accumulator) contains the value −17 (decimal) when the machine executes the 3-byte instruction whose hexadecimal representation is 0E2021. Show the contents of the memory locations 21, 33 and 2700 after the execution of the instruction. Your answers must be in decimal. Opcode of STA is 0C. Show the working.	3
	21 CONTAINS 2400 33 CONTAINS 2700 2700 CONTAINS -17	1mx3
3a)	Explain the addressing modes used in SIC machine.	2
	❑ There are two addressing modes available ❑ Indicated by x bit in the instruction ❑ (X) represents the contents of reg. X	1mx2

Mode	Indication	Target address calculation
Direct	x = 0	TA = address
Indexed	x = 1	TA = address + (X)

3b) Write the object program for the following object code with program name 'TEST'. 5

Address	Object Code	Address	Object Code
1000	141033	1030	000000
1003	482039	2039	041030
1006	001036	203C	001030
1009	281030	203F	E0205D
100C	301015	2042	30203F
100F	482061	2045	D8205D
1012	3C1003	2048	281030
1015	00102A	204B	302057
1018	0C1039	204E	549039
101B	00102D	2051	2C205E
101E	0C1036	2054	38203E
1021	482061	2057	101036
1024	081033	205A	4C0000
1027	400000	205D	F1
102A	454F46	205E	001000
102D	000003		

H^TEST ^001000^001061

T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D

T^00101E^15 ^0C1036^482061^081033^400000^454F46^000003^000000

T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203E

T^002057^0A^101036^4C0000^F1^001000

E^001000

Scanned with CamScanner

3c) Generate the object code and the symbol table for the following SIC/XE program. Use the opcodes given in question 1c. 8

0000	FIRST	CLEAR	X	B410
0002		CLEAR	A	B400
0004		LDCH	=X'04'	53201F
0007		RMO	A,S	AC04
0009		+LDT	#4096	75101000
0000		USE	CDATA	
		LTORG		
0000	*	=X'04'		04
000D		USE		
000D	LOOP	TD	=X'F3'	E32017
001O		JEQ	LOOP	332FFA
0013		RD	=X'F3'	DB2011
0016		COMPR	A,S	A004
0018		JEQ	EXIT	332008
001B		STCH	BUF,X	57A00D
001E		TIXR	T	B850
0020		JLT	LOOP	3B2FEA
0023	EXIT	STX	RECL	132002
0001		USE	CDATA	
		LTORG		
0001	*	=X'F3'		F3
0002	RECL	RESW	1	
0005	BUF	RESB	4096	

Scanned with CamScanner

Mobile phones are banned

M S Ramaiah Institute of Technology

Department of Information Science and Engineering

Term:	31/01/2019 to 22/05/2019	**Course Code:**	IS61
Course:	System Software	**Semester:**	6
CIE:	Test II	**Max Marks:**	30
Date:	**08/04/2019**	**Time:**	9 :30 –10:30 AM

Portions for Test: Lecture Nos. from 15 to 30 as per lesson plan

Instructions to Candidates: Answer any **Two** out of Three questions.

Q.No	Questions	Marks	Bloom's Level	CO
1.a)	Describe the working of load and go assembler.	5	U	2
b)	Discuss why the bit-mask method of specifying program relocation is efficient than the modification record method for the SIC program. Write the bit-mask for the following object code of SIC program. **Loc** **Source Statements** **Object code** 0015 ENDFIL LDA EOF 00001E 0018 STA THREE 0C0021 001B LDL ZERO 080024 001E EOF BYTE C'EOF' 454F46 0021 THREE WORD 3 000003 0024 ZERO WORD 0 000000 END	4	Ap	3
c)	Design the algorithm for the pass-2 of a linking loader.	6	U	3
2.a)	Provide the multi-pass assembler operations for the following sequence of symbol defining statements that involve forward references. HF EQU MAX MAX EQU BUF-BU PRE EQU BUF-1 BUF RESB 1024 BU EQU * Location value for BUF is 1034	5	Ap	2
b)	Illustrate with an example the working of Bootstrap loader with SIC source code.	6	Ap	3
c)	List machine independent loader features. Explain any one.	4	U	3
3.a)	Define control section. How does a control section differ from program blocks?	5	An	2

Scanned with CamScanner

b)	Given the following ESTAB and the object program depict the contents of the memory.	6	Ap	3

