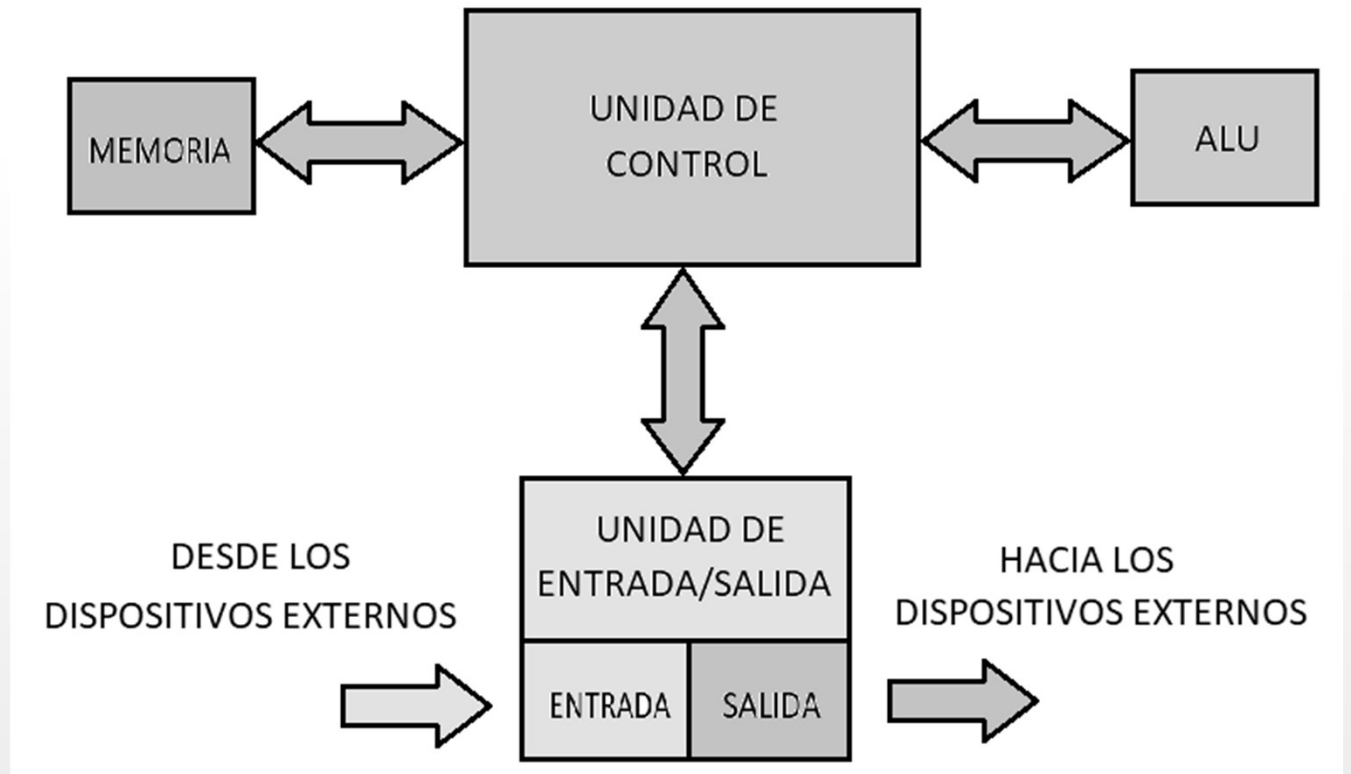


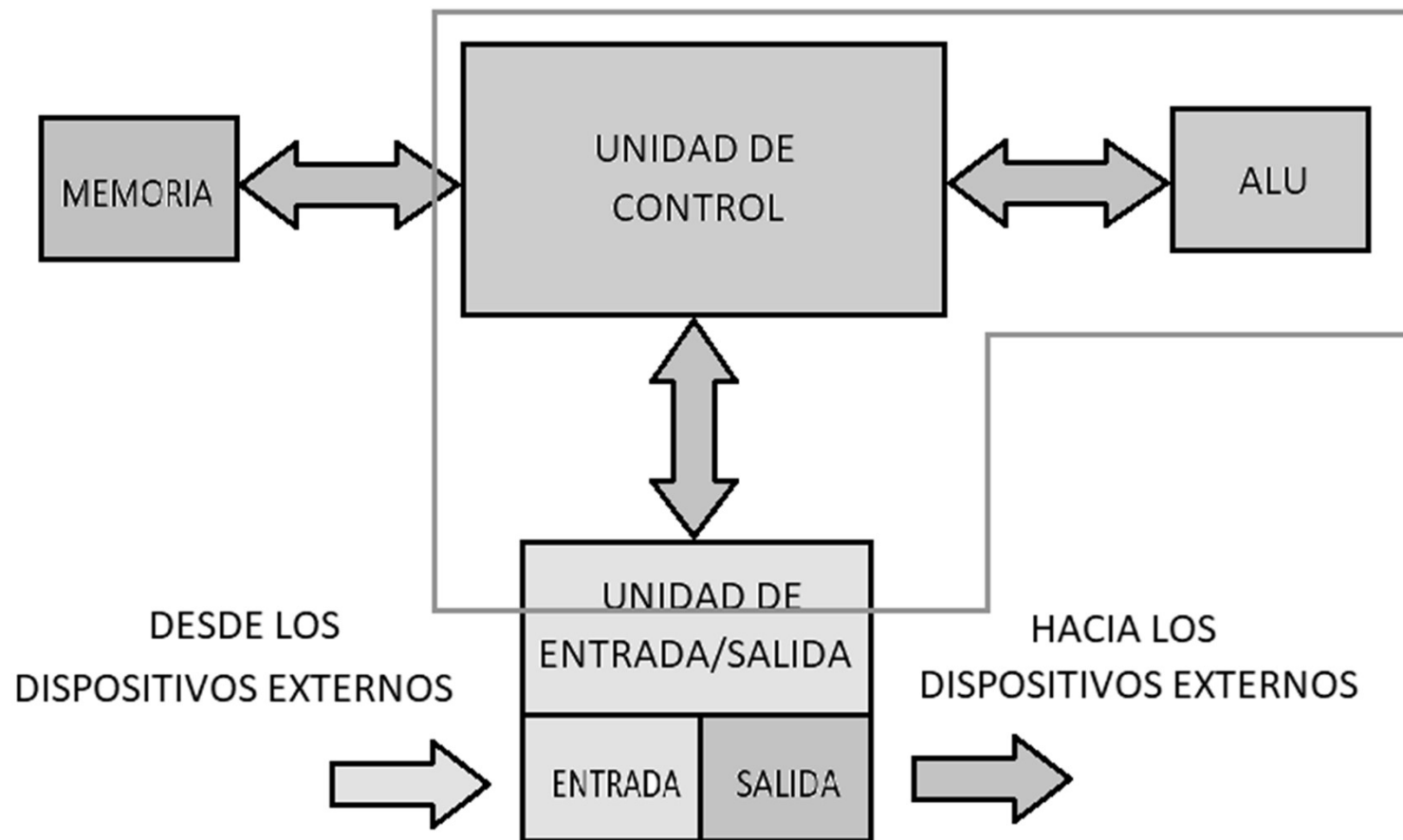
MICROCOMPUTADORA

PARTES BÁSICAS DE UNA MICROCOMPUTADORA

- MICROPROCESADOR
 - UNIDAD ARITMÉTICA Y LÓGICA
 - UNIDAD DE CONTROL
 - REGISTROS
- MEMORIA
 - ROM
 - RAM
- SISTEMA DE ENTRADAS Y SALIDAS


MICROCOMPUTADORA







MEMORIA

- Contiene las instrucciones para la ejecución del programa.
 - Contiene los datos usados en el programa.
 - Un programa es un grupo de instrucciones que le indican a la microcomputadora qué hacer con los datos.
- 

UNIDAD ARITMÉTICA Y LÓGICA

- Realiza las operaciones aritméticas y lógicas requeridas por las rutinas del programa.
- Genera los bits de estado o códigos de condición (*STATUS* bits, *Condition codes*), que le dan la capacidad de tomar decisiones a la microcomputadora.

