LECTURAS SOBRE COMPUTADORAS DIGITALES –LECTURA Nº4
MATERIA: ARQUITECTURA DE LAS COMPUTADORAS

1.- REPERTORIO DE INSTRUCCIONES DE LA PETICOMPUTADORA

Decimal	Binario	Mnemónico	Función que verifica
0	000	STP	La máquina se detiene
1	001	ADD	Suma: (Acum.) + (Mem.) ⟶ (Acum.)
2	010	SUB	Resta: (Acum.) - (Mem.) ⟶ (Acum.)
3	011	STR	Se copia el contenido del acumulador en la posición de memoria que indica la instrucción
4	100	JUP	Salto por positivo
5	101	JUN	Salto por negativo
6	110	JUI	Salto incondicional
7	111	WRT	Se imprime el contenido del Acumulador

TABLA1: Repertorio de instrucciones

El diseñador debe escribir un diccionario acerca de la función que cumple cada una de las instrucciones que se sintetizan en la Tabla 1.

Así como explicar dos elementos que hacen a todo lenguaje: la semántica y la sintaxis. En la Tabla 1 se encuentra la semántica, es decir, el significado de cada una de las palabras que forman parte de este lenguaje y la sintaxis es la forma de escribir cada una de las instrucciones.

Si se observa la Tabla 1 se pueden distinguir una columna que dice binario, esto constituye el LENGUAJE DE MÁQUINA, de la PetiComputadora.

La columna que dice Mnemónico hace referencia a una sigla de dos o tres letras que le indican al programador la función que cumple el conjunto de ceros y unos asociados. La palabra mnemónico significa recordar y hace referencia a las reglas de memorización que se denominan reglas mnemotécnicas.

El conjunto de Mnemónicos define el LENGUAJE ASSEMBLY de esa máquina. Para pasar del Assembly al Lenguaje de Máquina es necesario generar un programa traductor que se denomina Compilador.

En el Lenguaje Assembly, las instrucciones deben escribirse de la siguiente manera:

Mnemónico,espacio en blanco Posición de Memoria

Por ejemplo: ADD, 5 Esta es la sintaxis o forma de escribir
El sentido semántico es: (acum.) + (MEM 5) ⟶ (acum..)

Y se lee “Suma del contenido del Registro Acumulador y del contenido de Memoria 5 y el resultado queda en el Registro Acumulador).

Las instrucciones se clasifican según las funciones que cumplen, de esta manera se tienen las Instrucciones Matemáticas, que en este caso incluyen la suma(ADD) y la resta(SUB). Los modos de escribir las instrucciones matemáticas son similares.

La resta SUB, se escribe SUB, 5
Y esto significa: (acum.) – (MEM 5) = (acum.)
Si antes de ejecutar la operación los contenidos son: (acum.) = 3 , (MEM 5) = 2
Después de ejecutar la operación los contenidos son: (acum.) = 1 , (MEM 5) = 2

En el repertorio de instrucciones existen **Instrucciones de Control**: STP Y WRT, la sintaxis de las mismas es: Mnemónico, espacio en blanco no importa lo que se coloque

STP, nn En la ejecución la máquina se detiene

WRT, nn En la ejecución siempre se imprime el contenido del acumulador.

En los repertorios de todos los microprocesadores se incluyen **Instrucciones de carga(load) y almacenamiento(store),** en el repertorio de PetiComputadora no hay instrucciones de carga pero si de almacenamiento.

Las instrucciones de almacenamiento tienen por objetivo "copiar" el contenido del Registro Acumulador en una posición de memoria. El mnemónico es **STR** y su sintaxis es:

Mnemónico, espacio en blanco dirección de memoria en donde se copia el contenido del acumulador

STR, 5
(acum.) se copia en (MEM 5)
(acum.) ⟶ (MEM5)

Si antes de la ejecución de la instrucción los contenidos son: (acum.) = 7 y (MEM 5) = 2
Después de la ejecución de la instrucción los contenidos son: (acum.) = 7 y (MEM 5) = 7

Ahora estamos en condiciones de hacer un problema elemental, por ejemplo, restar dos números que se encuentran en memoria e imprimir el resultado.