ESTAB

Control Section	Symbol Name	Address	Length
PROGA		5000	0063
	LA	5040	
	EA	5054	
PROGB		5063	007F
	LB	50C3	
	EB	50D3	
PROGC		50E2	0051
	LC	5112	
	EC	5124	

Object Program

```
H PROGA 000000 000063
D LA 000040 EA 000054
R LB EB LC EC
..
T 000020 0A 03201D,77100004,050014
.
.
T 000054 0F 000014 FFFFF6 00003F 000014 FFFFC0
M 000024 05+LB
M 000054 06+LC
M 000057 06+EC
M 00005A 06+LC
M 00005D 06+EC
E 000020
```

c)	Explain with a neat diagram loading and calling of a subroutine using dynamic linking.	4	U	3

U- Understand, Ap- Apply ,An-Analyze

Scanned with CamScanner

M S Ramaiah Institute of Technology

Department of Information Science and Engineering

Term:	31/01/2019 to 22/05/2019	**Course Code:**	IS61
Course:	System Software	**Semester:**	6
CIE:	Test II	**Max Marks:**	30
Date:	**08/04/2019**	**Time:**	9 :30 –10:30 AM

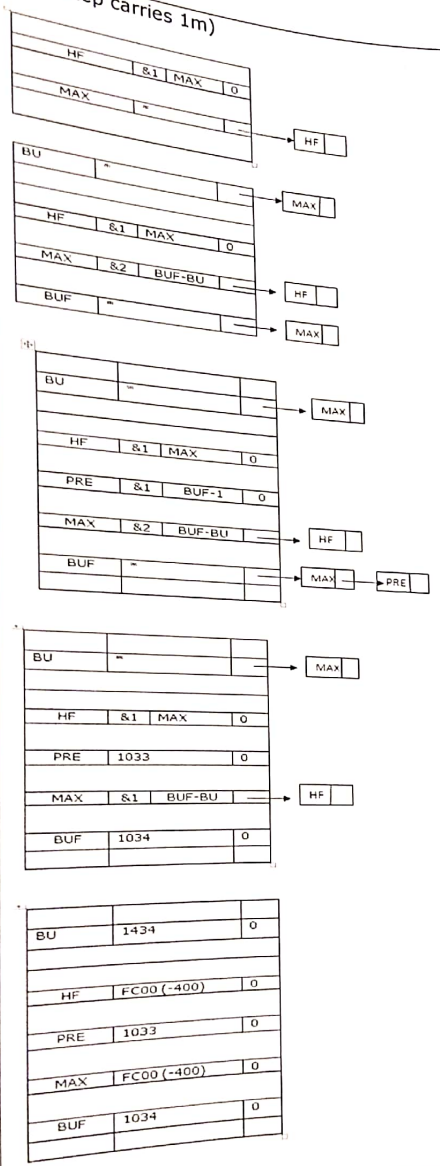
Scheme of Evaluation

Q.No	Questions	Marks	CO
1.a)	• Load-and-go assembler generates their object code in memory for immediate execution. • No object program is written out, no loader is needed. • Handling Forward references explanation---(3m) }-(2m)	5	2
b)	Bit mask method is more efficient than the modification record for the SIC program because in SIC most of the instructions are direct addressing. Bit mask is **E00** **OR** T 000015 E00 00001E 0C0021 080024 454F46 000003 000000	4	3

c)	Pass 2:
	Begin
	set CSADDR to PROOADDR
	set EXECADDR to PROOADDR
	while not end of input do
	} ------(1m)
	begin
	read next input record {Header record}
	set CSLTH to control section length
	while record type != 'E' do
	begin
	read next input record
	if record type = 'T' then
	} ----(1m)
	begin
	{if object code is in character form, convertinto internal representation}
	move object code from record to location (CSADDR + specified address)
	} ----(1m)
	end {if 'T'}
	else if record type = 'M' then
	begin
	search ESTAB for modifying symbol name
	if found then
	add or subtract symbol value at location (CSADDR + specified address)
	Else
	set error flag (undefined external symbol)
	end {if 'M' }
	} ---(2m)
	end {while != 'E'}
	if an address is specified {in End record} then
	set EXECADDR to (CSADDR + specified address)
	add CSLTH to CSADDR
	} ---(1m)
	end {while not EOF}
	jump to location given by EXECADDR {to start execution of loaded program)
	end {Pass 2}
		6	3

Scanned with CamScanner

2 a)	(Each step carries 1m)	5	2
b)	**Main Program—2m ; Subroutine—2m ;Illustration—2m**	6	3

