Práctica 1

Elementos en la Organización de una Computadora de Propósito General

---

Luis Eduardo Galindo Amaya
1274895

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	2022-08-14

1. Intrucciones

El alumno se familiarizará con la herramienta Marie.js para la ejecución de código en lenguaje ensamblador.

1. Identificar las caracteristicas de la herramienta marie.js
    - Ingrese a la página donde se encuentra la herramienta.
    - Consulte la documentación y tutoriales de Marie, en especial las secciones Introduction to Marie y Marie Codes.
2. Realizar los programas
    - solicite al usuario 2 números y despliegue el resultado de la ecuación $2x + 3y - 5$.
    - solicite 2 números y los reste. Desplegar un 1 si el resultado fue negativo o un 0 en caso contrario.
3. Realizar un reporte que incluya:
    - Diagrama de bloques de una máquina de von Neumann y una breve descripción de cada componente.
    - Lista de las instrucciones de Marie y su y su función.
    - Describir el funcionamiento de los registros del Acumulador, Registro de instrucción, Contador del Programa, Registro de Acceso a Memoria (MAR), y Registro de Buffer de Memoria (MBR).

2. Máquina De Von Neumann

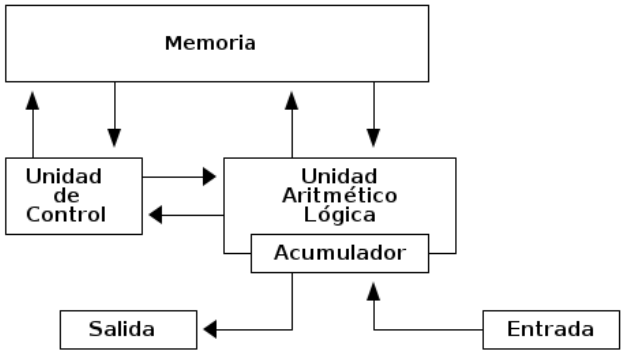
Memoria Son un conjunto de celdas usadas para cualquier proceso, se utiliza para almacenar los programas que va a ejecutar el procesador e información que el programa necesite almacenar.

Unidad Aritmético Lógica es un circuito digital electrónico que realiza las operaciones aritméticas y lógicas bit a bit en números binarios enteros.

Unidad de Control es un componente que dirige las operaciones del procesador. Le dice a la memoria, ALU y los dispositivos de entrada y salida cómo responder a las instrucciones de un programa.

Entrada/Salida Es la comunicación entre la computadora y un humano u otro sistema de procesamiento de información. Inputs son las señales o datos recibidos por el sistema y outputs son las señales o datos enviados por éste.

Registros son unidades de almacenamiento pequeñas que son típicamente dirigidas por mecanismos distintos de la memoria principal y a los que se puede acceder más rápido.



3. Registros

Program counter (PC) Registro contador del programa, contiene la dirección de la instrucción siguiente que hay que leer de la memoria.

Instruction register (IR) El registro de instrucción, contiene la instrucción que hay que ejecutar.

Memory address register (MAR) Registro de direcciones de memoria, donde ponemos la dirección de memoria a la que queremos acceder.

Memory buffer register (MBR) Registro de datos de memoria; registro donde la memoria deposita el dato leído o el dato que queremos escribir.

Acumulador (AC) el acumulador es un registro en el que son almacenados temporalmente los resultados aritméticos y lógicos intermedios que serán tratados por el circuito operacional de la unidad aritmético-lógica.

4. Instrucciones de Marie.js

Dec	Hex	Instrucción	Resumen
0	0	JnS X	Jumps and Store: Stores value of PC at address X then increments PC to X+1
1	1	Load X	Loads Contents of Address X into AC
2	2	Store X	Stores Contents of AC into Address X
3	3	Add X	Adds value in AC at address X into AC
4	4	Subt X	Subtracts value in AC at address X into AC
5	5	Input	Request user to input a value
6	6	Output	Prints value from AC
7	7	Halt	End the program
8	8	Skipcond (C)	Skips the next instruction based on C: if(C)= + 000: Skips if AC <0 + 400: Skips if AC = 0 + 800: Skips if AC >0
9	9	Jump X	Jumps to Address X
10	A	Clear	AC $\leftarrow$ 0
11	B	AddI X	Add Indirect: Use the value at X as the actual address of the data operand to add to AC
12	C	JumpI X	Uses the value at X as the address to jump to
13	D	LoadI	Loads value from indirect address into AC e.g. LoadI addresspointer Gets address value from addresspointer, loads value at the address into AC
14	E	StoreI	Stores value in AC at the indirect address. e.g. StoreI addresspointer Gets value from addresspointer, stores the AC value into the address

