LECTURAS SOBRE COMPUTADORAS DIGITALES –LECTURA Nº6
MATERIA: ARQUITECTURA DE LAS COMPUTADORAS

SOFTWARE

1.- DEFINICIÓN

El software es el conjunto de programas y documentación que permiten el adecuado funcionamiento, operación y programación de las computadoras.

A partir de 1965 con la aparición de la IBM /360, las computadoras se diseñaron para funcionar con parte física (hardware) y parte lógica(software) imprescindiblemente.

Se puede afirmar que una computadora es una construcción indivisible de hardware y software, que no funciona correctamente o no funciona sencillamente, si una cualquiera de las partes se encuentra dañada total o parcialmente.

2.- CLASIFICACIÓN

Desde el punto de vista de programas, el software se clasifica en:

- Software de base
- Software del usuario
- Software de ambiente o de productividad.

2.1.- Software de Base

El software de Base incluye:

a) Sistemas Operativos
b) Lenguajes

2.1.1.- Sistemas Operativos

2.1.1.1 Definición: El Sistema Operativo (SO de aquí en más) es un conjunto de programas que, asociados al hardware, realiza la gestión de datos y la administración de tareas de una computadora.

Cuando se dice que el SO , "realiza la gestión de datos", significa que es el mismo el que busca, despliega y archiva los datos ya sean estos, archivos o programas, en su lugar de almacenamiento, sea este la memoria principal o una memoria auxiliar, como puede ser un disco rígido, un disco flexible, un pen driver, un CD-ROM, un DVD u otro elemento de almacenamiento masivo de datos.

Cuando se dice que el SO, "realiza la administración de tareas", significa que el mismo genera las señales eléctricas necesarias para vincular las partes de la computadora, ya sean internas o externas que se requieren para realizar una determinada tarea.

Otra Definición: También se puede decir que el S.O. es el programa de control maestro de la computadora (Peter Norton).

Una computadora sin sistema operativo no puede, ni siquiera, comenzar a funcionar pues de esta manera han sido concebidas desde 1965 hasta la actualidad.

2.1.1.2 Clasificación: Los S.O. interactúan con los usuarios de una computadora a través de la pantalla del monitor, la misma constituye la Interfaz del usuario.

Los S.O., desde el punto de vista del manejo del monitor, pueden ser de "MODO TEXTO" o de "MODO GRÄFICO".

El SO que trabaja en modo Texto, más popular y extendido es el DOS de Microsoft. El mismo, según la versión, requiere de una memoria principal de 1(un) Megabyte para poder trabajar correctamente, 4 MBy para un funcionamiento estándar en versiones de mediana complejidad y 8 MBy en las últimas versiones de alta complejidad. Como puede verse, requiere de poca memoria principal.

Hay que tener en cuenta que el DOS funcionaba con monitores monocromáticos, con una paleta de 4 colores o una paleta de 16 colores.

Los SO que trabajan en Modo Gráfico como el Windows de Microsoft requieren de mucha memoria, es decir, de 128MBy en adelante, según la versión y la complejidad del mismo.

El requerimiento de memoria está directamente relacionado con el control de la pantalla del monitor y de los programas que componen el SO.

El SO de modo gráfico tiene que controlar cada pixel(puntito) de la pantalla en lo que respecta a su ubicación espacial (coordenadas), color, intensidad y brillo. Además de una paleta de millones de colores.

Los SO de modo texto trabajan digitando instrucciones, por ejemplo, si se desea Imprimir un archivo habrá que escribir PRINT PEPE.BAS, y así para cada función que se quiera realizar, por este motivo, se dice que generan un entorno de comandos.

Los SO de modo gráfico trabajan cliqueando sobre íconos en un menú que se encuentra en alguna de las ventanas del monitor. Estos sistemas son más fáciles de usar que los de modo texto, por ese motivo se dice que son más "amigables". Por la disposición de las prestaciones en pantalla, se dice que generan un entorno de ventanas.

2.1.1.3 SOPORTE FÍSICO DE LOS SO

Los programas del SO son activos cuando se encuentran en la memoria principal pero debido a que gran parte de la misma es volátil, se diseñó un a estrategia de ahorro de recursos y eficiencia de operación.

Parte del SO se encuentra residiendo en la ROM (parte de la memoria principal) y parte en un medio de almacenamiento masivo, habitualmente en un disco duro o rígido.

El programa que se encuentra en la ROM se denomina BIOS (Basic Input Output System), por eso a la ROM se la llama ROMBIOS. Múltiples son las funciones de la misma, pero, tal vez, la tarea clave es el momento del arranque de la computadora.