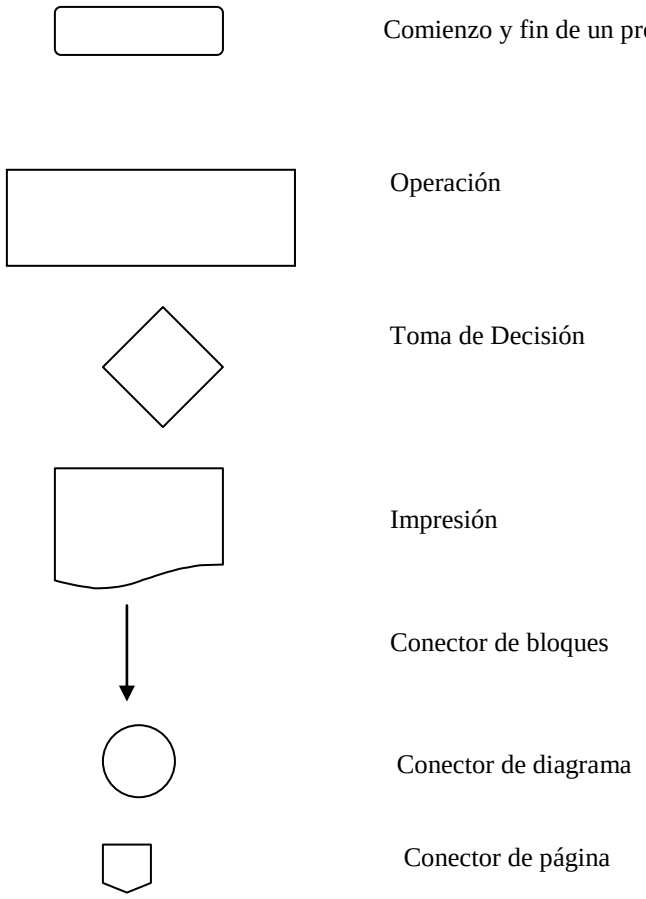
Para poder resolver este problema debemos pensar cómo la máquina lo resuelve, primeramente será necesario limpiar el acumulador, luego colocar o sumar el minuendo, luego restar el sustraendo y luego imprimir:

MEM	Instrucción/dato	Interpretación……
0	STR, 08	Se almacena el (acum.) en la (MEM 08)
1	SUB, 08	Se le resta al (acum.) el (MEM 08)
2	ADD, 06	Se suma al (acum.) el (MEM 06)
3	SUB, 07	Se resta al (acum.) el (MEM 07)
4	WRT, 07	Se imprime el (acum.) que es 03 nótese que el 07 no se toma en cuenta para nada)
5	STP, 05	La máquina se detiene (nótese que el 05 no cuenta)
6	07	Minuendo(dato)
7	04	Sustraendo(dato)
8	Reservar	Posición de memoria usada para limpiar (acum.)

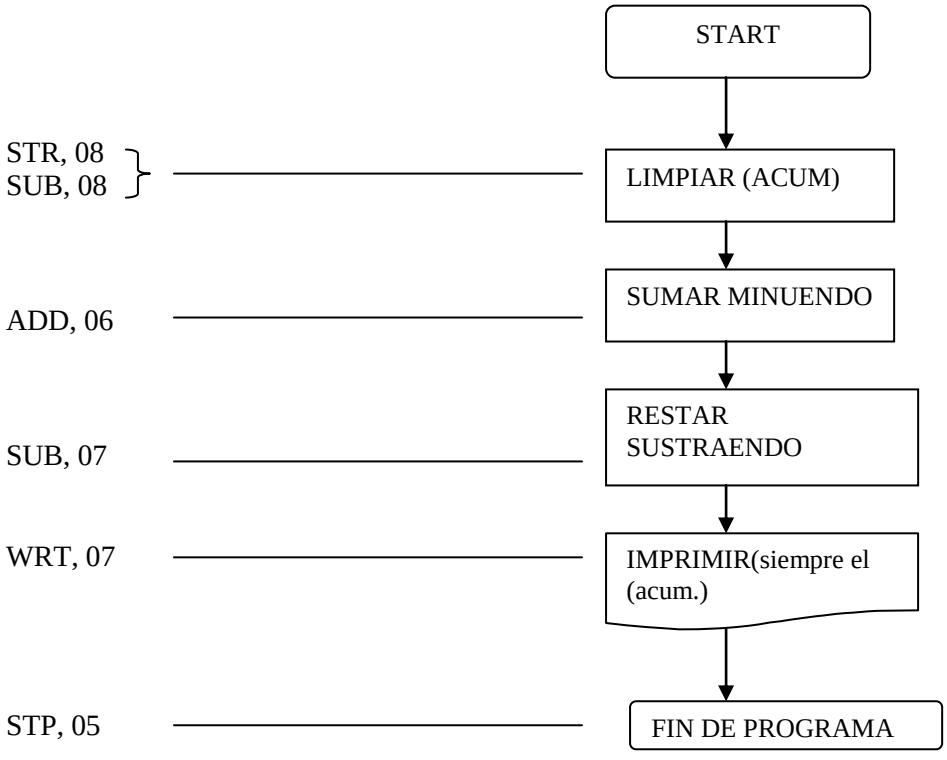
Otra manera de pensar un programa es a través de una manera convencional de representar cada operación que se denomina diagrama de flujo.

