SNS COLLEGE OF ENGINEERING

(Autonomous)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

19SB405 – MICROPROCESSORS AND ADVANCED MICROCONTROLLERS

Introduction

Assembler: is a program that accepts an assembly language program as input and converts it into an object module and prepares for loading the program into memory for execution.

Loader (linker) further converts the object module prepared by the assembler into executable form, by linking it with other object modules and library modules.
The final executable map of the assembly language program is prepared by the loader at the time of loading into the primary memory for actual execution.
The assembler prepares the relocation and linkages information (subroutine, ISR) for loader.

Thus the basic task of an assembler is to generate the object module and prepare the loading and linking information.

Assembler Directives

- Pseudo operations that control the assembly process.
- They indicate how an operand or section of a program to be processed by the assembler.
- They generate and store information in the memory.

Directives

DB (DEFINE BYTE)

The DB directive is used to declare a byte type variable, or a set aside one or more storage locations of type byte in memory.

Ex - PRICES DB 49H, 98H, 29H;Declare array of 3 bytes named PRICES and initialize them with specified values.

DD (DEFINE DOUBLE WORD)

The DD directive is used to declare a variable of type double word or to reserve memory locations, which can be accessed as type double word.

Example:

ARRAY DD 2562 9261H

                0010010101100012

DQ (DEFINE QUADWORD)

The DQ directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage in memory.

Example: BIG_NUMBER DQ 2435 9874 0192 A92B H

Directives

DT (DEFINE TEN BYTES)

The DT directive is used to tell the assembler to declare a variable, which is 10 bytes in length or to reserve 10 bytes of storage in memory.

Example:

PACKED_BCD DT 11 22 33 44 55 66 77 88 99 00

DW (DEFINE WORD)

The DW directive is used to tell the assembler to define a variable of type word or to reserve storage locations of type word in memory

ASSUME

ASSUME tells the assembler what names have been chosen for Code, Data Extra and Stack segments.

Example

ASSUME CoSeg: Name of code segment

Directives

ENDS (END SEGMENT)

This directive is used with the name of a segment to indicate the end of that logical segment.

CODE SEGMENT; Start of logical segment containing code instruction statements; CODE ENDS; End of segment named CODE

PROC

ENDP (END PROCEDURE)

The ENDP directive is put after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statements after an ENDP directive, so you should make sure to use only one END directive at the very end of your program module. A carriage return is required after the END directive.

EQU (EQUATE)

EQU is used to give a name to some value or symbol. Each time the assembler finds the given name in the program, it replaces the name with the value or symbol you equated with that name.