En ese momento, la BIOS pone en funcionamiento un programa que le permite cargar otro programa que se encuentra en la Memoria Auxiliar a la Memoria RAM, llamado genéricamente "bootstrap" que significa "tirabotas" , como una parábola que el sistema se autoeleva. El Bootstrap, que tiene distintos nombres según el SO, carga el resto del sistema en Memoria Principal y la máquina queda lista para operar.

Antes de convocar al bootstrap (nótese que el volver a arrancar la computadora se denomina "buteo"), el BIOS tiene un programa de chequeo de la arquitectura de la computadora y su

contrastación contra el inventario que se encuentra en la memoria del Set Up, en caso de no coincidir, emite mensajes para que el usuario tome los recaudos que crea que son necesarios.

Además la BIOS, tanto la parte que está en ROM, como otra parte que se agrega después del buteo en memoria RAM, intervienen en las operaciones de conexión de entrada y salida de datos e información.

Los programas del SO operativo suelen tener comandos internos, que se encuentran durante toda la operatoria en memoria principal, y comandos externos, que son convocados a memoria principal en el momento que sea necesario utilizarlos y luego de ser usados, el lugar de memoria que era ocupado por estos comandos se puede utilizar para otra aplicación.

2.1.1.4 CONTROLADORES

Cada elemento que se agrega a la estructura de una computadora tanto en forma interna o en forma externa (periféricos) tiene que ser reconocido por el sistema, a tal fin existe programas llamados drivers o handlers o controladores, que pueden pertenecer al SO o ir agregándose a medida que se incorporan nuevas prestaciones.

Se puede decir que el SO mantiene organizado el hardware de una computadora pues comunica la CPU con los demás elementos del hardware y actúa como intermediario entre el software y el hardware usando controladores.

Debido a que el usuario trabaja con el hard a través del SO se dice que "ve" una máquina virtual pues no tiene un contacto directo con los componentes internos de la estructura física.

2.1.1.5 COMENTARIO FINAL

Una de las tareas más complejas que enfrenta el SO en una máquina es la administración de las memorias, tanto principal como auxiliar, al punto tal que se señala que el sistema DOS de Microsoft estuvo concebido fundamentalmente para manejar adecuadamente a los discos rígidos, de ahí su nombre de Disk Operation System, que podría pensarse como el Sistema Operativo para el Disco, posteriormente se opinó que era el Sistema Operativo en Disco.

De todas formas es cierto, también, que los SO han sido diseñados para trabajar con el hardware de la computadora y realizar las tareas del usuario con la mayor facilidad. Por este motivo también se dice, que el usuario no ve a la máquina real sino que ve a una máquina virtual, precisamente la que el SO le muestra.

Asimismo, desde los inicios hasta la actualidad, los SO han ido evolucionando y cambiando sus prestaciones, siempre procurando facilitar la tarea del operador.

Sabido es que la empresa líder en el mercado es Microsoft con sus SO Windows 3x, 9x, CE, NT, XP, etc en el momento actual y con el DOS en el pasado. Pero también existen los SO de Apple, el Macintosh es el más usado. IBM con su OS/2 y AS 400, así como el UNIX y el pionero del software libre: el LINUX.-

2.1.2.- Lenguajes

2.1.2.1. Clasificación

Los lenguajes se clasifican según el grado de detalle que el programador tiene que realizar respecto de la estructura física de la computadora.

Con este criterio:

Lenguajes

- **De máquina (el único que la máquina entiende)**
- **De bajo nivel**
- **De alto nivel**

Lenguaje de máquina

Un lenguaje de computación es un medio de comunicación entre el usuario y la computadora, pero también es el software que transforma lo escrito de una manera determinada en el lenguaje de máquina.

En la Lectura 4, se propuso el problema siguiente:

Problema Nº 1:
Escribir en lenguaje de máquina

MEMORIA	CONTENIDO
0	STR 10
1	SUB 10
2	ADD 9
3	ADD 8
4	SUB 7
5	WRT 0
6	STP 5
7	05
8	02
9	07
10	02

REGISTRO	CONTENIDO
ACUM	6
PC	0

SOLUCIÓN:

MEMORIA	CONTENIDO
00000	01101010
00001	01001010
00010	00101001
00011	00101000
00100	01000111
00101	11100000
00110	00000101
00111	00000101
01000	00000010
01001	00000111
01010	00000010

REGISTRO	CONTENIDO
ACUM	00000110
PC	00000

Las tareas del Programador

El único lenguaje que la máquina entiende es el que se encuentra consignado en el párrafo que dice “solución”, en el cual el Programador debió:

1. Pensar cómo se resuelve el problema planteado (búsqueda de un algoritmo de solución)
2. Pensar cómo la máquina resuelve el problema planteado. Para ello debe conocer perfectamente la arquitectura interna de la computadora.