El diagrama de flujo es el proceso cognitivo que realiza la máquina para resolver un problema, expresado en símbolos convencionales

SIMBOLOS DE LOS DIAGRAMA DE FLUJO



Por ejemplo, el programa que se hizo anteriormente da origen al siguiente diagrama de flujo :



Nótese que en el diagrama existe una relación biunívoca y recíproca entre una o más de las instrucciones escritas en Assembly y los símbolos del diagrama de flujo.

En el mismo no figuran los datos ni las posiciones de máquina a reservar, ni las posiciones de memoria en las cuales hay que colocar las instrucciones.

Continuando con el análisis del repertorio de instrucciones se tienen las instrucciones de salto, en esta máquina son tres:

JUP que se define como salto por positivo,
JUN que se define como salto por negativo y,
JUI que se define como salto incondicional…

Las dos primeras son "saltos condicionales" y la tercera es un "salto sí o sí".

Primeramente; ¿por qué las instrucciones de salto? y, en segundo término: ¿de dónde se obtiene la condición?.

Las condiciones de salto aparecen en el momento que hay que tomar una decisión y existen opciones. Se debe pensar el programa como un camino del pensamiento cognitivo a fin de llegar a la solución de un problema. Si los programas se colocaran en una memoria que fuera un plano, ante la pregunta: ¿El resultado es positivo? Y las aternativas: "Si es positivo" sumar 1, imprimir y parar. Si "No es positivo" sumar 2, imprimir y parar. El programa podría hacerse como puede verse en la figura 1.

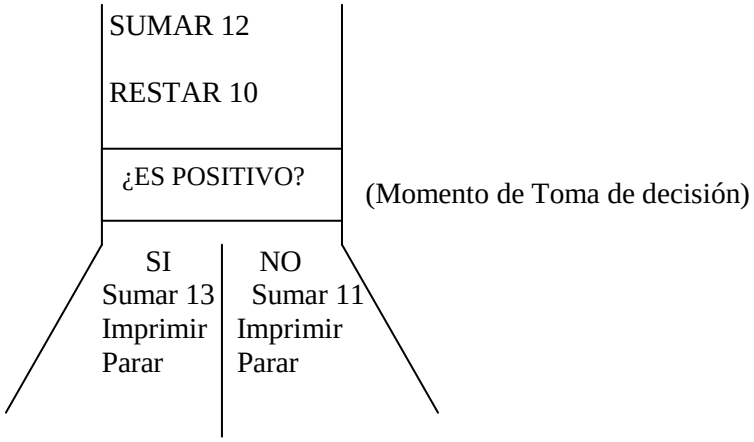


Figura 1: En un plano, cuando deben tomarse decisiones, se puede tomar por uno u otro camino

Pero si las instrucciones deben colocarse en una pila de registros, tal como es la memoria, en el momento de tener una respuesta SI a la pregunta, habrá que saltar las instrucciones del programa que están en el "camino del SI" y que hacen al programa del NO.

Como se muestra en el programa escrito a continuación que no está en Assembly sino en lenguaje humano común, para una mejor comprensión.

Memoria	Instrucción	Comentarios
02	SUMAR 12	Comienza la secuencia de la figura 1
03	RESTAR 10	Continúa la secuencia
04	SALTO POR POSITIVO A 8	Momento de toma de decisión, supongamos que la la respuesta es NO hay que saltar a 8 y desechar lo que indican las Memorias 5,6 y 7. Romper la secuencia y continuar en 8.
05	SUMAR 13	En memoria 13 se encuentra un 2
06	IMPRIMIR	En esta máquina se imprime el (acum.)
07	PARAR	Fin del programa
08	SUMAR 11	En la memoria 11 se encuentra un 1
09	SALTO SÍ O SÍ A 06	El programa regresa a la memoria 6 para imprimir
10	04	Sustraendo
11	01	Dato
12	07	Minuendo
13	02	Dato

Este programa, escrito en el Assembly de la Tabla 1 quedará expresada como:

Memoria	Instrucción	Comentarios
02	ADD, 12	Comienza la secuencia de la figura 1
03	SUB, 10	Continúa la secuencia
04	JUP, 08	Momento de toma de decisión, supongamos que la la respuesta es NO hay que saltar a 8 y desechar lo que indican las Memorias 5,6 y 7. Romper la secuencia y continuar en 8.
05	ADD 13	En memoria 13 se encuentra un 2
06	WRT, 05	En esta máquina se imprime el (acum.)
07	STP, 09	Fin del programa
08	ADD, 11	En la memoria 11 se encuentra un 1
09	JUI, 06	El programa regresa a la memoria 6 para imprimir
10	04	Sustraendo
11	01	Dato
12	07	Minuendo
13	02	Dato

Este programa puede analizarse paso a paso y con el supuesto, que el acumulador ya estaba en cero. Haremos el análisis siguiendo los pasos del ciclo de la instrucción.

MEM 02

1.- (mem 02) ⟶ (IR) El contenido de la memoria 2 se copia en el Registro de Instrucción
ADD, 12 = (IR)

2.- (PC) + 1 ⟶ (PC) = 03 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 03.

3.- (acum.) + (mem 12) ⟶ (buffer) Etapa de Interpretación de la Instrucción
0 + 07 ⟶ (buffer) = 07 Se hace la suma indicada

BÚSQUEDA

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 04 El ciclo de la instrucción concluye.

MEM 03

1.- (mem 03) ⟶ (IR) El contenido de la memoria 3 se copia en el Registro de Instrucción
SUB, 10 = (IR)

2.- (PC) + 1 ⟶ (PC) = 04 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 04.

3.- (acum.) - (mem 10) ⟶ (buffer) Etapa de Interpretación de la Instrucción
07 - 04 ⟶ (buffer) = 03 Se hace la resta indicada

BÚSQUEDA

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 04 El ciclo de la instrucción concluye.

MEM 04

1.- (mem 04) ⟶ (IR) El contenido de la memoria 4 se copia en el Registro de Instrucción
JUP, 08 = (IR)

2.- (PC) + 1 ⟶ (PC) = 05 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 05.

3.- La máquina se pregunta si el último resultado es positivo y se contesta : **SI**, luego el salto tiene que ocurrir.

BÚSQUEDA

4.- (etapa de ejecución) El (PC) = 08 (el salto ocurrió) El ciclo de la instrucción concluye

Nótese que al ejecutarse la instrucción de salto se rompió la secuencia natural del programa

MEM 08

1.- (mem 08) ⟶ (IR) El contenido de la memoria 8 se copia en el Registro de Instrucción
ADD, 11 = (IR)

2.- (PC) + 1 ⟶ (PC) = 09 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 09.

3.- (acum.) +(mem 11) ⟶ (buffer) Etapa de Interpretación de la Instrucción
03 + 01 ⟶ (buffer) = 04 Se hace la suma indicada

BÚSQUEDA

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 04 El ciclo de la instrucción concluye.

MEM 09

1.- (mem 09) ⟶ (IR) El contenido de la memoria 9 se copia en el Registro de Instrucción
JUI, 06 = (IR)

2.- (PC) + 1 ⟶ (PC) = 10 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 10.

3.- La máquina visualiza que el salto debe ocurrir Si ó SI, luego el salto tiene que ocurrir.

BÚSQUEDA

4.- (etapa de ejecución) El (PC) = 06 (el salto ocurrió) El ciclo de la instrucción concluye

Nótese que al ejecutarse la instrucción de salto se rompió la secuencia natural del programa

MEM 06

1.- (mem 06) ⟶ (IR) El contenido de la memoria 6 se copia en el Registro de Instrucción
WRT, 5 = (IR)

2.- (PC) + 1 ⟶ (PC) = 07 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 07.

3.- La máquina VISUALIZA QUE DEBE IMPRIMIR EL (ACUM)
El 5 que completa a la instrucción no se toma en cuenta

BÚSQUEDA

4.- (etapa de ejecución) Se imprime el 4 que es el (acum) El ciclo de la instrucción concluye

MEM 07

1.- (mem 07) ⟶ (IR) El contenido de la memoria 7 se copia en el Registro de Instrucción

STP, 9 = (IR)

2.- (PC) + 1 ⟶ (PC) = 08 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 08.

