Mumbai Education Trust's

INSTITUTE OF ENGINEERING, BKC, NASHIK.

T.E (Computer Engineering)

SYSTEM PROGRAMMING & OPERATING SYSTEM (2019 Pattern)

INSEM OCT-2022

Time : 1 Hour ***[Max. Marks : 30]***

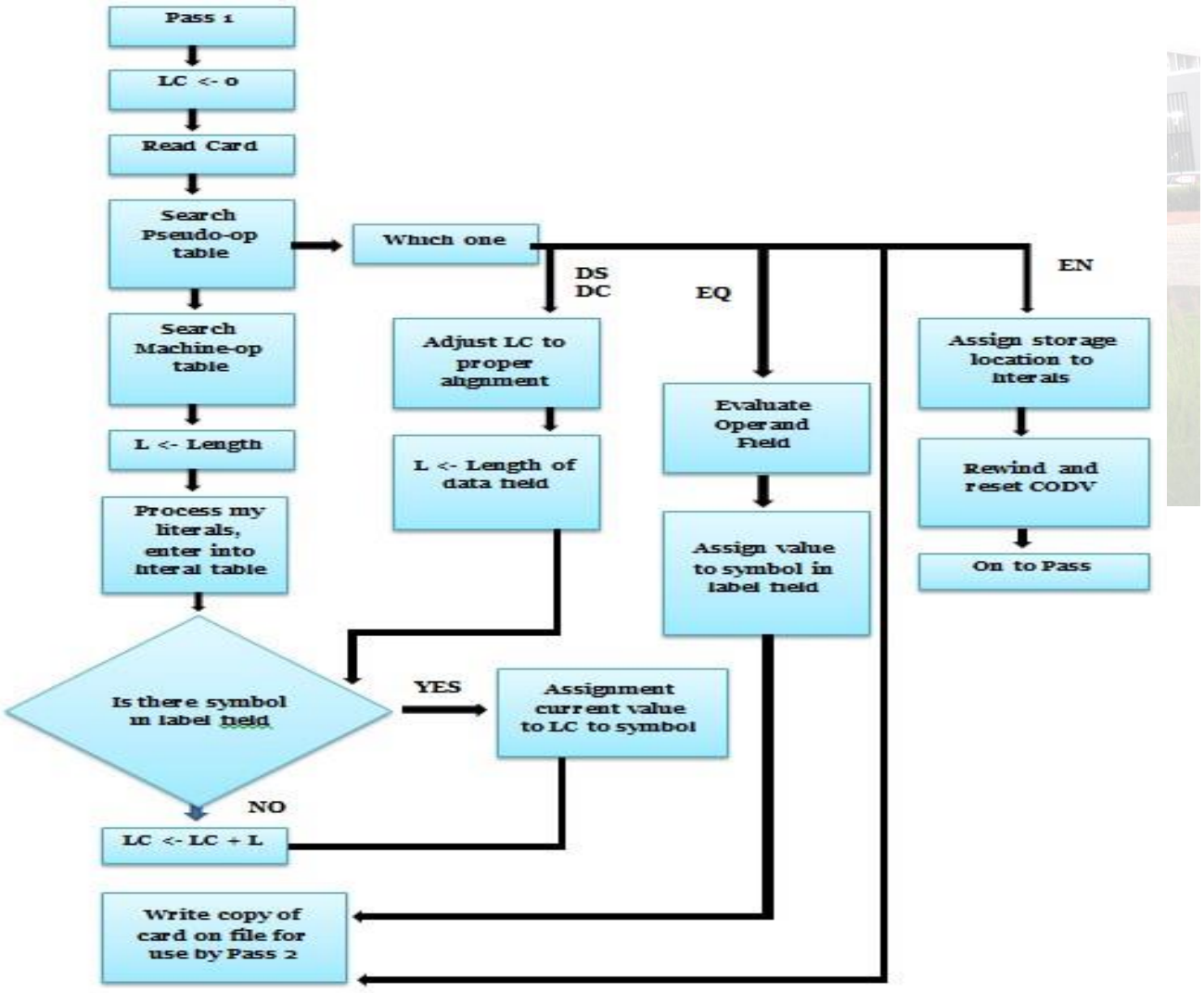
Instructions to the candidates:

1) *Solve Que. 1 or 2, Que.3 or 4 .*
2) *Neat diagrams must be drawn wherever necessary.*
3) *Assume suitable data if necessary.*
4) *Figures to the right indicate full marks.* **Date : 08/10/2022**

SPOS INSEM MODEL ANSWER OCT-2022

Q 1. a) Draw and explain flowchart of pass I of two pass assembler with example. 8-Marks

Ans :- **Flowchart of Pass-I of Two Pass Assembler :**



Explaination :

The primary function performed by the analysis phase is the building of the symbol table. For this purpose it must determine the addresses with which the symbol names used in a program are associated. It is possible to determine some address directly, e.g. the address of the first instruction in the program, however others must be inferred.

To implement memory allocation a data structure called location counter (LC) is introduced. The location counter is always made to contain the address of the next memory word in the target program. It is initialized to the constant. Whenever the analysis phase sees a label in an assembly statement, it enters the label and the contents of LC in a new entry of the symbol table. It then finds the number of memory words required by the assembly statement and updates; the LC contents. This ensure: that LC points to the next memory word in the target program even when machine instructions have different lengths and DS/DC statements reserve different amounts of memory. To update the contents of LC, analysis phase needs to know lengths of different instructions. This information simply depends on the assembly language hence the mnemonics table can be extended to include this information in a new field called length. We refer to the processing involved in maintaining the location counter as LC processing.

b) Difference between Literal and Immediate operand. 7-Marks

Ans :

Sr No	Literal	Sr No	Immediate Operand.
1.	Assembler generates specified value as a constant at some other mem. location.	1.	The operand value is assembled as part of the m/c instruction.
2.	Example: MOVER B, ='5'	2.	Example: MOVE A, #5
3.	Literal use '=' symbol	3.	Immediate Operand use '#' sym.
4.	There is mem. reference	4.	There is no mem reference
5.	LTORG (AD) instn is needed	5.	LTORG Assembler directive is not needed
6.	**Literal table is used to store Literal**	6.	**Symbol table is used to store Constant.**
7.	**Literal contains number or String**	7.	**It contain number only.**

OR

Q 2. a) Explain in details with suitable example the formats and content of the database used in assembler design ? 8-Marks

➔ **Data structure used :**

Data Structure used are as follows:

1. Symbol table
2. Literal Table
3. Mnemonic Opcode Table
4. Pool Table

Symbol Table:

Name of Symbol	address

- MOT:

Mnemonic	Machine Opcode	Class	Length

Literal Table:

Literal	address

- Pool Table:

Literal Number

```
        START 200
        MOVER AREG, ='5'
        MOVEM AREG, X
L1      MOVER BREG, ='2'
        ORIGIN L1+3
        LTORG

NEXT    ADD AREG, ='1'
        SUB BREG, ='2'
        BC LT, BACK
        LTORG

BACK    EQU L1
        ORIGIN NEXT+5
        MULT CREG, ='4'
        STOP
X       DS 1
        END
```

```
START 200
        MOVER AREG, ='5'        200
        MOVEM AREG, X           201
L1      MOVER BREG, ='2'        202
        ORIGIN L1+3
        LTORG
                ='5'            205
                ='2'            206
NEXT    ADD AREG, ='1'          207
        SUB BREG, ='2'          208
        BC LT, BACK             209
        LTORG
                ='1'            210
                ='2'            211
BACK    EQU L1
        ORIGIN NEXT+5
        MULT CREG, ='4'         212
        STOP                    213
X       DS 1                    214
        END
                ='4'            215
```

index	Symbol Name	Address
0	X	214
1	L1	202
2	NEXT	207
3	BACK	202

index	Literal	Address
0	5	205
1	2	206
2	1	210
3	2	211
4	4	215

Literal number
0
2
4

Note : You should theory of each database

b) Explain Algorithm of Pass-1 of two pass Assembler. 7-Marks

Ans :-

Pass I:

(i) Separate the symbol, operand fields and mnemonic opcode

(ii) Make the symbol table

(iii) Perform the LC processing

(iv) Constructs intermediate representation.