3. Dar direcciones de memoria principal a cada dato o instrucción, se las llama direcciones absolutas y, de esta forma, se está "administrando" a la memoria principal.
4. Escribir el programa y los direccionamientos que correspondan, así como los contenidos de los registros en el código binario de la máquina, en este caso el lenguaje propio de la PetiComputadora.
5. Conocer el lenguaje en el cual programa.

Desde el punto de vista operativo, si tenemos el lenguaje escrito en lenguaje de máquina, programa que se denomina PROGRAMA OBJETO o PROGRAMA EJECUTABLE y al cual llamamos, en este caso, Pepe.Exe, basta colocarlo en la máquina y dar la orden de ejecutar para obtener la solución del problema, tal como puede verse en la figura 1.

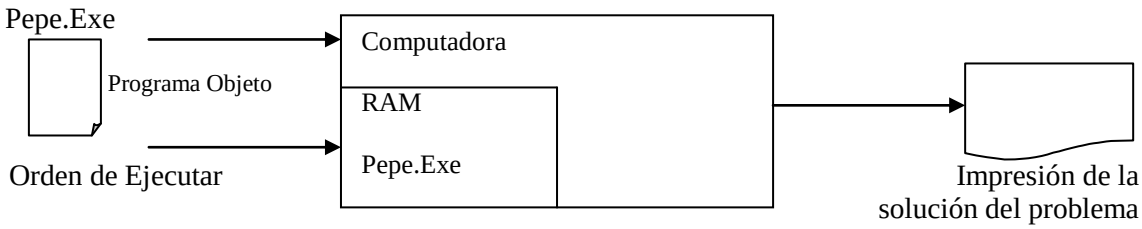


Figura 1: Operación de un programa escrito en lenguaje de máquina

Naturalmente que realizar los cuatro pasos que debe realizar el programador es factible si el problema a resolver es sencillo y la computadora de muy baja complejidad, pero a medida que las computadoras se tornaron más complejas y se buscó facilitar el trabajo del programador mediante otros lenguajes, primeramente fueron:

Los Lenguajes de bajo nivel

Los lenguajes de bajo nivel facilitan la tarea del programador pero no mucho, progresivamente los podemos clasificar en:

- Lenguajes numéricos (octal y hexadecimal)
- Assembly
- Assembly simbólico
- Macro Assembler V1
- Macro Assembler V2

Lenguajes numéricos

El programa del "Problema Nº 1" de página 4 puede escribirse en hexadecimal de la manera siguiente:

MEMORIA	CONTENIDO
00	6 A
01	4 A
02	2 9
03	2 8
04	4 7
05	E 0
06	0 5
07	0 5
08	0 2
09	0 7
0A	0 2

REGISTRO	CONTENIDO
ACUM	0 6
PC	00

El programa que se acaba de escribir se denominará en forma genérica "Programa Fuente escrito en hexadecimal" y la máquina no lo entiende, se lo llamará Pepe.Hex.

Se puede definir como "Programa Fuente" todo programa escrito en cualquier lenguaje que no sea el lenguaje de máquina.

Operativamente el Programa Fuente requiere de un Programa Traductor que lo transforme en el Programa Objeto.

El Programa Traductor es un programa utilitario que el usuario debe comprar al fabricante de la computadora que lo llama "Compilador", en este caso, "Compilador Hexadecimal".

El compilador es un programa objeto que toma como dato al Programa Fuente y entrega como solución del problema al Programa Objeto.

Las etapas operativas van a ser las de la figura 2.