3.- La máquina VISUALIZA QUE DEBE DETENERSE

El 9 que completa a la instrucción no se toma en cuenta

BÚSQUEDA

4.- (etapa de ejecución) LA MÁQUINA SE DETIENE El ciclo de la instrucción concluye

FINALMENTE

Se listan los registros y posiciones de memoria que han cambiado su contenido
(PC) = 08
(acum) = 04
El resto de las posiciones de memoria permanece sin cambio

2.- ESCRIBIENDO UN PROGRAMA

Cuando se tiene que resolver un problema mediante un programa, lo primero que hay que hacer es determinar el problema, y cómo se quiere la solución.

A continuación, pensar cómo la máquina resuelve ese problema paso a paso, haciendo el diagrama de flujo respectivo.

Hacer una prueba de escritorio siguiendo el diagrama de flujo.

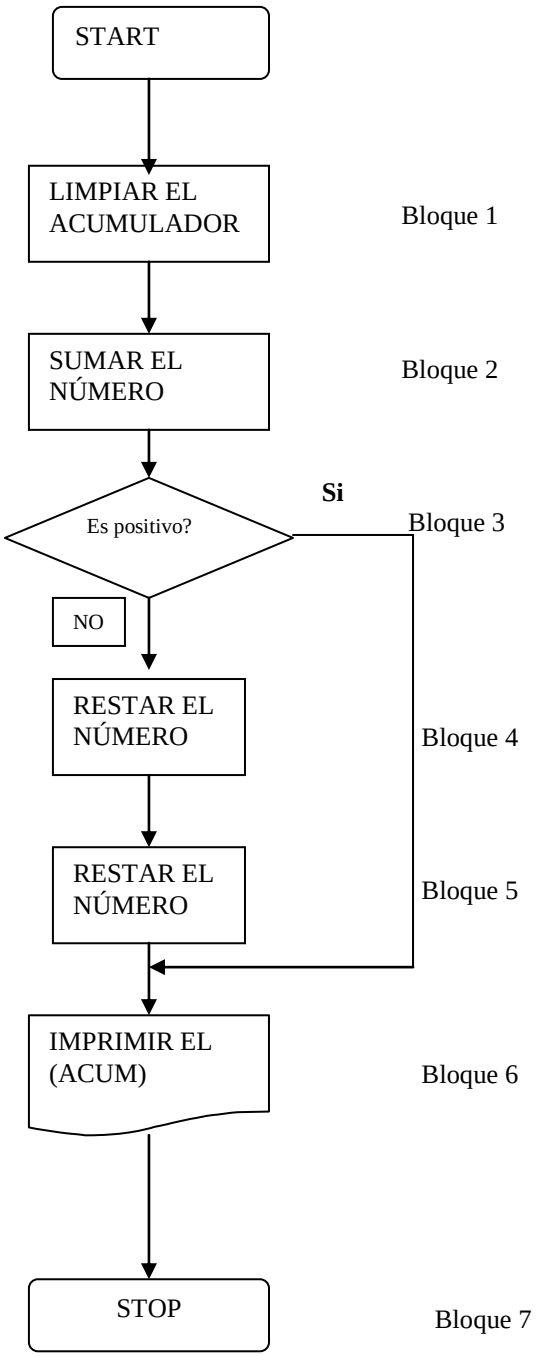
Luego pasar el diagrama de flujo a instrucciones en Assembly.

EJEMPLO:

- **Definición del Problema**: Escribir un programa en el lenguaje Assembly de la PetiComputadora que imprima el valor absoluto de un número que se encuentra ubicado en la memoria 30 de la Memoria Principal.
- **Pensando como la PC**:

0	Para poder trabajar con un número que se encuentra en memoria principal primeramente se lo debe ubicar en el registro acumulador.
1	Pero el acumulador puede tener algún dato previo, en consecuencia, lo primero que hay que hacer es limpiar el acumulador.
2	Luego sumarle el contenido de la memoria 30 y se tiene el número cuyo valor absoluto se quiere obtener ubicado en el acumulador.
3	Si el número es negativo habrá que cambiarle el signo, esto se puede hacer restando dos veces el mismo número: si fuera -7, se hace (-7) – (-7) – (-7) = 7 que es el valor absoluto. Y luego imprimir y parar.
4	Si el número es positivo basta con imprimirlo y parar.

- **Trazando el diagrama de flujo:**



- **PRUEBA DE ESCRITORIO**

La misma consiste en seguir paso a paso el diagrama de flujo y ver si realmente se cumple lo señalado en la definición del problema.
Supongamos tener el número 3 en la posición de memoria 30.