Sistema de entradas y salidas

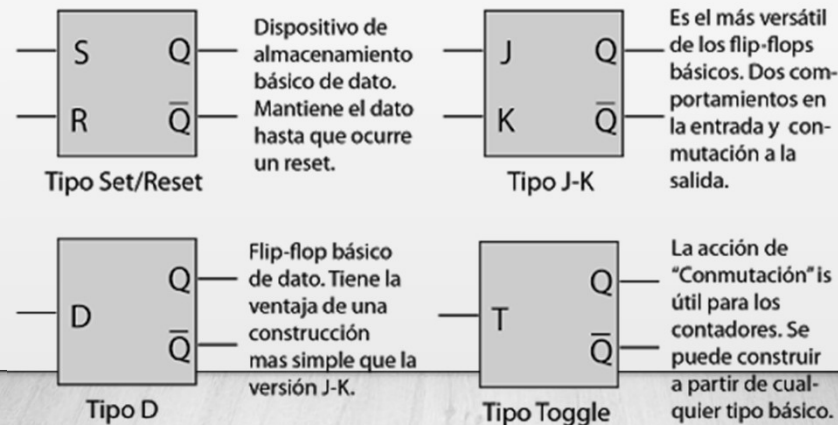
- Controla la comunicación entre la microcomputadora y los dispositivos externos.
- La microcomputadora debe recibir datos información de *estado* del exterior, por ejemplo de sensores, unidades de almacenamiento, etc.
- La microcomputadora debe producir salidas que depende del programa y de los datos que recibe.
- La salidas pueden ser datos a desplegar, almacenar o bien pulsos eléctricos que activan actuadores para regular procesos físicos.

UNIDAD DE CONTROL

- Formada por componentes electrónicos como flip-flops, registros, etc. para regular todos los procesos en la ejecución del algoritmo indicado mediante el programa.
- Genera la secuencia adecuada de eventos que ocurren durante la ejecución de cada una de las instrucciones procesadas.
- Requiere de una fuente de reloj para crear los eventos de traer la instrucción, decodificarla y ejecutarla.

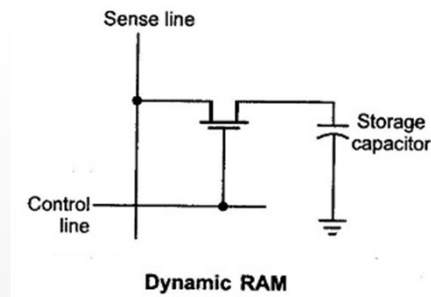
Conceptos sobre la unidad de memoria.

- Todas las microcomputadoras requieren de memoria para almacenar los programas y los datos.
- Dependiente del tipo de memoria pueden estar construidas con materiales magnéticos, flip-flops, capacitores como unidad elemental de almacenamiento.
- La memoria estática suele estar conformada por flip-flops, un flip-flop puede almacenar un bit de memoria.



Conceptos sobre la unidad de memoria, continuación.

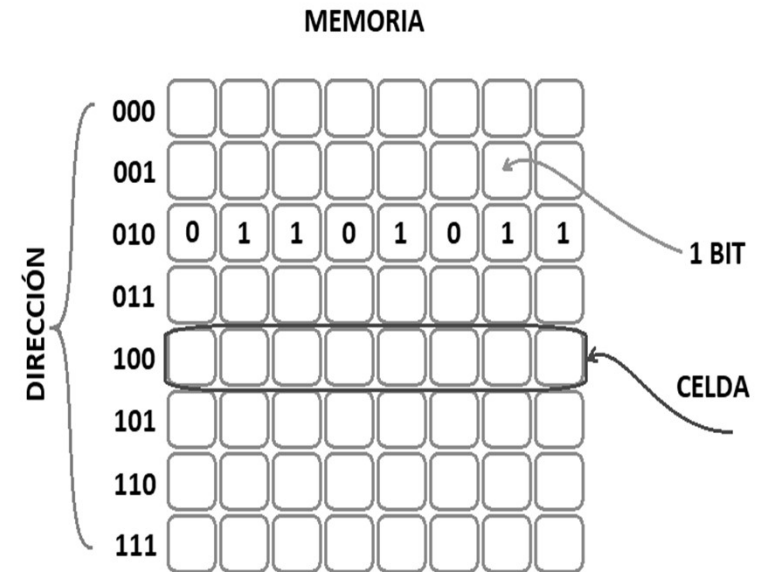
- La memoria dinámica suele estar fabricada con capacitores, requiere de una lógica de refresco.



- Típicamente los bits están organizados en palabras de 4, 8, 16, etc. bits, siendo común el tamaño de palabra generalmente de 8 (byte) o 16 bits.
- Las microcomputadoras están diseñadas para el manejo de memoria en unidades de palabra, es decir que transfieren y procesan una palabra a la vez, lo que define el tamaño del bus de datos.