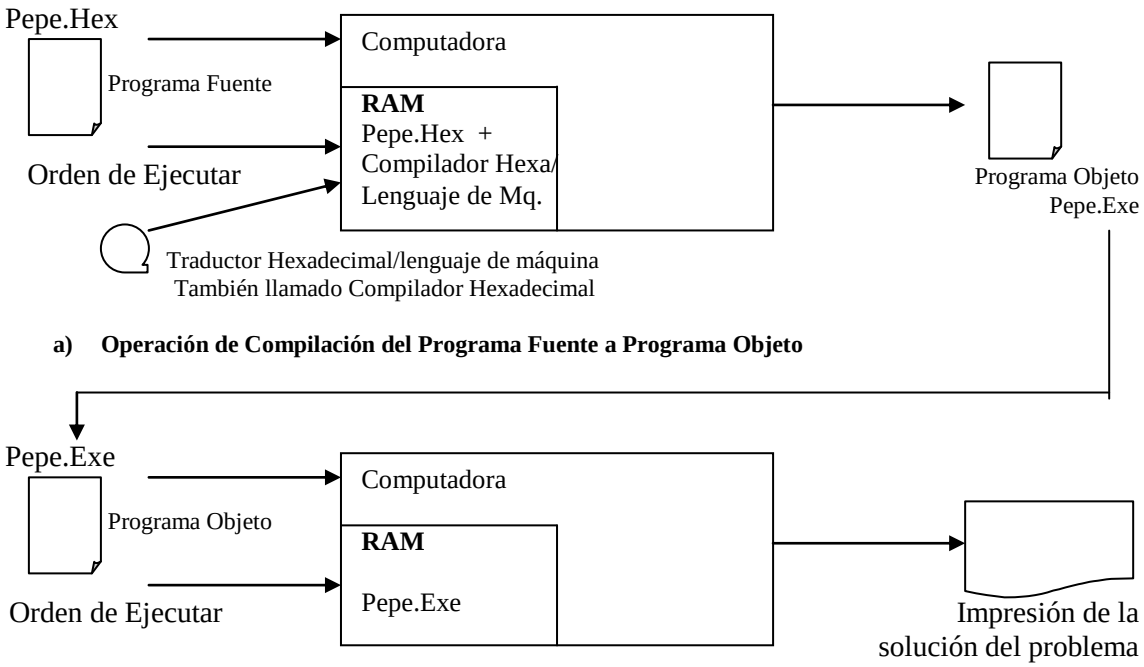


Figura 2: Operación de un programa escrito en lenguaje numérico

Como puede verse el programador se "aleja" de la máquina y las tareas que antes el realizaba, como la conversión del programa hexadecimal en binario, la tiene que hacer un programa, en este caso el compilador hexa/binario.

Nótese que en el momento de la ejecución del programa objeto, en memoria principal basta que se encuentre el mismo y ningún otro programa

Cuando se trabaja en el lenguaje numérico, en este caso hexadecimal, el programador se exime del listado de "Las tareas el programador", sólo de las indicadas en el punto 4.

Lenguaje Assembly

El lenguaje Assembly es un avance respecto al lenguaje numérico pues el programador lo hace en un lenguaje que está más cerca del humano, el lenguaje de mnemónicos.

Si volvemos al Problema Nº 1, tenemos como dato un programa Assembly, un programa fuente escrito en Lenguaje Assembly, el mismo, a fin de obtener la solución del problema, debe ser convertido en programa objeto. Esto se logra a través de los pasos que se describen en la Figura 3.