- Mediante el Bloque 1 se coloca el (ACUM) = 0
- Mediante el Bloque 2 se coloca el (ACUM) = 3
- Mediante el Bloque 3 la máquina se pregunta si el número es positivo, debido a que contesta que SI, pasa al….

- Bloque 6, en el cual se produce la Impresión del (ACUM) = 3
- Mediante el bloque 7, la máquina se detiene.

Conclusión: el valor absoluto de 3 es precisamente 3.

Supongamos tener el número -5 en la posición de memoria 30.

- Mediante el Bloque 1 se coloca el (ACUM) = 0
- Mediante el Bloque 2 se coloca el (ACUM) = - 5
- Mediante el Bloque 3 la máquina se pregunta si el número es positivo, debido a que contesta que NO, se continúa con el….
- Bloque 4, en el cual se produce la resta de (-5) – (-5), quedando el (ACUM) = 0
- Bloque 5, se produce la resta (0) – (-5), quedando el (ACUM) = 5
- Bloque 5, se imprime el contenido del acumulador, es decir, 5
- Mediante el bloque 7, la máquina se detiene.

Conclusión: el valor absoluto de -5 es 5.

- **SE ESCRIBE EL PROGRAMA EN LENGUAJE ASSEMBLY**

MEMORIA	**CONTENIDO**
0	STR, 31
1	SUB, 31
2	ADD, 30
3	JUP, 6
4	SUB, 30
5	SUB, 30
6	WRT, 9
7	STP, 5
.	.
.	.
.	.
30	DATO
31	RESERVAR

REGISTROS	**CONTENIDO**
Acum	20
PC	0

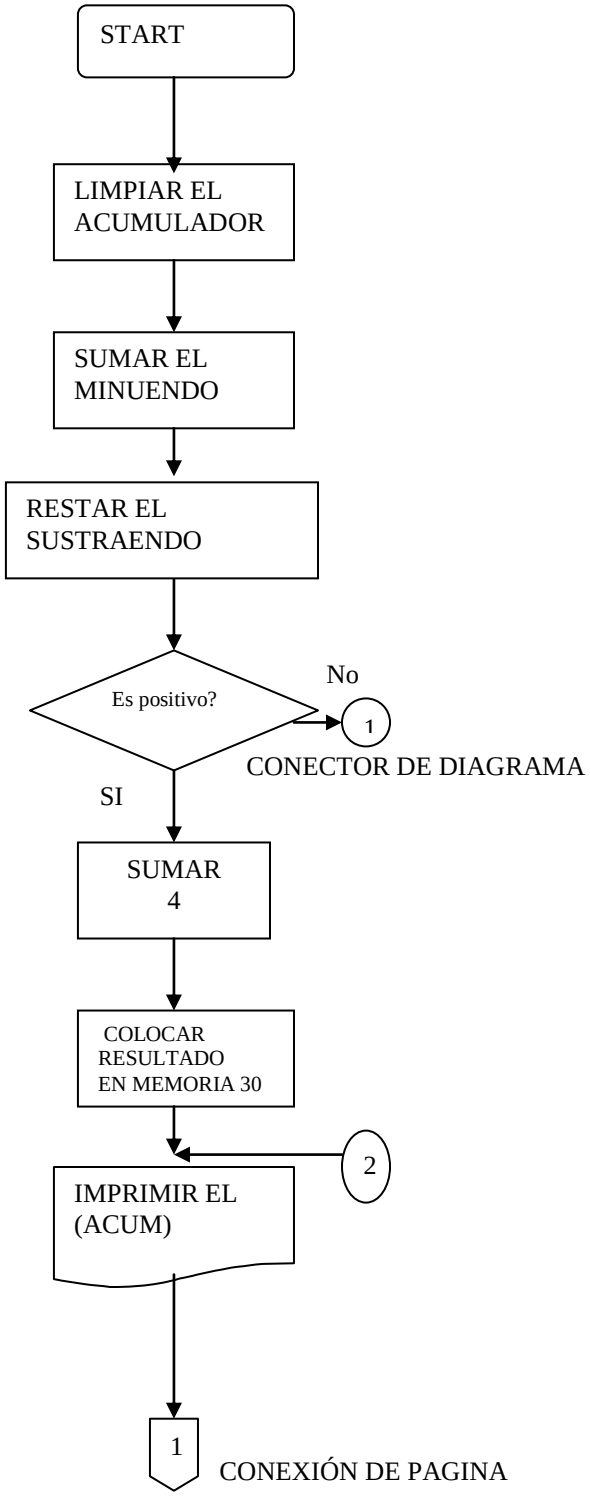
SEGUNDO EJEMPLO(se propone que este problema lo resuelvan los alumnos)