Conceptos sobre la unidad de memoria, continuación.

- Organizada en grupos de bits del tamaño de palabra utilizado.
- Cada celda tiene una dirección única, mediante la cuál se puede realizar la acción de lectura o escritura.
- Puede almacenar datos o instrucciones.
- Las instrucciones suelen ocupar 1, 2, 3, etc. bytes.
- Cada palabra en memoria tiene dos parámetros, su **dirección** y el **dato**.



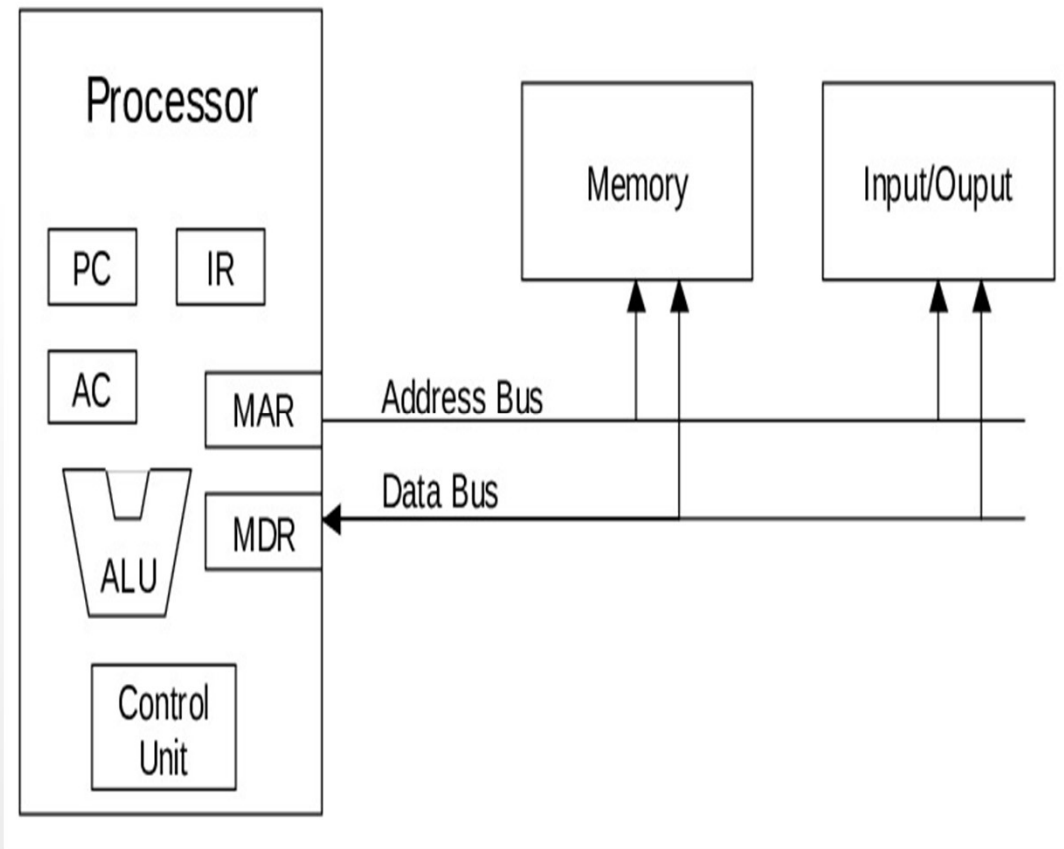
Se puede almacenar $2^8 = 256$ combinaciones distintas

Se requieren $\log_2 8 = 3$ bits para direccionar las 8 celdas

Conceptos sobre la unidad de memoria, continuación.

El proceso de acceder a un dato en la memoria requiere dos registros:

- MAR (Memory Address Register), que contiene la dirección de la palabra accedida en el momento y
- MDR (Memory Data Register) que contiene el dato que esta siendo leído o escrito, estos registros se pueden considerar parte de la Unidad de Memoria o de la Unidad de Control.



Conceptos sobre la unidad de memoria, continuación.

