# Unidad III: Modularización

### 3.1 Procedimientos

Es una colección de instrucciones que realizan una tarea específica. Dependiendo de su extensión y complejidad, un programa puede contener uno, algunos o inclusive cientos de procedimientos. Para emplear un procedimiento en un programa se requiere definirlo y llamarlo. Al definir un procedimiento escribimos las instrucciones que contiene. Al llamar al procedimiento transferimos el fluijo al procedimiento para que sus instrucciones se ejecuten. Se define como:

*PROC nomProc*
*proposicion*
*[proposicion]*
*...*
*ENDP [nomProc]*

La llamada a un procedimiento tiene la siguiente forma:
*CALL nomProc*

Pra regresar de un procedimiento se utiliza
*RET*

Un buen procedimeinto debe:

- Hacer solo una tarea.
- Ser tan pequeño como sea posible y tan largo como sea necesario.
- Contener un comentario con su proposito, datos de entrada y salida.
- Entenderse por si solo.
- Funcionar como lo haría una instrucción del microprocesador.
- No usar variables globales ni para recibir datos, ni regresar un resultado, ni almacenar temporalmente resultados intermedios.

### 3.2 Macros

Es un conjunto de instrucciones asociadas a un identificador: el nombre de la macro.
Este conjunto de instrucciones es invocado como una sola instrucción o macroinstrucción. Para emplear una macro en un programa debemos de definir la macro e invocar la macro.
La definición de una macro establece el nombnre al que se le asocia la macro, el número y nombre de sus parametros formales y qué instrucciones contiene la macroinstrucción. La sintaxis de la definiocion de una macro es la siguiente:

*MACRO nombMacro [parForm[, parForm]...]*
*proposición*
*[proposición]*
*...*
*ENDM [nombMacro]*

Aunque la definición de una macro puede ir en cualquier parte de un programa, el lugar más recomendable para su localizaci{on es al principio de un archivo, antes de los segmentos de datos y de código.
Al encotrar una invocación de una macro, el macroensamblador substituye la línea con la invocación por las proposiciones que contiene la definición de la macro. Este proceso de substitución se conoce como expansión de la macro. La sintaxis de la invocación de la macro es similar a cualquier instrucción:
*nomMacro [parReal[, parReal]...]*

donde cada parReal es conocido como un parametro real de la macro. Al expandirse la macro cada una de las ocurrencias de un parametro formal en la definicion de la macro se substituye por su correspondiente parámetro real.