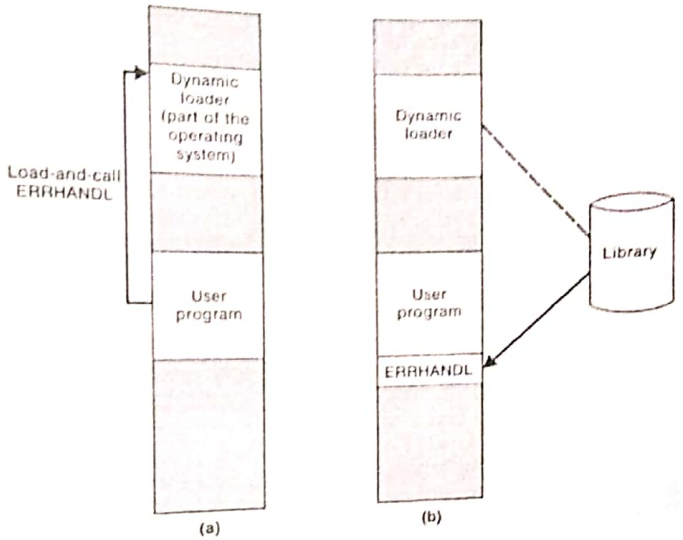
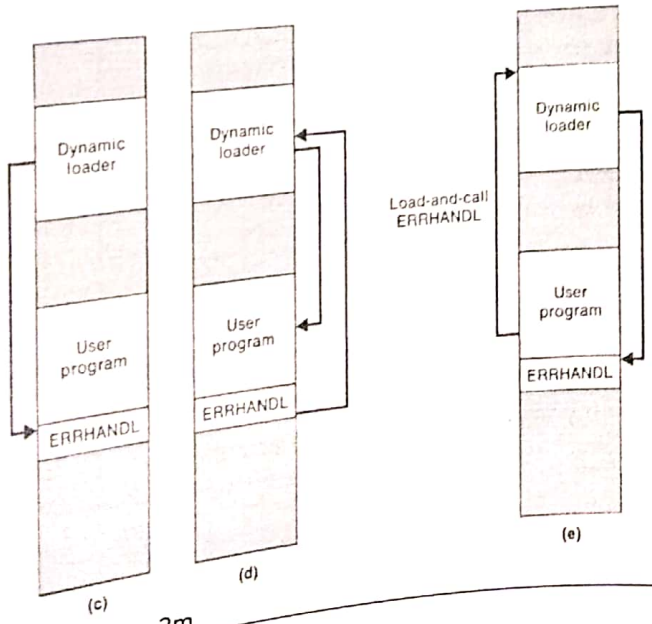


Scanned with CamScanner

```
BOOT      START     0          BOOTSTRAP LOADER FOR SIC/XE
.
. THIS BOOTSTRAP READS OBJECT CODE FROM DEVICE F1 AND ENTERS IT
. INTO MEMORY STARTING AT ADDRESS 80 (HEXADECIMAL). AFTER ALL OF
. THE CODE FROM DEVF1 HAS BEEN SEEN ENTERED INTO MEMORY, THE
. BOOTSTRAP EXECUTES A JUMP TO ADDRESS 80 TO BEGIN EXECUTION OF
. THE PROGRAM JUST LOADED.  REGISTER X CONTAINS THE NEXT ADDRESS
. TO BE LOADED.
.
          CLEAR     A        CLEAR REGISTER A TO ZERO
          LDX       #128     INITIALIZE REGISTER X TO HEX 80
LOOP      JSUB      GETC     READ HEX DIGIT FROM PROGRAM BEING LOADED
          RMO       A,S      SAVE IN REGISTER S
          SHIFTL    S,4      MOVE TO HIGH-ORDER 4 BITS OF BYTE
          JSUB      GETC     GET NEXT HEX DIGIT
          ADDR      S,A      COMBINE DIGITS TO FORM ONE BYTE
          STCH      0,X      STORE AT ADDRESS IN REGISTER X
          TIXR      X,X      ADD 1 TO MEMORY ADDRESS BEING LOADED
          J         LOOP     LOOP UNTIL END OF INPUT IS REACHED
. SUBROUTINE TO READ ONE CHARACTER FROM INPUT DEVICE AND
. CONVERT IT FROM ASCII CODE TO HEXADECIMAL DIGIT VALUE. THE
. CONVERTED DIGIT VALUE IS RETURNED IN REGISTER A. WHEN AN
. END-OF-FILE IS READ, CONTROL IS TRANSFERRED TO THE STARTING
. ADDRESS (HEX 80).
.
GETC      TD        INPUT    TEST INPUT DEVICE
          JEQ       GETC     LOOP UNTIL READY
          RD        INPUT    READ CHARACTER
          COMP      #4       IF CHARACTER IS HEX 04 (END OF FILE),
          JEQ       80           JUMP TO START OF PROGRAM JUST LOAD
          COMP      #48      COMPARE TO HEX 30 (CHARACTER '0')
          JLT       GETC     SKIP CHARACTERS LESS THAN '0'
          SUB       #48      SUBTRACT HEX 30 FROM ASCII CODE
          COMP      #10      IF RESULT IS LESS THAN 10, CONVERSION
          JLT       RETURN      COMPLETE. OTHERWISE, SUBTRACT 7 MOR
          SUB       #7          (FOR HEX DIGITS 'A' THROUGH 'F')
RETURN    RSUB               RETURN TO CALLER
INPUT     BYTE      X'F1'    CODE FOR INPUT DEVICE
          END       LOOP
```