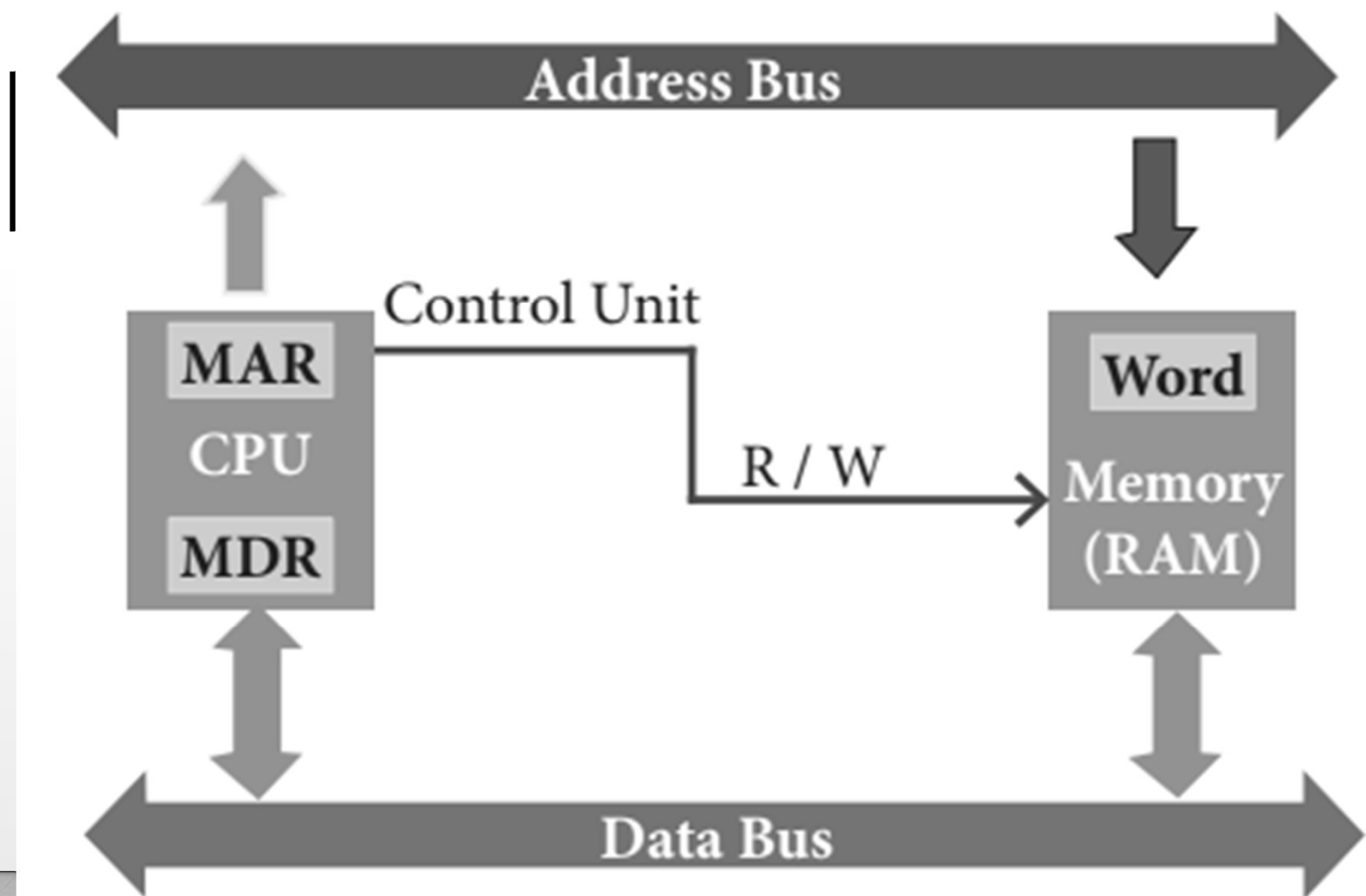
La memoria puede operar en dos modos:

- LECTURA

- 1. La localidad a ser leída se coloca en el registro MAR.
- 2. Se da una orden de LECTURA
- 3. El dato se transfiere desde la palabra direccionada en la memoria al registro MDR, en donde es accesible al microprocesador.

- ESCRITURA

- 1. La localidad a ser escrita es cargada al registro MAR.
- 2. El dato a ser escrito se coloca en el registro MDR.
- 3. Se da la orden de ESCRITURA (por medio de la señal Read/Write), lo que transfiere el dato desde el registro MDR al la localidad de memoria seleccionada.