Pass I uses the subsequent data structures:

OPTAB: A table of associated information and mnemonic opcodes

SYMTAB: It is a symbol table

LITTAB: A table literally utilized in the program

OPTAB consists of the field mnemonic opcode, information and class. The class field shows whether the opcode corresponds to a declaration statement (DL) an imperative statement (IS) or an assembler directive (AD).

SYMTAB entry consists of the fields address and length.

A *LITTAB* entry consists of literals and address.

Firstly, We will take a small assembly language program to understand the working in their respective passes. Assembly language statement format :

```
[Label] [Opcode] [operand]


Example:  M  ADD  R1, ='3'
where, M - Label; ADD - symbolic opcode;
R1 - symbolic register operand; (='3') - Literal
```

```
Assembly Program:
Label  Op-code   operand   LC value(Location counter)
JOHN   START     200
       MOVER     R1, ='3'    200
       MOVEM     R1, X       201
L1     MOVER     R2, ='2'    202
       LTORG                 203
X      DS        1           204
       END                   205
```

Let’s take a look on how this program is working:

1. **START:** This instruction starts the execution of program from location 200 and label with START provides name for the program.(JOHN is name for program)
2. **MOVER:** It moves the content of literal(=’3′) into register operand R1.
3. **MOVEM:** It moves the content of register into memory operand(X).
4. **MOVER:** It again moves the content of literal(=’2′) into register operand R2 and its label is specified as L1.
5. **LTORG:** It assigns address to literals(current LC value).
6. **DS(Data Space):** It assigns a data space of 1 to Symbol X.
7. **END:** It finishes the program execution.

Working of Pass-1: Define Symbol and literal table with their addresses.

Note: Literal address is specified by LTORG or END.

Step-1: START 200 (here no symbol or literal is found so both table would be empty)

Step-2: MOVER R1, =’3′ 200 (=’3′ is a literal so literal table is made)

Literal	Address
=’3′	– – –

Step-3: MOVEM R1, X 201

X is a symbol referred prior to its declaration so it is stored in symbol table with blank address field.

Symbol	Address
X	– – –

Step-4: L1 MOVER R2, =’2′ 202

L1 is a label and =’2′ is a literal so store them in respective tables

Symbol	Address
X	– – –
L1	202

Literal	Address
=’3′	– – –
=’2′	– – –

Step-5: LTORG 203

Assign address to first literal specified by LC value, i.e., 203

Literal	Address
=’3′	203
=’2′	– – –

Step-6: X DS 1 204

It is a data declaration statement i.e X is assigned data space of 1. But X is a symbol which was referred earlier in step 3 and defined in step 6.This condition is called Forward Reference Problem where variable is referred prior to its declaration and can be solved by back-patching. So now assembler will assign X the address specified by LC value of current step.

Symbol	Address
X	204
L1	202

Step-7: END 205

Program finishes execution and remaining literal will get address specified by LC value of END instruction. Here is the complete symbol and literal table made by pass 1 of assembler.

Symbol	Address
X	204
L1	202

Literal	Address
=’3′	203
=’2′	205

Now tables generated by pass 1 along with their LC value will go to pass-2 of assembler for further processing of pseudo-opcodes and machine op-codes.

Q 3. a)Define macro. What are the advantages of macro facility? How they are different from function. 8-Marks

Ans :-

- Macro allows a sequence of source language code to be defined once and then referred many times.
- **" A macro is a sequence of instructions, assigned by a name and could be used anywhere in the program "**
- In NASM, macros are defined with %macro and %endmacro directives.
- The macro begins with the %macro directive and ends with the %endmacro directive.
- Syntax:

%macro macro_name number_of_params

<macro body>

%endmacro

Advantages of Macro :

- Macros hold the details of an operation in a module that can be used "as if" it were a single instruction.
- A frequently used sequence of instructions can be defined as a macro.
  - Now the macro can be used rather than retyping the sequence.
  - The sequence can be debugged just once, rather than each time it is typed in.
  - By using acual arguments, variations of the same idea are implemented.
- Macros are used to build up complex operations out of simpler operations.
- Libraries of useful macros can be created. Programmers can use these libraries without knowing much about the detailed code the macros expand into.

