B31

PRN - 2122200504

Assignment - 2

Q.1] Write a short note on
a) Macro definition
b) Macro call
c) Macro expansion

→ a) Macro definition

- A macro definition is enclosed between a macro header statement and macro end statement.
- Macro definition are typically located at the start of program.
- Macro definition cosists of
  - A macro prototype statement
  - One or more model statement
  - Macro preprocessor statement
- A macro prototype statement :
  - The macro prototype statement declares the ~~m~~ name of a macro and the names and kinds of its parameters
  - <macro name> [<formal parameter spec>, ...]
  - Where name appears in the mnemonic field of assembly statement and <formal parameter spec> is of the form and <parameter name>[<parameter kind>
- Model statement :
  - A model statement is a statement from which an assembly language statement may be generated during macro expansion.
- Macro Preprocessor statement :
  - A preprocessor statement is used to perform auxiliary functions during macro expansion

```
e.g.          Macro
              INCR    &MEM VAL, &INCR VAL, &REG
macro      -> MOVE R  &REG, &MEM_VAL   }
prototype     ADD     &REG, &INCR_VAL  }  Model statement
statement     MOVEM   &REG, &MEM_VAL   }
              MEND
```

b] Macro call

- A macro is called by writing the macro name in the mnemonicfield of an assembly statement.
  <macro name> [<actual parameter spec>,...]
- Where an actual parameter typically an operand specification in an assembly language statement
- e.g. INCR A, B, AREG

c] Macro Expansion

- A macro call leads to macro expansion, during macro expansion, the macro call statement is sequentially replaced by a assembly statements.
- '+' is used to differentiate between the the original statement of program and macro statement.
- Performed by two kinds of language processor.

Macro Assembler:

- performs expansion of each macro call in a program into sequence of assembly statements and assembles the resulting assembly program

Macro Preprocessor :

- merely performs expansion of macro calls in program.

Q.2] With suitable example describe data struture generated in Macro Expansion.

⇒ - we use the following data structure to perform macro expansion:

APTAB	Actual parameter table
EVTAB	EV table
MEC	Macro Expansion counter
APTAB_ptr	APTAB pointer
EVTAB_ptr	EVTAB pointer

- Number of entries in APTAB equals to the sum of values in the # PP and # KP fields of the MNT entry of macro.
- Macro preprocessor followed by conventional assembler is an expensive way of handling macro since the number of passes over the source program is large and many function get duplicated.
- Example :

  A source statement to detect macro calls require us to process the mnemonic field. Similar function is required in the first pass of the assembler. Similar functions of the preprocessor and assembler can be merged if macros are handled by a macro assembler which perform macro expansion and program assembly simultaneously.
- Macro expansion performs in single pass is not true, as certain kinds of forward references in macros cannot be handled in a single pass.
- This problem leads to the classical two pass organization for macro expansion
  - First pass collects information about the symbols defined in a program.

- second pass perform macro expansion

- Pass structure of a macro assembler:

First merge the function of macro preprocessor with the function of conventional assembler, then the functions can be structured into passes of the macro assembler.

Pass-I

Macro definition processing
SYMTAB construction

Pass-II

macro expansion
Memory allocation and LC processing
Processing of literals
Intermediate code generation.

Pass-III

Target code generation

- The pass structure can be simplified if attributes of actual parameter are not to be supported

Pass I :

Macro definition processing
Macro expansion
Memory allocation, LC Processing & SYMTAB construction
Processing of literals
Intermediate code generation

Pass II

Target code generation

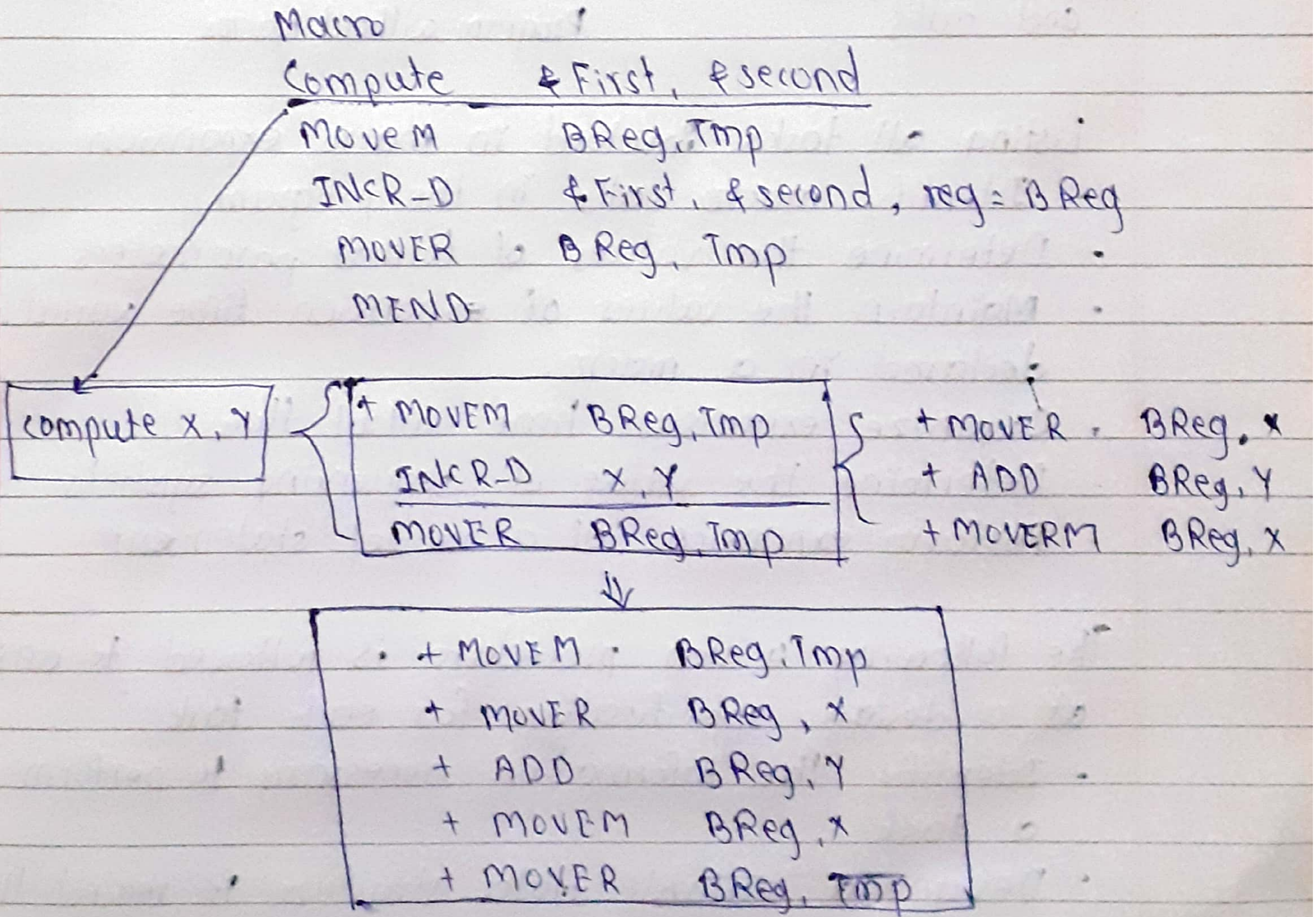
Q.3] With suitable example describe nested macro calls.