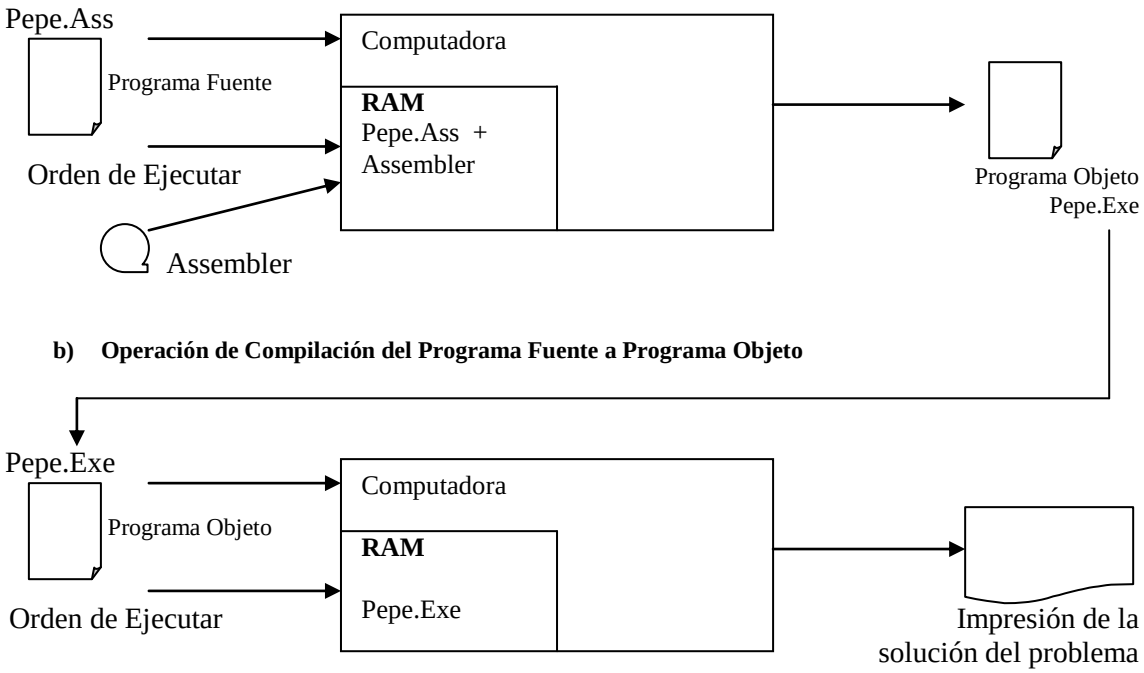


Figura 3: Operación de un programa fuente escrito en Assembly

El Assembler es el compilador de un programa fuente escrito en Lenguaje Assembly. El compilador toma como datos las sentencias y datos del fuente y las transforma en sentencias y datos en objeto.

El compilador Assembler es un programa objeto, pues puede ser ejecutado para realizar la tarea de "traducción".

Nótese que en el momento de la ejecución del programa objeto, en memoria principal basta que se encuentre el mismo y ningún otro programa

Cuando se trabaja en el lenguaje Assembly, el programador se exime del listado de "Las tareas el programador", sólo de las indicadas en el punto 4. Pero trabaja en un lenguaje más "humano".

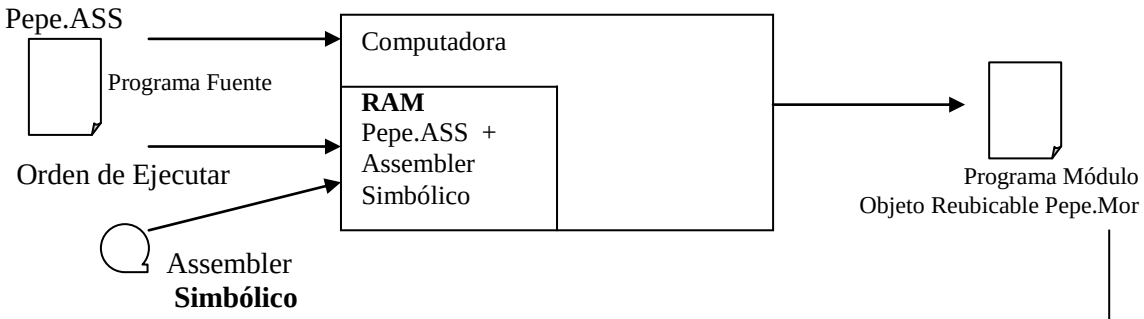
Lenguaje Assembly de direccionamiento simbólico

El lenguaje **Assembly de direccionamiento simbólico** permite que el Programador se libre de la tarea indicada con el número 3(tres) en el listado de las tareas del programador, pues no

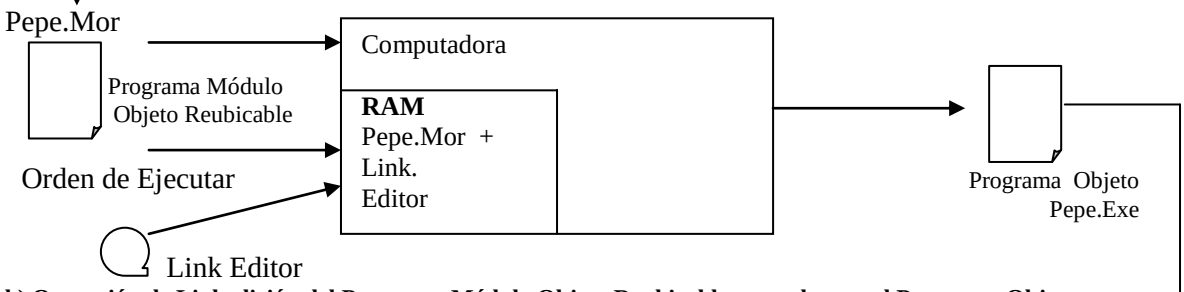
requerirá dar direcciones absolutas en memoria principal a las instrucciones y datos del programa.
Escribir en este lenguaje el Problema Nº 1, daría por resultado:

		REGISTRO	CONTENIDO
	STR A	ACUM	6
	SUB A		
	ADD B		
	ADD C		
	SUB D		
	WRT		
	STP		
D	05		
C	02		
B	07		
A	02		

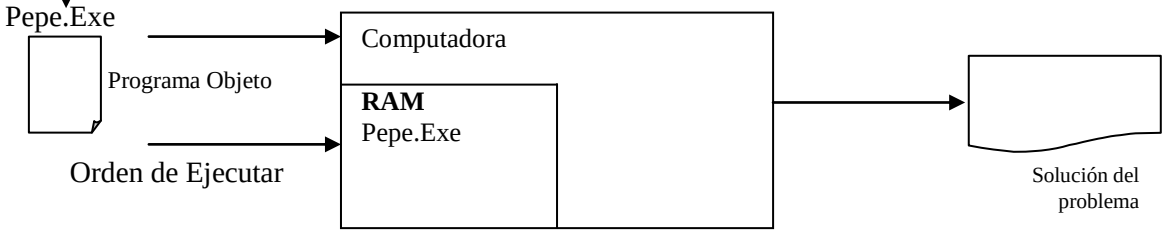
Como puede verse se direcciona a los datos en forma simbólica mediante las letras A, B, C y D. Pero lo que no hace el programador, lo tiene que hacer un programa objeto…llamado Link-Editor que es el encargado de dar direcciones absolutas y encadenar los módulos de un programa. Puede verse en la figura 4 el proceso de trabajo de un programa fuente escrito en Assembly de direccionamiento simbólico.



a)Operación de Compilación del Programa Fuente a Programa Módulo Objeto Reubicable



b) Operación de Linkedición del Programa Módulo Objeto Reubicable para obtener el Programa Objeto



c) Ejecución del Programa Objeto para obtener la solución del Problema

Figura 4: Operación de un programa fuente escrito en Assembly de Direccionamiento Simbólico

El Programa Módulo Objeto Reubicable es un programa escrito en lenguaje de máquina para al cual le faltan las direcciones absolutas de memoria principal de las instrucciones y los datos.

El Link-Editor es el programa encargado de dar esas direcciones absolutas.-

El programador no requiere dar direcciones absolutas a los datos e instrucciones

Lenguaje Macro Assembler V1

El Macro Assembler representa un gran avance en lo que respecta a los lenguajes de computación pues amplía el conjunto de códigos de operación que puede tener un lenguaje.

En los lenguajes anteriores, es el decodificador del código de operación el que contiene las instrucciones del lenguaje en forma microprogramada, es decir, a nivel de las moléculas del microprocesador o más precisamente, a nivel de la Unidad de Control.

El repertorio de instrucciones en Assembly es un conjunto de palabras que constituyen el lenguaje, que tienen una función bien determinada, esas palabras se denominan ***palabras reservadas.***

Algunos autores definen un lenguaje de computación como una colección de palabras reservadas.

Pero hay programas que se usan permanentemente que podrían resumirse en una palabra reservada, por ejemplo,

STR 31
SUB 31

⇒ CLR 31 (esto significa limpiar(clear) el acumulador usando a la memoria 31 como memoria de carga y resta)

Puede tomarse como limpiar
El Acumulador y reconocerse
con un mnemónico

Si tomamos esta instrucción como parte del conjunto de palabras reservada tendremos que se suma a las otras ya definidas y podemos hacer esta tabla llamada librería

Assembly con soporte en Soft	Assembly	Lenguaje de Máquina
CLR x	STR x	011 x
.	SUB x	010 x
STP x	STP x	000 x
ADD x	ADD x	001 x
SUB x	SUB x	010 x
STR x	STR x	011 x
JUP x	JUP x	100 x
JUN x	JUN x	101 x
JUI x	JUI x	110 x
WRT x	WRT x	111 x

Un programa fuente escrito en este Assembly con soporte de soft que se denomina Macro Assembler V1 permite ser escrito de esta manera:

Escribir en este lenguaje el Problema Nº 1, daría por resultado:

		REGISTRO	**CONTENIDO**
		ACUM	6
	CLR A		
	ADD B		
	ADD C		
	SUB D		
	WRT		
	STP		
D	05		
C	02		
B	07		
A	02		

Hemos ahorrado una instrucción respecto del programa escrito en Assembly Simbólico pero esto se puede potenciar aún más, supongamos que incorporamos a nuestro lenguaje la multiplicación:

Columna 1 Assembly con soporte en Soft	Columna 2 Assembly	Columna 3 Lenguaje de Máquina
CLR x	STR x	011 x
.	SUB x	010 x
STP x	STP x	000 x
ADD x	ADD x	001 x
SUB x	SUB x	010 x
STR x	STR x	011 x
JUP x	JUP x	100 x
JUN x	JUN x	101 x
JUI x	JUI x	110 x
WRT x	WRT x	111 x
MUL	STR x	011 x
	SUB x	010 x
	ADD y	010 y
	STR z	011 z
	STR x	011 x
	SUB x	010 x
	ADD a	º
	SUB a	º
	JUN c	º
	STR a	
	STR x	
	SUB x	
	ADD z	etc.
	JUI f	
	STR x	
	SUB x	
	ADD b	
	SUB y	
	STR b	
	WRT h	
	STP h	

Cuando en un programa se escribe MUL se convoca al programa en Assembly que se encuentra en la segunda columna de la librería, esta palabra se usa por Library que significa Biblioteca.