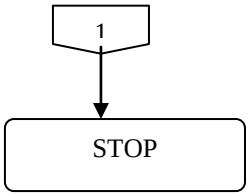
- **Definición del Problema**: Escribir un programa en el lenguaje Assembly de la PetiComputadora que reste dos números que se encuentran en memoria, si el resultado es positivo sumar 4, colocar el resultado en memoria 30,imprimir y parar, si el resultado es negativo, sumarle 2, colocar el resultado en memoria 29, imprimir y parar..
- **Pensando como la PC**:

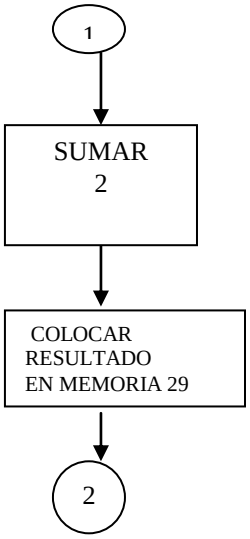
0	Para poder trabajar con números que se encuentra en memoria principal primeramente se lo debe ubicar en el registro acumulador.
1	Pero el acumulador puede tener algún dato previo, en consecuencia, lo primero que hay que hacer es limpiar el acumulador.
2	Luego sumarle al acumulador, el contenido del minuendo..
3	Luego restar el sustraendo.
4	Si el número es positivo sumarle 4 y seguir en (5), sino pasar al paso 8.
5	Colocar el resultado en memoria 30

6	Imprimir
7	Parar
8	Debido a que el número obtenido de la resta es negativo, sumarle 2.
9	Pasar el resultado a la memoria 29
10	Imprimir
11	Parar.

- **Trazando el diagrama de flujo:**







No se incluye la prueba de escritorio.

- **SE ESCRIBE EL PROGRAMA EN LENGUAJE ASSEMBLY**

MEMORIA	CONTENIDO
0	STR, 31
1	SUB, 31
2	ADD, 15
3	SUB 12
4	JUN, 09
5	ADD, 13
6	STR, 30
7	WRT, 3
8	STP, 9
9	ADD, 14
10	STR, 29
11	JUI, 07
12	SUSTRENDO
13	4
14	2
15	MINUENDO
.	.
.	.
.	.
31	RESERVAR

REGISTROS	CONTENIDO
Acum	4
PC	0

- **EJEMPLO DE ANÁLISIS DE UN PROGRAMA**

Frente a un programa ya escrito y con el propósito de optimizarlo conviene hacer un análisis paso a paso.

Para ejemplificar, analicemos el siguiente programa: Dado el siguiente programa, analizar instrucción por instrucción los cambios que se producen en cada registro y posiciones de memoria.

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

MEM 0

1.- (mem 0) ⟶ (IR) El contenido de la memoria 2 se copia en el Registro de Instrucción
STR 10 = (IR)

2.- (PC) + 1 ⟶ (PC) = 01 El contador de programa se incrementa en uno e indica que la próxima instrucción del programa se encuentra en memoria 01.

3.- Interpretación: (acum.) ⟶ (10)
4 ⟶ (10)
(10) = 2

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 04 El ciclo de la instrucción concluye.

MEM 1

1.- (mem 01) ⟶ (IR) El contenido de la memoria 1 se copia en el Registro de Instrucción
SUB, 10 = (IR)

2.- (PC) + 1 ⟶ (PC) = 02 El contador de programa se incrementa en uno.

.

3.- (acum.) - (mem 10) ⟶ (buffer) Etapa de Interpretación de la Instrucción
02 - 02 ⟶ (buffer) = 0 Se hace la resta indicada

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 0 El ciclo de la instrucción concluye.

MEM 02

1.- (mem 02) ⟶ (IR) El contenido de la memoria 2 se copia en el Registro de Instrucción
ADD,09 = (IR)

2.- (PC) + 1 ⟶ (PC) = 03 El contador de programa se incrementa en uno.

3.- (acum.) + (9) = (buffer) Etapa de Interpretación
0 + 7 = (buffer), (buffer) = 7

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 7 El ciclo de la instrucción concluye.

MEM 03

1.- (mem 03) ⟶ (IR) El contenido de la memoria 3 se copia en el Registro de Instrucción
ADD, 8 = (IR)

2.- (PC) + 1 ⟶ (PC) = 04 El contador de programa se incrementa en uno.