c)	Listing Machine independent Loader features ---(1m) • Automatic Library search • Loader options Any one explanation---(3m)	4	3
3 a)	Definition of Control section **------(2m)** A control section is a part of the program that maintains its identity after assembly. **Difference between Control section and Program blocks----(3m)** **Program Blocks** • refer to segments of code that are rearranged within a single object program unit • Each program block may actually contain several separate segments of the source program **Control Section** • are most often used for subroutines or other logical subdivisions of a program • the programmer can assemble, load, and manipulate each of these control sections separately • instruction in one control section may need to refer to instructions or data located in another section because of this, there should be some means for linking control sections together	5	2

Scanned with CamScanner

b)	******** ******** ******** ******** 5000 . 5010 . 5020 03201D77 1050C705 0014**** ---(3M) .. .. 5050 ******** 00512600 511A0051 51005138 ----(3M)	6	3
c)	 --1m  --(1m) Explanation ---2m	4	3

U- Understand, Ap- Apply ,An-Analyze

Scanned with CamScanner

Mobile phones are banned

M S Ramaiah Institute of Technology

Department of Information Science and Engineering

Term:	31/01/2019 to 22/05/2019	**Course Code:**	15CS61
Course:	System Software	**Semester:**	6
CIE:	Test III	**Max Marks:**	30
Date:	09/05/2019	**Time:**	9:30 – 10:30 AM

Instructions to Candidates: Answer any **Two** out of Three questions. **Portions for Test:** Lecture Nos. from 35 to 56 as per lesson plan

Q.No	Questions	Marks	Bloom's Level	CO
1.a)	Explain the structure of lex program supported with a typical example	7	U	5
b)	Expand the following macro invocation statements using the given macro definition- Function f2,Q,(10,20,30) followed by Function f1,R, 40 **Macro definition** Function MACRO &I, &Len, &List &C1 SET %NITEMS(&List) +LDT #4096 $Loop TD =X'&I' JEQ $Loop RD =X'&I' &C2 SET 1 WHILE (&C2 LE &C1) COMP =X'0000&List[&C2]' JEQ $Exit &C2 SET &C2+1 ENDW TIXR T JLT $Loop &Exit STX &Len MEND	8	Ap	4
2.a)	Write a Lex program to find the length of the longest word in a given file.	7	Ap	5
b)	Develop the data structures and algorithm for a one pass macro processor.	8	U	4
3.a)	Write a YACC program to recognize a valid identifier, which starts with a letter or underscore, followed by any number of letters or digits.	8	Ap	5
b)	Given below is a macro definition. Explain the data structures used and indicate the data stored in these data structures for the macro call . **RDB F1,B,1048** Macro Definition RDB MACRO &IN, &BUF, &REC CLEAR X CLEAR A +LDT #4096 TD =X'&IN' JEQ *-3 RD =X'&IN' COMPR A,S JEQ *+11 STCH &BUF TIXR T JLT *-19 STX &REC MEND	7	AP	4

U- Understand, Ap- Apply

Scanned with CamScanner

System Software (IS61)