La Biblioteca o Librería de un Lenguaje forma parte del lenguaje y la forma de operar para obtener la solución de un problema es la siguiente:

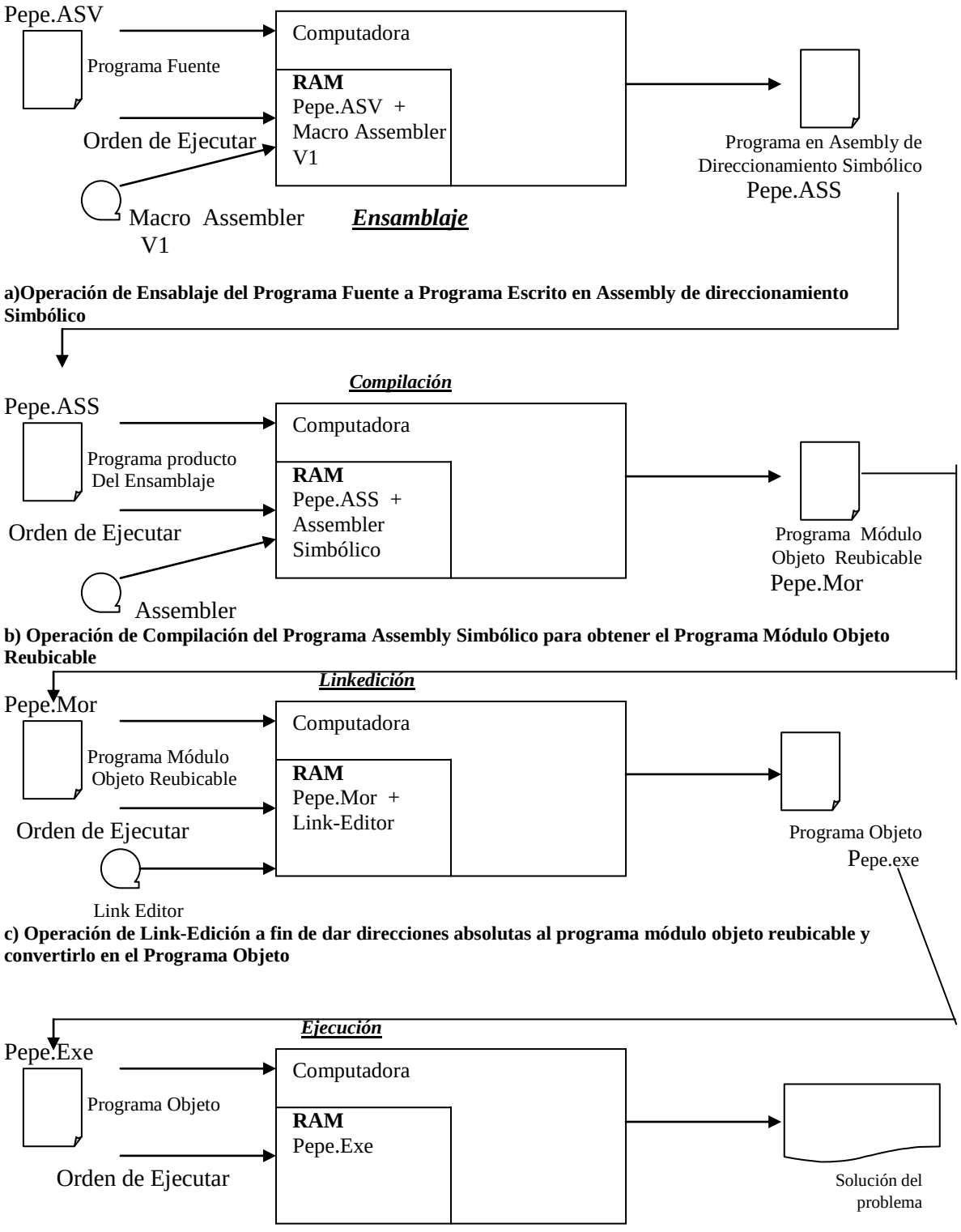


Figura 5: Operación de un programa fuente escrito en MacroAssembler V1

Como puede verse en la figura 5, se cumple el principio de que las acciones que no realiza el programador las tiene que hacer un programa.

