Chapter 4 (2 or 3)Q
Macro Processors

1) What is macro?

Ans: Macro represents commonly used group of statements in the source program.

-> The macro processor replace each macro instruction with corresponding group of source statements.
  - this operation is called 'expanding the macro'

-> using macro allows programmer to write a shorthand version of a program.

2) Macroprocessor

-> Its function essentialy involves the substitution of one group of lines or another.

-> Design of macroprocessor generally is machine independent.

* (b) it doesnot concern with the meaning of the involved statements during macro ~~expansion~~ expansion.

-> Macro are mostly used in ~~machine~~ assembly language programming, also they are used in high level programming language such as C, C++.

-> Macroprocessor involves definition, invocation and expansion.

III 3) What is macro defination and expansion? Explain with example.

Ans: **Macro defination**

→ Two assembler directives used in macro defination are:-
  MACRO : identify the beginning of a macro defination
  MEND : identify the end of macro defination
→ Macro's name appear before the MACRO directive.
→ Macro's parameters appear after the MACRO directive.
→ Each parameter begins with '&'.
→ between MACRO and MEND is the body of macro. These are the statements that will be generated as the expansion of the macro defination

```
           Sum      MACRO    &1D    → Parameter
name of
macro               LDA      X&1D1
                    ADD      X&1D2
                    ADD      X&1103
                    STA      X&1D4
                     :
                    MEND     → end of macro
```

Macro Invocation

→ Macro invocation statement (macro call) gives the name of the macro instruction being invoked and the arguments in expanding the macro.

Macro Expansion

→ each macro invocation statement will be expanded into the statements that form the body of the macro.

→ Macro expansion is the process where macro processor replace each macro instruction with corresponding group of source statements.

→ In macro expansion, whenever a macro invocation is done, the invoked statements will be expanded to form a body of macro.

→ The arguments from macro invocation are substituted for parameters in macro prototypes.

→ The arguments & parameters are associated according their positions.

4) ~~Z~~ Data structure of Macro (Not asked)

-> 3 main data structure in macro are :-

a) Defination Table (DEFTAB)

-> The macro defination themselves are stored in defination table (DEFTAB) which contains macro prototype and statements that makeup macro body.

b) Name Table (NAMTAB)

-> for each macro instruction defined, NAMTAB contains pointers to beginning and end of defination in DEFTAB.

c) Argument Table (ARGTAB)

-> is used during expansion of macro invocation

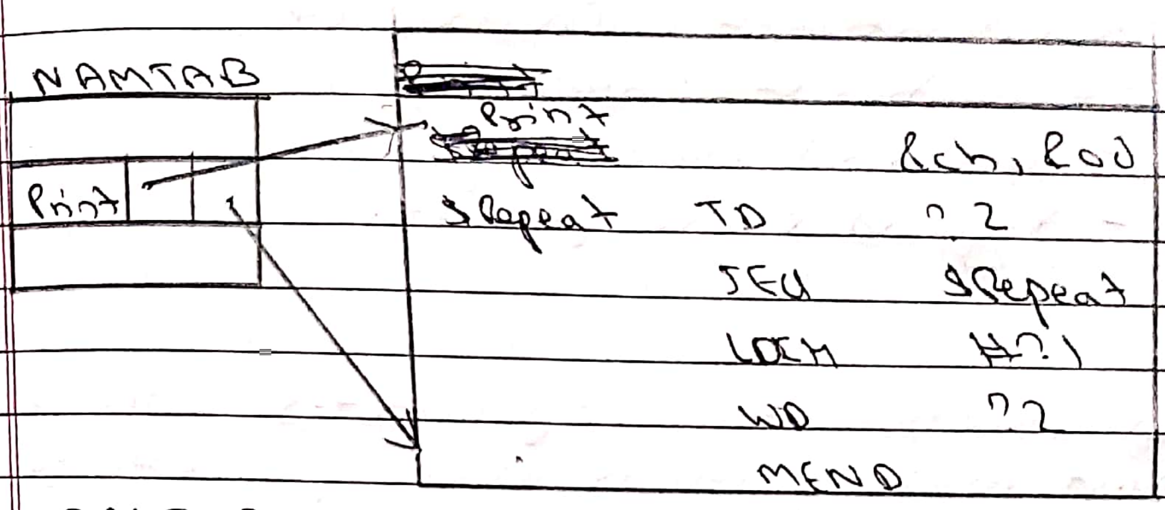
-> when macro invocation statements are recognized, the arguments are stored in ARGTAB according to their position in argument list.

-> As the macro is expanded, argument from ARGTAB are substituted for the corresponding parameters in macro body.

Example :- "Print 64,41" {This will be asked as separate question}

print	MACRO	&ch, &od
$Repeat	TD	&od
	JEQ	$Repeat
	LDCH	# &ch
	WD	&od
	MEND	

Data Structure



```
~~Print~~
Print                  &cb, Rod
~~Repeat~~
$Repeat     TD         ?2
            JEQ        $Repeat
            LDCH       #?1
            WD         ?2
            MEND
```

ARGTAB

1	64
2	F1

Expansion

```
.print                     64,F1
$AARepeat     TD           F1
              JEQ          $AARepeat
              LDCH         #64
              WD           F1
```

5) Machine Indepent Macro Processor Features
- Concatenation of Macro Parameters
- Generation of Unique labels
- Conditional Macro Expansion
- Keyword Macro parameters

6) Concatenation of macro parameters with example.

Ans: Most macro processors allow parameters to be concatenated with other character string

-> Say, a program contains one series of variable named by the symbols XA1, XA2, XA3, ... another series named by XB1, XB2, XB3 ... , etc.