Difference between Macro & Function :

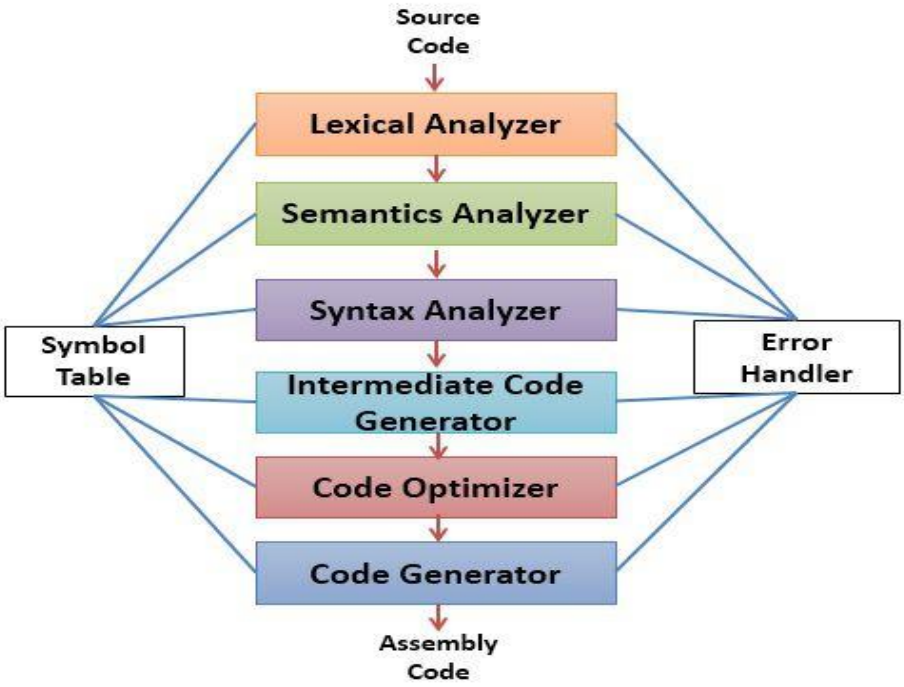
Sr. No	Macro	Function
1	Macro can be called only in the program it is defined.	Subroutine can be called from other programs also.
2	More space is required	Less space is required

3	Execution speed is faster.	Execution speed is slower
4	Macro Can not handle labels	Subroutines can handle labels
5	Macro is executed by assembler	Subroutine is executed by hardware
6	Code size increase	Code size does not increase.
7	Simple to write and use	Complex to write and understand
8	Macro name[parameter] Mend	Subroutine name(parameter) end

b)Write short note on : 1) Compiler 2) Interpreter 7-Marks

➔ **Definition : Compiler is a program which convert high level language to machine level language.**

Phases of Compiler :



Compiler

Phases of Compiler

***Lexical Analysis Phase*:**

– The lexical phase reads the characters in the source program and groups them into a stream of tokens in which each token represents a logically cohesive sequence of characters, such as, An identifier, A keyword, A punctuation character.

– The character sequence forming a token is called the lexeme for the token.

***Syntax Analysis Phase*:**

– Syntax analysis imposes a hierarchical structure on the token stream. This hierarchical structure is called syntax tree.

– A syntax tree has an interior node is a record with a field for the operator and two fields containing pointers to the records for the left and right children.

– A leaf is a record with two or more fields, one to identify the token at the leaf, and the other to record information about the token.

***Error Detecting and Reporting*:**

– Each phase encounters errors.

– Lexical phase determine the input that do not form token.

– Syntax phase determine the token that violates the syntax rule.

– Semantic phase detects the constructs that have no meaning to operand.

Interpreter :

All high level languages need to be converted to machine code so that the computer can understand the program after taking the required inputs. The software by which the conversion of the high level instructions is performed line-by-line to machine level language, other than compiler and assembler, is known as INTERPRETER. If an error is found on any line, the execution stops till it is corrected. This process of correcting errors is easier as it gives line-by-line error but the program takes more time to execute successfully. Interpreters were first used in 1952 to ease programming within the limitations of computers at the time. It translates source code into some efficient intermediate representation and immediately execute this.