De esta manera, al trabajar con el Macro Assembler V1, se tienen que comprar tres programas utilitarios, que son programas objeto, a saber, el ***MacroAssembler V1***, que ensambla es como si transformara en la Librería de la columna 1 a la columna 2 de la Figura 6, el ***Assembler Simbólico***, que compila de la columna 2 a la columna 3 de la Figura 6, y el ***Link-Editor,*** que otorga a cada dirección

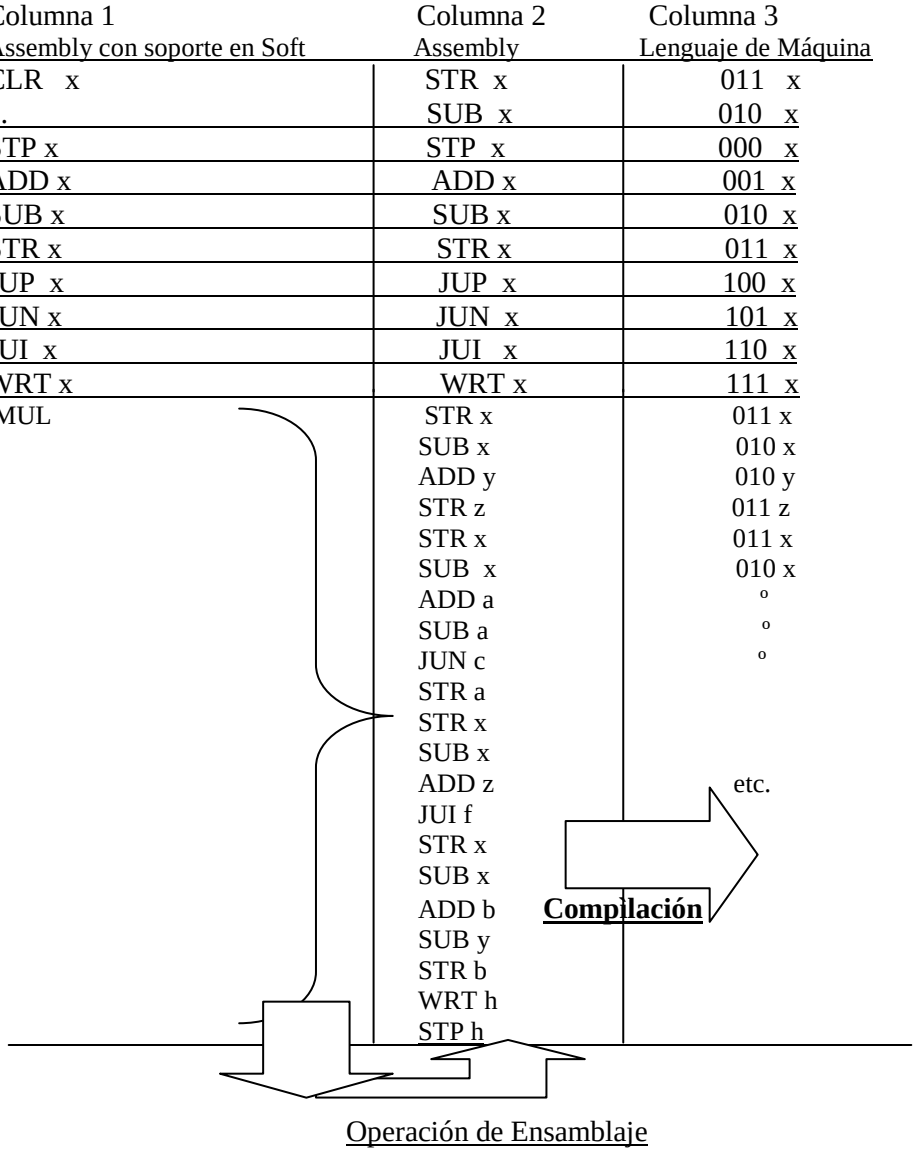


Figura 6: Relación de la Librería del Macro Assembler V1 con las distintas operaciones aque se realizan para obtener el Programa Objeto.

Lenguaje Macro Assembler V2

De la observación de la Figura 6 surge que si la Librería elimina la columna 2 y se vinculan directamente la 1 y la 3, se realizan menos traducciones y qse gana tiempo, precisamente esto hicieron los diseñadores y obtuvieron una librería más compacta y con la cual se logran resultados más rápidos. Los conceptos desarrollados por el MacroAssembler abrieron la puerta para los lenguajes orientados al problema que serán motivo de la Lectura Nº 7. Los vistos hasta ahora se denomina orientados a la máquina.-