Memoria RAM

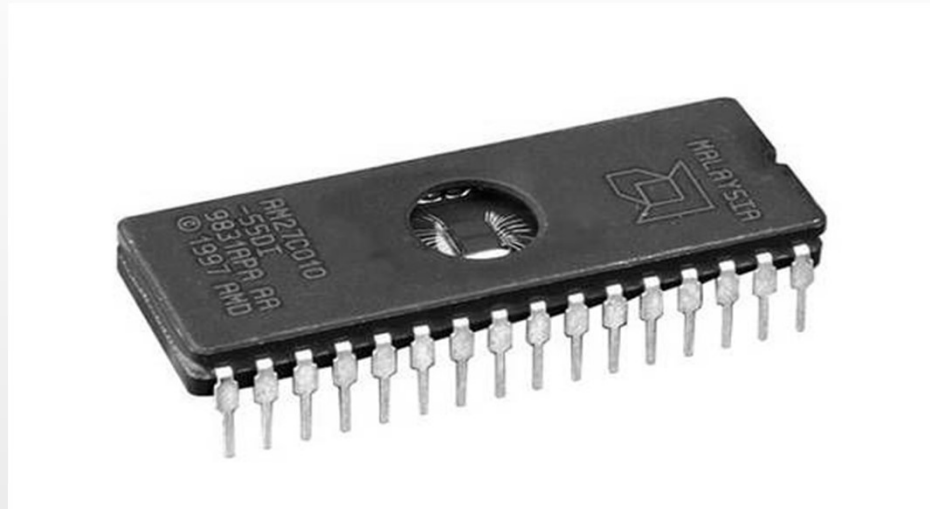
- El término RAM (Random Access Memory), que significa memoria de acceso aleatorio puede entenderse mejor como memoria de lectura / escritura.
- Construida principalmente con compuertas MOS, almacenan su información en flip-flops o capacitores, designados como memoria RAM estática y dinámica.
- La RAM estática requiere más elementos electrónicos en la placa de silicio para cada celda (bit).
- La RAM dinámica, al usar capacitores para almacenar un bit, requiere de pulsos frecuentes para refrescar el dato (mantener el dato binario almacenado).
- Ambos tipos son volátiles, al perder la alimentación eléctrica pierden la información almacenada.
- Cada reinicio los datos deben reescribirse a la memoria.

Memoria ROM

- La memoria ROM (Read Only Memory), memoria de solo lectura, solo se puede leer durante la operación de la microcomputadora.
- Tipo de memoria no volátil o permanente, no pierde la información ante la falta de alimentación eléctrica.
- Si un programa esta en una memoria ROM, el microprocesador puede reiniciar ante una falla de energía eléctrica sin tener que cargar de nuevo su programa (*bootstrap*) .
- La operación de lectura es similar a la lectura de datos de una memoria RAM.
 - 1. La localidad a ser leída se coloca en el registro MAR.
 - 2. Se da una orden de LECTURA a la memoria ROM.
 - 3. El dato se transfiere desde la palabra direccionada en la memoria al registro MDR, en donde es accesible al microprocesador.

Memoria ROM

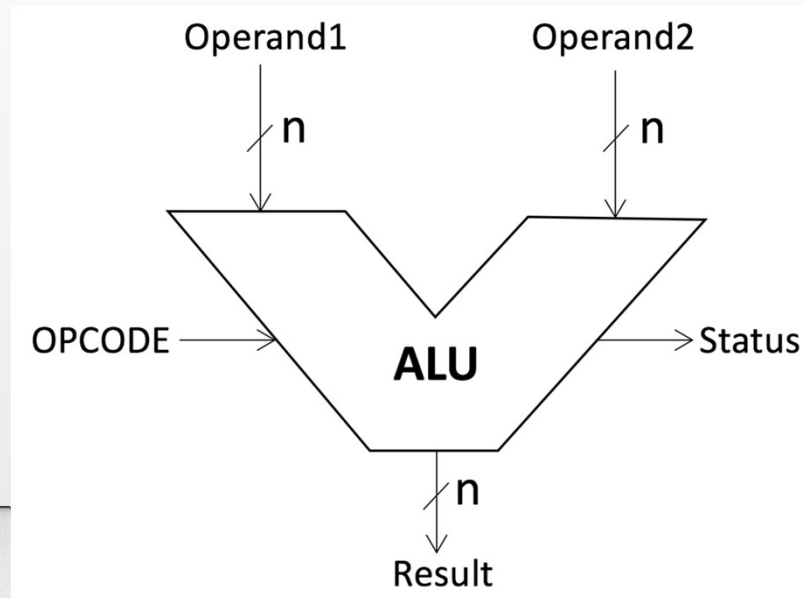
- En sistemas de diseño o desarrollo se usan memorias PROM (Programmable ROM), en donde el contenido de las palabras puede ser cambiado mediante diferentes métodos, dependiendo de la tecnología de la memoria PROM, permitiendo el desarrollo y depuración de los programas.