Advantage of interpreter is that it is executed line by line which helps users to find errors easily.

Disadvantage of interpreter is that it takes more time to execute successfully than compiler.

Some examples of programming languages that use interpreters are Python, Ruby, Perl, PHP and Matlab.

OR

Q 4. a) What are the different data structure required for Two Pass Macro Processor? Justify which data structure are implemented at that time of processing macro definition, macro call and macro expansion. 8-Marks

Ans :- Data Structures

1. MACRO Definition table
2. MACRO Name Table
3. Argument Array List

Format of Data Structures

MACRO Definition Table

Index	Definition	Arguments
1	INCR	&arg1, &arg2
2	A	#1
3	A	#2
4	MEND	

Note: Argument list contains the index of the argument in the argument list array.

MACRO Name Table

Index	Name	MDT Index
1	"INCR"	1

Note: MDT index contains the starting index of the MACRO in MDT.

Argument List Array

Index	Argument
1	"DATA 1bbb"
2	"DATA 2bbb"

Note: MDT index contains the starting index of the MACRO in MDT.

Pass 1 (Definition of MACROS)

- Pass1 of macro processor makes a line-by-line scan over it's input.
- Set MDTC = 1 and MNTC = 1.
- Read next line of input program.
- If it is a MACRO pseudo-op, the entire macro definition except MACRO line is stored in MDT.
- The name is entered in the MNT along with a pointer to the 1st location of MDT entry.
- When the END pseudo-op is encountered all the macro-definitions have been processed, so control is transferred to pass2.

Pass 2 (Replacing MACRO calls by its definition)

- This algorithm reads one line of i/p program at a time.
- For each line it checks if op-code of that line matches any of the MNT entry.
- The initial value of MDTP is obtained from MDT index field of MNT entry.
- The macro expander prepares the ALA consisting of a table of dummy argument indices and corresponding arguments to the call.
- The value from the argument list is substituted for dummy arguments indices in the macro definition table.
- Reading MEND line in MDT **terminates expansion of macro and scanning continues in the input file.**
- When END pseudo-op is encountered, the expanded source program is given to the assembler.

b) Explain Argument passing mechanism in macro with suitable example. 7-Marks

Ans : There are two types of macro parameters:

1. Positional
2. Keyword

1. Positional Parameters :

Positional parameters are symbolic parameters that must be specified in a specific order every time the macro is called. The parameter will be replaced within the macro body by the value specified when the macro is called.

```
      MACRO
&LAB INCR &ARG0, &ARG1=X,&ARG2=Y
&LAB  ADD AREG, &ARG1
       ADD AREG, &ARG2
      ADD AREG, &ARG3
      MEND
```

Positional parameters: is written as & parameter name
eg. &LAB and &ARG0 are Positional parameters

2. Keyword parameters are symbolic parameters that can be specified in any order when the macro is called. The parameter will be replaced within the macro body by the value specified when the macro is called. These parameters can be given a default value. If no default value is specified and if the parameter is not given a value when the macro is called, then the parameter will be replaced by a null string.

```
        MACRO
&LAB INCR &ARG0, &ARG1=X,&ARG2=Y
&LAB  ADD AREG, &ARG1
        ADD AREG, &ARG2
        ADD AREG, &ARG3
        MEND
```

. Keyword Parameter: are used for assigning default values to the parameter
eg. &ARG1 with default value X, &ARG2 with default value Y

3. Actual parameter: when Macro call is defined actual parameter will be used
LOOP INCR DATA

LOOP INCR **DATA1, DATA2,DATA3**

Example :

```
        MACRO
&LAB INCR &ARG0, &ARG1=X,&ARG2=Y
&LAB  ADD AREG, &ARG1
        ADD AREG, &ARG2
        ADD AREG, &ARG3
        MEND
```

1.Positional parameters: is written as & parameter name
eg. &LAB and &ARG0 are Positional parameters

2.Keyword Parameter: are used for assigning default values to the parameter
eg. &ARG1 with default value X, &ARG2 with default value Y

3. Actual parameter: when Macro call is defined actual parameter will be used
LOOP INCR DATA
LOOP INCR DATA1, DATA2, DATA3

********** THE END **********