Scheme

1 A) Structure of lex----------3

Any typical lex program example----------4

b) i) +LDT #4095 ----------4

```
$AALOOP   TD=X 'f2'
          JEQ $AALOOP
          RD=X'f2'
          COMP=X'000010'
          JEQ $AAExit
          COMP=X'000020'
          JEQ $AAExit
          COMP=X'000030'
          JEQ $AAExit
          TIXR T
          JLT $AALoop
$AAexit   STX   Q
```

ii) +LDT #4095 ----------4

```
$ABLOOP   TD=X 'f1'
          JEQ $ABLOOP
          RD=X'f1'
          COMP=X'000040'
          JEQ $AAExit
          TIXR T
          JLT $ABLoop
$Abexit   STX   R
```

2 a) `*lex code to find the length of the longest word*/` ----------7

```
% {
  int counter = 0; %
}

%
% [a - zA - Z] + {
  if (yyleng > counter) {
    counter = yyleng;
  }
```

Scanned with CamScanner

```
} %
%

main() {
  yylex();
  printf("largest: %d", counter);
  printf("\n");
}
```

- 2 b) data structures and algorithms 4*2= 8
  - The procedure PROCESSING
  - The procedure DEFINE
    - Called when the beginning of a macro definition is recognized, makes the appropriate entries in DEFTAB and NAMTAB.
  - The procedure EXPAND
    - Called to set up the argument values in ARGTAB and expand a macro invocation statement.
  - The procedure GETLINE
    - Called at several points in the algorithm, gets the next line to be processed.
  - EXPANDING is set to TRUE or FALSE.

```
begin {macro processor}
    EXPANDING := FALSE
    while OPCODE ≠ 'END' do
        begin
            GETLINE
            PROCESSLINE
        end {while}
end {macro processor}


procedure PROCESSLINE
    begin
        search NAMTAB for OPCODE
        if found then
            EXPAND
        else if OPCODE = 'MACRO' then
            DEFINE
        else write source line to expanded file
    end {PROCESSLINE}
```

Figure 4.5 Algorithm for a one-pass macro processor.

Scanned with CamScanner

```
procedure DEFINE
    begin
        enter macro name into NAMTAB
        enter macro prototype into DEFTAB
        LEVEL  := 1
        while LEVEL > 0 do
            begin
                GETLINE
                if this is not a comment line then
                    begin
                        substitute positional notation for parameters
                        enter line into DEFTAB
                        if OPCODE = 'MACRO' then
                            LEVEL := LEVEL + 1
                        else if OPCODE = 'MEND' then
                            LEVEL  := LEVEL - 1
                    end {if not comment}
            end {while}
        store in NAMTAB pointers to beginning and end of definition
    end {DEFINE}

procedure EXPAND
    begin
        EXPANDING  := TRUE
        get first line of macro definition {prototype} from DEFTAB
        set up arguments from macro invocation in ARGTAB
        write macro invocation to expanded file as a comment
        while not end of macro definition do
            begin
                GETLINE
                PROCESSLINE
            end {while}
        EXPANDING := FALSE
    end {EXPAND}

procedure GETLINE
    begin
        if EXPANDING then
            begin
                get next line of macro definition from DEFTAB
                substitute arguments from ARGTAB for positional notation
            end {if}
        else
            read next line from input file
    end {GETLINE}
```

Figure 4.5 *(cont'd)*

3 a) **Lex File**

```
%{
#include <stdio.h>
#include "y.tab.h"
extern yylval;
%}
%%
int | char | bool | float | void | for | do | while | if | else | return | void {printf("keyword
```

Scanned with CamScanner

```
is
%s\n",yytext);return KEY;}
[a-zA-Z0-9]+ {printf("identifier is %s\n",yytext);return ID;}
. ;
        %%
```

Yacc File

```
%{
#include <stdio.h>
#include <stdlib.h>
int id=0, key=0;
%}
%token ID KEY
%%
input:
| ID input { id++; }
| KEY input { key++; }
| ID { id++; }
| KEY { key++; }
;
%%
#include <stdio.h>
extern int yylex();
extern int yyparse();
extern FILE *yyin;
main() {
FILE *myfile = fopen("sam_input.c", "r");
if (!myfile) {
        printf("I can't open sam_input.c!");

return -1;
}
yyin = myfile;
do {
yyparse();
} while (!feof(yyin));
printf("numbers = %d\nKeywords = %d\nIdentifiers = %d\
dig, key,id);
}
void yyerror() {
printf("EEK, parse error! Message: ");
exit(-1);
        }
```

b) Explanation of different data structures NAMETAB,ARGTAB and DEFTAB for the macro call ---3
Macro expansion ----------4

Scanned with CamScanner