3.- (acum.) +(mem 8) ⟶ (buffer) Etapa de Interpretación de la Instrucción
07 + 02 ⟶ (buffer) = 09 Se hace la suma indicada

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 09 El ciclo de la instrucción concluye.

MEM 04

1.- (mem 04) ⟶ (IR) El contenido de la memoria 4 se copia en el Registro de Instrucción
SUB, 7 = (IR)
2.- (PC) + 1 ⟶ (PC) = 05 El contador de programa se incrementa en uno.
3.- (acum.) - (mem 7) ⟶ (buffer) Etapa de Interpretación de la Instrucción
09 - 05 ⟶ (buffer) = 04 Se hace la suma indicada

4.- (etapa de ejecución) (buffer) ⟶ (acum.) = 04 El ciclo de la instrucción concluye.

MEM 05
1.- (mem 05) ⟶ (IR) El contenido de la memoria 6 se copia en el Registro de Instrucción
WRT, 0 = (IR)
2.- (PC) + 1 ⟶ (PC) = 06 El contador de programa se incrementa en uno.
3.- La máquina VISUALIZA QUE DEBE IMPRIMIR EL (ACUM)
El 0 que completa a la instrucción no se toma en cuenta
4.- (etapa de ejecución) Se imprime el 4 que es el (acum) El ciclo de la instrucción concluye
MEM 07
1.- (mem 07) ⟶ (IR) El contenido de la memoria 7 se copia en el Registro de Instrucción
STP, 5 = (IR)
2.- (PC) + 1 ⟶ (PC) = 08 El contador de programa se incrementa en uno .
3.- La máquina VISUALIZA QUE DEBE DETENERSE
El 5 que completa a la instrucción no se toma en cuenta

4.- (etapa de ejecución) LA MÁQUINA SE DETIENE El ciclo de la instrucción concluye

FINALMENTE

Se listan los registros y posiciones de memoria que han cambiado su contenido
(PC) = 08
(acum) = 04
(mem 10) = 6

- **EJERCITACIÓN PARA LOS ALUMNOS**

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

Cada Instrucción tiene tres bits para el código de operación y cinco bits para el direccionamiento del segundo operando.

Problema Nº 2

Analizar instrucción por instrucción los cambios que se producen en cada registro y posición de memoria.

MEMORIA	CONTENIDO
0	STR 31
1	SUB 31
2	ADD 30
3	SUB 29
4	JUP 8
5	JUN 11
6	WRT 3
7	STP 6
8	ADD 28
9	WRT 2
10	STP 5
11	ADD 27
12	WRT 2
13	STP 4
.	.
.	.
.	.
27	07
28	02
29	03
30	05
31	02

REGISTRO	CONTENIDO
ACUM	6
PC	0

Solución

(ACUM) = 04
(PC) = 11(en decimal)
(MEM 31) = 04

TAREAS ADICIONALES: Analizar el programa para ver si se pueden eliminar instrucciones.
Escribirlo en Lenguaje de Máquina

Problema Nº 3

Analizar instrucción por instrucción los cambios que se producen en cada registro y posición de memoria.

MEMORIA	CONTENIDO
0	STR 31
1	SUB 31
2	ADD 16
3	JUN 11
4	ADD 30
5	STR 30
6	SUB 30
7	ADD 02
8	ADD 29
9	WRT 02
10	JUI 00
11	STR 31
12	SUB 31
13	ADD 30
14	WRT 09
15	STP 06
16	2
17	7
18	-1
.	.
.	.
.	.
29	01
30	00
31	02

REGISTRO	CONTENIDO
ACUM	6
PC	0

TAREAS ADICIONALES: Analizar el programa para ver si se pueden eliminar instrucciones.
Escribirlo en Lenguaje de Máquina

COMENTARIO: Este problema introduce los siguientes conceptos

- De suma de valores pertenecientes a una tabla.
- De Bandera (-1)
- Incógnita adicional: ¿Cuántos valores puede tener la tabla en esta máquina? (Recordar que tiene 32 posiciones de memoria.)

Problema Nº 4

Escribir en el Assembly de la PetiComputadora un programa que tome un número que se halla en la memoria 30, si es positivo imprimir el doble y parar. Si es negativo imprimir cero.

Problema Nº 5

Escribir en el Assembly de la PetiComputadora un programa que realice la multiplicación de dos números positivos, distintos de cero y que el resultado no exceda de 127.

AYUDA: Pensar que m x n = m + m + ...+m
{ n veces }