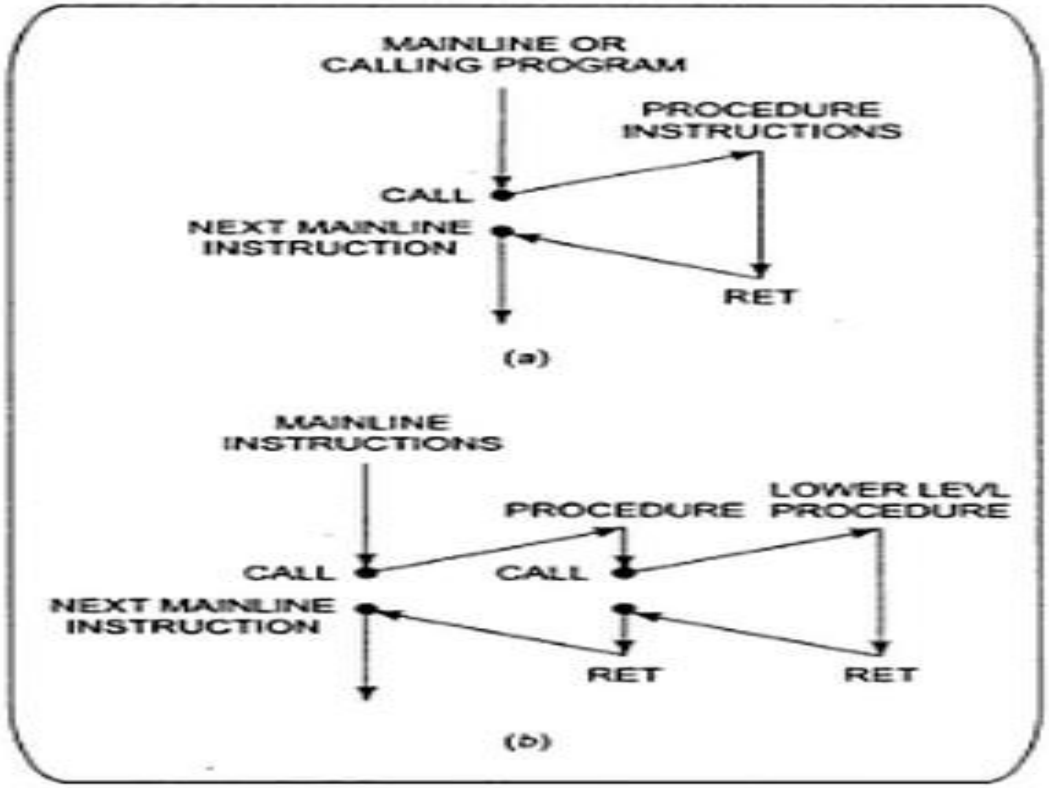
Clear _carry EQU CLC

PROCEDURE

A procedure is group of instructions that usually performs one task. It is a reusable section of a software program which is stored in memory once but can be used as often as necessary.

.

.



TYPES OF PROCEDURE

A procedure can be of two types.

1) Near Procedure
2) Far Procedure
    - Near Procedure: A procedure is known as NEAR procedure if is written(defined) in the same code segment which is calling that procedure. Only Instruction Pointer(IP register) contents will be changed in NEAR procedure.
    - FAR procedure : A procedure is known as FAR procedure if it is written (defined) in the different code segment than the calling segment. In this case both Instruction Pointer(IP) and the Code Segment(CS) register content will be changed.

.

MACRO

A Macro is a group of instructions with a name. When a macro is invoked, the associated set of instructions is inserted in place in to the source, replacing the macro name. This“macro expansion” is done by a Macro Preprocessor and it happens before assembly. Thus the actual Assembler sees the “expanded” source!

Macro: When a macro is “invoked”, the corresponding text is “inserted” in to the source. Thus multiple copies exist in the memory leading to greater space requirements.

However, there is no execution overhead because there are no additional call and return instructions. The code is in-place.

MACRO Definition:

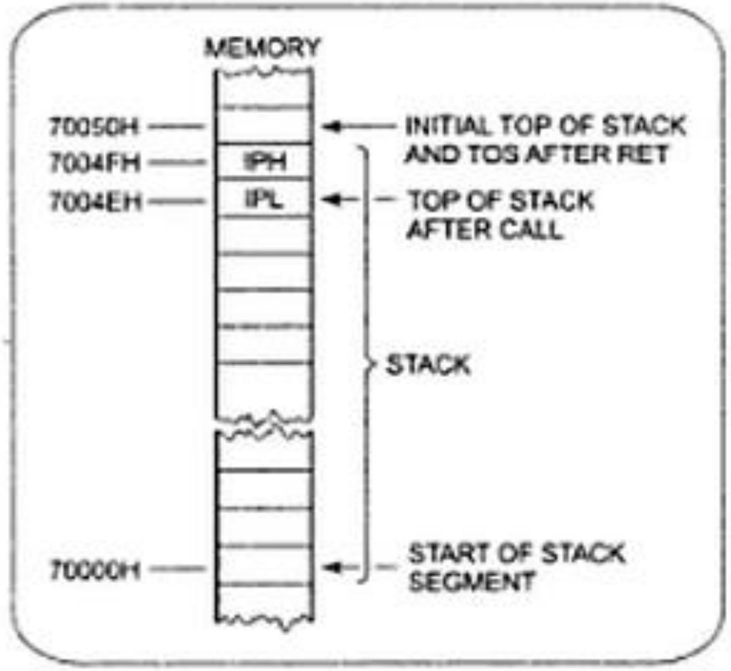
A macro has a name. The body of the macro is defined between a pair of directives, MACRO and ENDM.

PROCEDURE Vs MACRO

Procedure	Macro
Procedures are used for large group of instructions to be repeated.	Procedures are used for small group of instructions to be repeated.
Object code is generated only once in memory.	Object code is generated everytime the macro is called.
CALL & RET instructions are used to call procedure and return from procedure.	Macro can be called just by writing its name.
Length of the object file is less	Object file becomes lengthy.
Directives PROC & ENDP are used for defining procedure.	Directives MACRO and ENDM are used for defining MACRO
More time is required for it's execution	Less time is required for it's execution
Procedure can be defined as Procedure_name PROC ---- ------ Procedure_name ENDP	Macro can be defined as MACRO-name MACRO [ARGUMENT ,......... ARGUMENT N] ------ ------- ENDM
For Example Addition PROC near ------ Addition ENDP	For Example Display MACRO msg --------- ENDM

STACK

- Section of memory you set aside for storing return addresses.
- Also used to store the contents of the registers for the calling program while a procedure executes.
- Hold data or address that will be acted upon by procedures.



1. Explain ISR

THANK YOU