**Microoperaciones**

Una microoperacion es una operación aplicada a un registro, activada en un instante de tiempo determinado sincronizado por los pulsos del reloj. Al entender una instrucción se genera una microoperacion y la encargada de realizar esto es la unidad de control.

**Instrucciones**

***Codigo de Instrucción***: combinación de bits que interpreta la CU de la CPU para generar una microoperacion que permita la ejecución de la instrucción.

***Formato de Instrucción:*** forma de agrupar el código de instrucción en diferentes formas.

Cabe destacar que una misma Unidad de Control puede interpretar distintos formatos de instrucciones. Cuando esta en la misma, se encuentra en estado de ejecución.

El **COP** determina la cantidad de acciones distintas que se pueden realizar (2\*..\*n bits) . El **DATO/A** determina la dirección de la posición de memoria donde se aloja el dato.

**Monoprogramación:** ejecución de un solo programa por vez.

**Función de la CPU**

* ***Tratamiento de instrucciones:*** realizado por la unidad de control, sincronizada por pulsos de reloj.
* ***Operación de datos:*** realizado por la unidad aritmético lógica (ALU).

La unidad de control toma las instrucciones del programa que se encuentra almacenado en memoria para interpretarlas y ejecutarlas. Podemos dividir este proceso en 4 pasos:

1. Buscar la instrucción en memoria.
2. Interpretar el COP.
3. Buscar el dato que afecta.
4. Generar ordenes sobre el modulo que opera a ese dato.

La ejecución de instrucciones se divide en 2 pasos o etapas:

* ***Fase Fetch o Fase de Búsqueda***
* ***Fase Execute o Fase de Ejecución***

Cada vez que se completan ambas fases se completa un ciclo, denominado **ciclo de instrucción**.

Proceso (Fase de Búsqueda)

La dirección de la instrucción es retenida por la unidad de control en el IP (Instruction Pointer). Para pasar a la próxima instrucción, se incrementa en 1 el valor del IP ya que las instrucciones se almacenan en palabras sucesivas. Una vez que se envía a memoria el contenido del IP se lee la palabra y la misma se almacena en la MDR (Memory Data Register) y la instrucción leída es almacenada en el IR (Instruction Register) hasta el momento en que termine su ejecución.

El DATO de la instrucción se relaciona con la Memory Adress Register (MAR) cuando se debe tomar un dato de memoria, mientras que se relaciona con la IP si debe romperse la secuencia normal del programa.

Dentro de esta etapa, **el FLAG F se encuentra igualado a 1.**

Proceso (Fase de Ejecución)

A cada código de instrucción le corresponde una serie de microoperaciones propias.

Dentro de esta etapa, **el FLAG F se encuentra igualado a 0.**

**Instrucciones:**

* **LDA WORD:** Su COP es 0001. Transfiere el dato almacenado en la palabra WORD al MDR y de ahí a un acumulador en la ALU.
* **ADA WORD:** Realiza suma binaria entre el dato que contiene WORD (transferido a MDR) y lo suma con el dato almacenado en el acumulador de la ALU.
* **SHR:** Realiza operación de división, corre un bit a derecha el dato almacenado en el acumulador de la ALU.
* **SHL:** Realiza operación de multiplicación, corre un bit a izquierda el dato almacenado en el acumulador de la ALU.
* **STA WORD:** Almacena en el dato de la palabra WORD el dato contenido en el acumulador de la ALU.
* **HLT:** Inhabilita la generación de operaciones. Determina el fin de programa.
* **JMP WORD:** realiza un salto de instrucciones hasta la palabra WORD. Es incondicional.
* **SNA WORD:** realiza un salto de instrucciones hasta la palabra WORD solamente si el dato que se encuentra en el acumulador es negativo.
* **SZA WORD:** realiza un salto de instrucciones hasta la palabra WORD solamente si el dato que se encuentra en el acumulador es 0.
* **SCA WORD:** realiza un salto de instrucciones hasta la palabra WORD solamente si el flag de acarreo del dato que se encuentra en el acumulador es 1.
* **ANA WORD:** realiza la operación lógica AND entre el dato almacenado en WORD y el dato que se encuentra en el acumulador de la ALU.
* **XOA WORD:** realiza la operación lógica XOR entre el dato almacenado en WORD y el dato que se encuentra en el acumulador de la ALU.
* **INP:** permite ingresar un dato desde un periférico de entrada.
* **OUT:** permite extraer un dato hacia un periférico.

**Sincronización del tiempo**

10\*…\*-3 = 1 milisegundo 10\*…\*3Hz = KHz

10\*…\*-6 = 1 microsegundo 10\*…\*6Hz = MHz

10\*…\*-9 = 1 nanosegundo 10\*…\*9Hz = GHz

Frecuencia = 1/tiempo

Tiempo = 1/frecuencia

Hz = 1/seg

Ciclo de Memoria = Tiempo de Acceso (memoria de lectura no destructiva)

Ciclo de Memoria = Tiempo de Acceso + Tiempo de Restauración (otras memorias)

Se denomina **ciclo de reloj** al tiempo transcurrido entre 2 pulsos adyacentes. Las computadoras manejadas por ciclos de reloj son **sincrónicas**. El tiempo de acceso a memoria es el tiempo que tarda la unidad de control en hallar la información y trasmitirla al MDR.

**Unidad Aritmetico-Lógica**

Es la unidad donde se operan los datos.

Contiene flags que se actualizan luego de cada operación.

Se produce overflow al superar la capacidad del acumulador.

El desplazamiento de los bits sucede por: división, multiplicación, testeo de bits.

2\*…\*10 = 1K

2\*…\*20 = 1M

2\*…\*30 = 1G

Practica

* Para determinar la cantidad de chips necesarios, dividimos el tamaño de la memoria principal por el tamaño de cada chip.
* Para obtener la cantidad de filas dividimos el tamaño de las filas de memoria principal por el tamaño de filas del chip. Los bits que la identifican son igual a la raíz de la cantidad de filas. Chip/cant. Filas = X chips en paralelo.
* Para obtener la cantidad de bits de dirección dentro del chip, utilizamos el tamaño del chip (2k = 2\*…\*11 = 11 bits de dirección).
* Para obtener la cantidad de bits de memoria realizamos lo mismo que con el chip, pero con el tamaño de memoria.