-> the body of macro ~~statement~~ defination might contain statements like

```
SUM    MACRO   &ID
       LDA     X&ID1
       ADD     X&ID2
       ADD     X&ID3
       STA     X&IDS
        :
       MEND
```

-> here, the begining of macro parameter is identified by starting symbol ~~is~~ '&', however the end of parameter is not marked.

Scanned with CamScanner

-> The problem is that the end of the parameter is not marked so, X&ID1 may mean 'X' + &ID+1 or 'X' + &SD1

-> So avoid this ambiguity, a special concatenation operator '->' is used.

-> Now, new form becomes X&ID->1
here, '->' will not appear in macro expansion.

eg:-

```
SUM    MACRO    &ID
       LDA      X&ID->1
       ADD      X&ID->2
       ADD      X&ID->3
       STA      X&ID->S
       MEND
```

Now,

SUM A	SUM BETA
⇓	⇓
LDA XA1	LDA XBETA1
ADD XA2	ADD XBETA2
ADD XA3	ADD XBETA3
STA XAS	STA XBETAS

Ans

classmate
Date
Page

Page No: 102 of Note

IIIII 7) Conditional Macro expansion with example
or
IIIII What is macro time variable? How macro processor manages value of macro time variable?

Ans: So far, when a macro instruction is invoked, the same sequence of statements are used to expand macro.

-> here, depending on the arguments supplied in the macro invocation, the sequence of statements generated for macro expansion can be modified.

-> this adds to ~~the~~ greatly to power and flexibility of macro language.

-> Macro time variable

-> it is a variable that begins with '&' and that is not a macro instruction parameter.

-> can be initialized to a value of 0.

-> can be set by macro processor directive 'SET'

-> can be used to
   (i) store working values during expansion.
   (ii) store the evaluation result of Boolean Expansion
   (iii) Control macro time conditional structure.

-> Macro time Conditional Structure
   i) IF - ELSE - END IF
   ii) WHILE - ENDW

Scanned with CamScanner

Implementation of conditional macro Expansion

i) IF-ELSE-ENDIF

-> Firstly a symbol table is maintained by macro processor that contains the values of all macro time variables used.

-> entries in this table are made or modified when SET statements are processed.

-> this table is used to look up the current value of macro time variable whenever it is required.

-> the testing of condition and looping are done while macro is being expanded.

-> When an IF statement is encountered during the expansion of a macro, the specified boolean expression is evaluated

-> if value is TRUE

- the macro processor continues to process lines from DETAB until it encounters the next ELSE or ENDIF statement.
- if ELSE is encountered, then skips to ENDIF

-> if value is FALSE

- the macroprocessor skips ahead in DEFTAB until it finds the next ELSE or ENDIF statement.

P.T.O ->

Scanned with CamScanner

ii) WHILE - ENDW

-> When a WHILE statement is encountered during the expansion of a macro, the specified Boolean expression is evaluated.

-> if value is TRUE
   - the macroprocessor continues to process lines from DEFTAB until it encounters next ENDW statement.
   - when ENDW is encountered, the macro processor returns to the preceding WHILE, reevaluates boolean expression and take action again.

-> if value is FALSE
   - the macroprocessor skips ahead in DEFTAB until it finds the next ENDW statement and then resumes normal macro expansion.

Example: (optional)

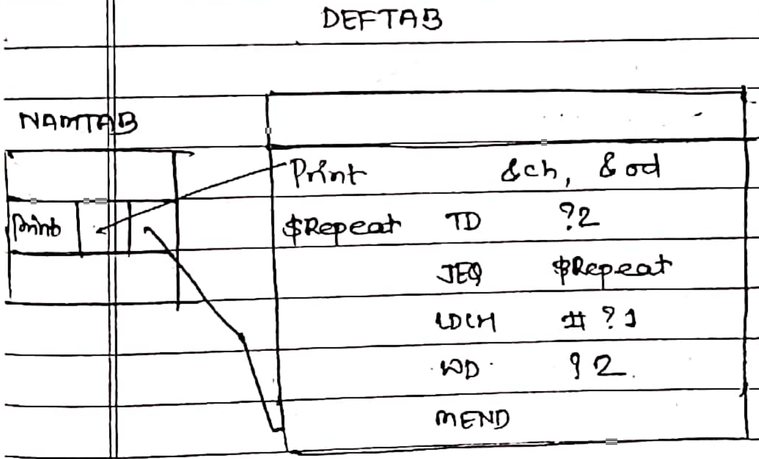
```
OPERATE    MACRO    &ID
           LDA      X&ID1
&COND      SET      1
           IF       (&COND1)
           ADD      X&ID2
           ADD      X&ID3
           ELSE
              SUB   X&ID2
              SUB   X&ID3
           ENDIF
              STA   X&IDS
           MEND
```

~~Lon~~ Assignment

1. Consider the macro definition given below and show macro expansion for the macro call statement "print 64, F1". Show all data structures used by macro processor clearly.

```
print     MACRO     &ch, &od
$Repeat   TD        &od
          JEQ       $Repeat
          LDCH      # &ch
          WD        &od
          MEND
```

Fig // Data structure



Print		&ch, &od
$Repeat	TD	?2
	JEQ	$Repeat
	LDCH	# ?1
	WD	?2
	MEND	

ARGTAB

1	64
2	F1

Date
Page

Pg 2

Expansion

```
• print          64, F1
$AARepeat  TD    F1
           JEQ   $AARepeat
           LDCH  # 64
           WD    F1
```