→ · A model statement in macro may constitute a call on another macro, such calls are known as nested macro calls.

- The macro containing the nested call is called outer macro
- The called macro called inner macro.

• Expansion of nested macro calls follows the last-in first-out (LIFO) rule.

- Example :



```
Macro
Compute    &First, &second
MoveM      BReg, Tmp
INCR_D     &First, &second, reg=BReg
MOVER      BReg, Tmp
MEND
```

```
compute X, Y
```

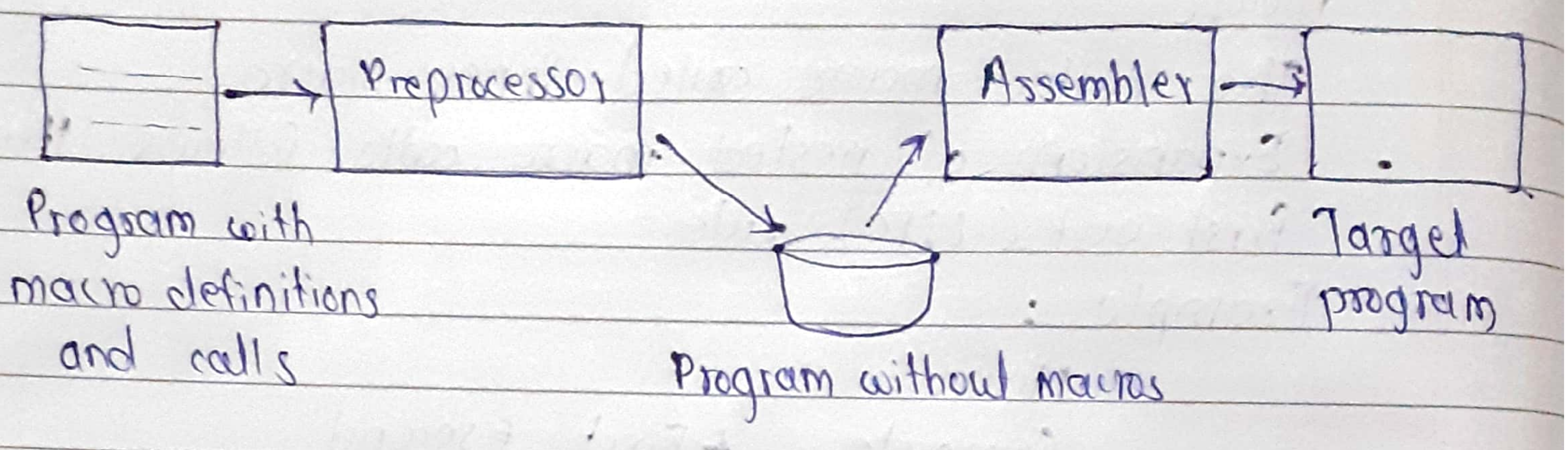
```
+ MOVEM    BReg, Tmp
INCR_D     X, Y
MOVER      BReg, Tmp
```

```
+ MOVER    BReg, X
+ ADD      BReg, Y
+ MOVERM   BReg, X
```

```
+ MOVEM    BReg, Tmp
+ MOVER    BReg, X
+ ADD      BReg, Y
+ MOVEM    BReg, X
+ MOVER    BReg, Tmp
```

Q.4] How macro preprocessors works ?

→ - The Macro preprocessor accepts an assembly program containing definitions and calls and translates it into an assembly program which does not contain any macro definition or call.



Listing all tasks involved in macro expansion

- Identify macro calls in the program.
- Determine the values of formal parameters.
- Maintain the values of expansion time variables declared in a macro.
- Organize expansion time control flow.
- Determine the value of sequencing symbols
- Perform expansion of a model statement

The following 4 step procedure is followed to arrive at a design specification for each task:

- Identify the information necessary to perform a task.
- Design a suitable data structure to record the information.
- Determine the processing necessary to obtain the information.
- Determine the processing necessary to perform the task.