5. Programas

Programa-1.mas

```
1  / Solicite al usuario 2 números y despliegue el resultado de la
2  / ecuación 2x+3y-5.
3
4  INPUT                          / Captura el valor de X
5  Store X
6  INPUT                          / Captura un valor de Y
7  Store Y
8
9  Clear                          / limpiar Acumulador, hacer AC ← 0
10
11 Add X                          / 2x
12 Add X
13 Add Y                          / 3y
14 Add Y
15 Add Y
16 Subt R                         / -5
17
18 Output
19 Halt
20
21 X, DEC 0
22 Y, DEC 0
23 R, DEC 5
```

Programa-2.mas

```
1  / Escriba un código que solicite 2 números y los reste.  Desplegar
2  / un 1 si el resultado fue negativo o un 0 en caso contrario.
3
4  INPUT                          / Captura el valor de X
5  Store X
6  INPUT                          / Captura un valor de Y
7  Store Y
8
9  load X                         / Carga el valor de X en el acumulador
10 Subt Y                         / Resta el valor de Y a X
11
12 / Para ese punto, el valor en AC es 'X-Y'
13
14 Skipcond 000                   / si AC es mayor o igual a 0
15 clear                          / el valor en AC se vuelve 0
16
17 Skipcond 400                   / si es diferente a 0
18 Load verdadero                 / el valor en AC se vuelve 1
19
20 Output
21 Halt
22
23 X, DEC 0
24 Y, DEC 0
25 verdadero, dec 1
```

6. Capturas De Pantallas

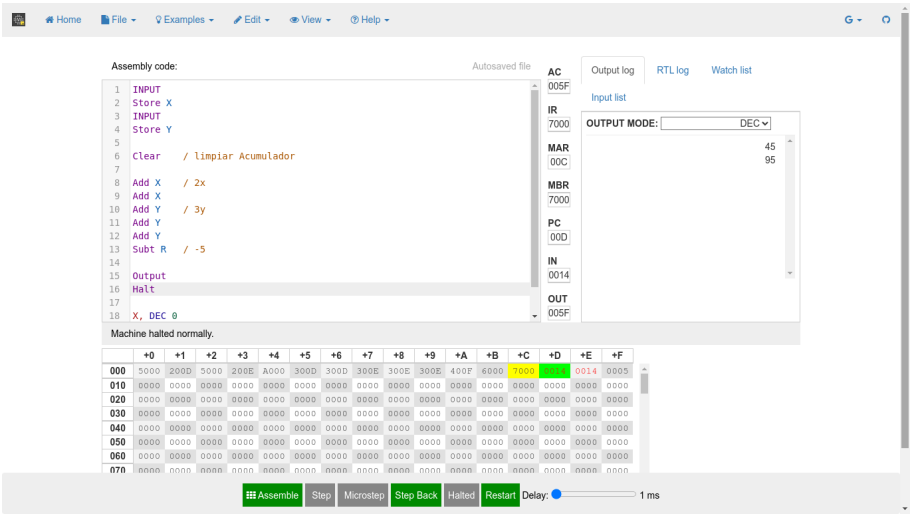


Figura 1: Ejecución del programa 1

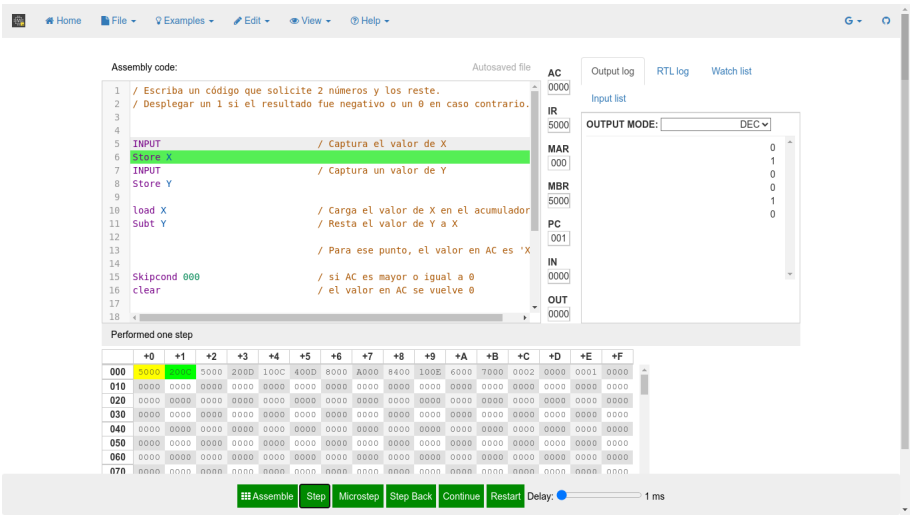


Figura 2: Ejecución del programa 2

7. Conclusiones y Comentarios

Al programa en ensamblador es necesario estar mas al pendiente de los estados de la memoria al mismo tiempo que las ejecuciones que realizamos, en otras palabras el estado previo de la memoria no puede ser ignorado o extraído para ser procesado. **Complicaciones**: En el segundo programa fue necesario que revisara en que orden se tienen que evaluar los operadores lógicos, ya que si evaluaba primero el '=' no se podía resolver.