Assignment-3

(1) Define a macro, macro call and macro expansion

Ans Macro - It starts with macro pseudo-op
It indicates the begining of a micro definition
It is terminated with the MEND pseudo-op

Macro call-
Once the macro has been defined the use of macro name as an operator mnemonic in an assembly program is equivalent to the corresponding instruction sequance are written with the macro name. This is called macro call

macro expansion - Whenever there is a macro call the macro processor substitutes the macro definition in the place of the macro call. This is called macro expansion. The macro definition itself doesnot appear in the expanded source code.

(2) What are the feature of a macro ?
Explain conditional macro expansion ?

Ans Features of macro are
(1) macro instruction arguments
(2) conditional macro expansion
(3) macro calls with macros
(4) conditional macro expansion

Ans The sequence of macro expansion can be recorded based on some conditions macro expansion
It allows condition selection of the machine instruction that appear in expansion of a macro call.
For this we use macro pseudo-ops AIF and AGO
AIF
It is a conditional branch pseudo-ops.
The format of AIF statement is as follow

AIF (expression) (sequence symbol)

AGO

AGO is an unconditional branch pseudo-op

The format of AGO statement is as follow

AGO <Sequencing Symbol>

Example: consider the following program

```
        :
Loop 1  A   1 Data 1
        A   2 Data 2
        A   3 Data 3
Loop 2  :
        A   1 Data 1
        A   2 Data 2
        :
        :
Loop 3  A   1, Data 1
Data 1  DC  F 5
Data 2  DC  F 10
Data 3  DC  F 15
```

In this example the operands, labels and the number of instruction generated change in each sequence. This program could be written as follow

```
&ARG10          MACRO
&ARG10          VARY    &COUNT &ARG1, &ARG2, &ARG3
                A       1, &ARG1
                AIF     (&COUNT EQ 1).FINI      Test if &COUN
                A       2 &ARG2
                AIF     (&COUNT EQ 2.)FINI      Test if &COU
                A       3, &ARG3
FINI            MEND
                .
                .
LOOP1           VARY    3 ,DATA1, DATA2 ,DATA3
                .
                .
LOOP2           VARY    2 DATA  3 DATA
                .
                .
                .
DATA1           DC F'5'
DATA2             F'10'
DATA3             F'15'
```

Expanded Zone

```
                .
                .
LOOP1           A   1 Data1
                A   2 Data2
                A   3 Data3
                .
                .
LOOP2           A   1 DAata3
                A   2 DATA2
                .
                .
LOOP3           A   1, DATA1
```

(3) ... statements, AGO

Ans AIF Statements

It is a conditional branch pseudo-op

The format of AIF statements is as follow

AIF <expression> <Sequencing symbol>

where expression is a related expression involving strings, format parameters, and their attributes.

AGO Statements

It is an unconditional branch pseudo-op.

The format of AGO statements is as follows.

AGO <Sequencing symbol>

It unconditionally transfers control to the statement containing Sequencing symbol in its label field.

AGO statement doesnot appear in the expanded source code

(4) Explain 2 pass macro-processor algorithm.

Ans While implementing two pass macro-processor algorithm the following assumption are made.

(1) Macro processor is functionally independent of the assembler

(2) Output from the macro processor will be fed into the the assembler

(3) Macro definition or call with in macro definition are not allowed because of its complications.

Like of an assembler, macro processor also scans and processes line of a text.

Like of an assembler, micro processor also scans and processes line of a text

The lines of macro language are not so closely interrelated as in assembler i.e macro definition refer to nothing outside themselves and macro call refer only to macro definition.

(4) A macro call substitutes text for parameter not values.

Example: Consider the following statements

```
INCR  Y
Y EQU 10
```

The macro call INCR Y is followed by a statement designing X

The macro processor substitutes Y for the macro definition argument

It will not assign a value 10 to Y because the EQU statement is not processed by the macro processor If consider it as a complete line of text

Assignment - 4

(1) Define loader & what are its function

The loader is program which accept the object program decks, prepares these program for execution by the computer and initiates the execution.

(*) Function of loader

(1) Allocation:
allocate space in memory for the programs

(2) Linking:
Resolve symbolic reference between object decks

(3) Relocation
Adjust all address dependents locations such as address constants to corresponds to the allocated space

(4) Loading:
physically place the machine instruction and data into memory.

(2) Name different types of loading scheme?

Ans different types of loading schemes are

(1) compile and go loader
(2) General loader scheme
(3) Relocating loaders
(4) Direct linking loaders
(5) Dynamic loading
(6) Dynamic linking.

(3) explain dynamic loading and dynamic binder

Dynamic loading -

If the total amount of core required by all these subroutines exceeds the amount available, as is common with large program & small computers, there is trouble

There are several hardware techniques, such as paging and segmentation, that attempt to solve this problem!

In this section we will present conventional dynamic loading schemes based upon the use of a binder prior to loading.

Dynamic Binder

- A binder is a program that performs the same function as direct linking loader in binding subroutine together but rather than placing the relocated and linked text directly into memory, its output (the text as a file or card deck.

(*) The output file is in a format ready to be loaded and is typically called a load module.

(*) The loader merely has to physically load the module into core

(*) The binder essentially performs the functions of allocation, relocation and linking: the module loader merely perform the function of loading.

* There are two major classes of binders
(a) Core image builder
(b) Linkage editor

(4) Explain Four types of cards used in direct linking loader

Ans External Symbol Dictionary cards.

→ The ESD card contains the information necessary to build the external symbol dictionary or symbol table.

→ External symbol are the symbol that can be referred beyond the subroutine level

→ There are three types of external symbols
(a) Segment Definition (SD) - name on START
(b) Local Definition (LD) - Specified on Entry card
(c) External Reference (ER) - Specified on EXTERN card

(2) TXT Cards
The txt cards contain block of data and the relative address at which the data is to be placed

→ once the loader is divided where to load the program it merely adds the program loader address (PLA) to the relative address and moves the data into the resulting location

→ The data on the TXT card may be instructions, non relocated data or initial values of address constants.

→ The RLD cards
The RLD cards contain the following information.
(a) The location and length of each address constant that needs to be changed for relocation or linking

(b) The external symbol by which the address constant should be modified (added or subtracted)

(c) The operation to be performed (add or subtract)

(3) END cards.

(a) The end card specifies the end of the object

(b) If the assembler END card has a symbol in the operand field it specifies a start of execution point for the entire program (all subroutines)

(c) This address is recorded on the END card.

(d) This is a final card required to specify the end of a collection of object

(e) The 360 loaders usually either a loader terminate (LDT) or END of file (EOF) card

(5) Write specification of database used in pass 1 and pass 2 of Direct link loader.

Ans The database use in pass 1 and pass 2 of Direct linking loader are

(a) pass 1 data passes

(1) Input object decks.

(2) A parameter the intial program load Address (IPLA) supplied by the programme or the operating system that specifies the address to load the first segment.

(3) A program Load Address (PLA) counter, used to keep track of each segments assigned location

(4) A table, the Global External symbol table (GLEST) that is used to store each external symbol and its corresponding assigned core address

(5) A copy of the input to be used later by pass 2

(6) A printed listing. the load map that specifies each external symbol and its assigned value.

Pass 2 data bases

(1) Copy of object programe inputted to pass 1
(2) The intial program load Address parameter (IPLA)
(3) The program Load address counter (PLA)
(4) The GEST, prepared by pass 1 containing each external symbol and its corresponding absolute address value.
(5) An array the local External Symbol Array (LESA) which is used to Establish a correspondence between the ESD ID numbers used on ESD and RLD cards and the corresponding external symbols absolute address value.