UNIDAD ARITMÉTICA Y LÓGICA

- Realiza todas las operaciones aritméticas (suma, resta, etc.) y lógicas (AND, OR, etc.) requeridas por la microcomputadora.
- Requiere de dos operandos como entradas, cada operando contiene tantos bits como el tamaño básico de la palabra de la microcomputadora.
- La mayoría de las ALUs de los microprocesadores implementan las siguientes operaciones:

- Suma
- Resta
- OR
- AND
- XOR
- Complemento
- Corrimiento
- Otras...



UNIDAD ARITMÉTICA Y LÓGICA, CONTINUACIÓN

- La ALU genera un resultado del mismo tamaño en bits como el tamaño de los operandos.
- Genera también banderas que indican características del resultado de la operación realizada, estas banderas están contenidas en un registro que se puede llamar *Condition Code, Status*, etc.
- Se pueden implementar operaciones aritméticas más complejas mediante Hardware (diseñado e implementado desde el fabricante) o bien desde el Software, como multiplicación y división, que pueden realizarse con sumas y corrimientos, mediante subrutinas de software procedentes de algoritmos.

UNIDAD DE CONTROL

- La función de la Unidad de Control es regular la operación de la microcomputadora .
- Decodifica la instrucción y causa todos los eventos para que se procesen las instrucciones mediante las señales en el momento adecuado y en el orden correcto.
- Consiste de un grupo de flip-flops, registros, circuitos de control para la base de tiempo.
- En un microcomputadora elemental los siguientes registros deben formar parte de la Unidad de Control:

UNIDAD DE CONTROL, CONTINUACIÓN

- En un microcomputadora elemental los siguientes registros deben formar parte de la Unidad de Control:
 - 1. Registros **MAR** (Memory Address Register) y **MDR** (Memory Data Register).
 - 2. Registro **PC** (Program Counter), contiene tantos bits como el registro MAR, contiene la dirección de la próxima instrucción a ser ejecutada. Generalmente se incrementa durante la ejecución de una instrucción, así es como apunta siempre a la próxima instrucción a ser ejecutada.
 - 3. El registro de instrucciones (**Instruction Register**), contiene la instrucción que está siendo ejecutada en ese momento.
 - 4. Registro del decodificador de instrucción (**Instruction Decoder**), este registro decodifica la instrucción que está siendo ejecutada en el momento. Su entrada proviene de *Instruction Register*.

UNIDAD DE CONTROL, CONTINUACIÓN

- 5. Registro Acumulador (Accumulator), contiene tantos bits como el registro MDR. Contiene el operando básico utilizado en cada instrucción.
- En las operaciones de la ALU donde se requieren dos operandos, uno de los operandos es cargado al acumulador como resultado de una previa instrucción (es decir, se usa antes una instrucción de movimiento de dato, generalmente desde la memoria, para tener uno de los operandos en el acumulador, así se puede realizar la operación que requiere de dos operandos).
- El resultado de la operación entre ambos operandos normalmente va al acumulador lo que sirve como operando para una siguiente instrucción o puede ir a otro registro o a la memoria.

EJECUCIÓN DE UNA SUBROUTINA SENCILLA

- El primer paso que realiza el microprocesador para ejecutar instrucciones es buscar (FETCH) el byte contenido en la dirección indicada en el PC. Muchos microprocesadores usan este primer byte como OPCODE (Operation Code), que es un código de operación que le indica al microprocesador qué hacer (qué instrucción ejecutar). Un *opcode* de 8 bits permite tener hasta 256 diferentes instrucciones.
- En muchos casos, el microprocesador luego realiza un segundo ciclo *fetch* para obtener la dirección requerida por la instrucción. Es así como con el *opcode* y la dirección obtenidos se puede ejecutar la instrucción.
- El número de bytes y ciclos requeridos para cada instrucción depende de la instrucción en sí misma y del modo de la instrucción.

Ejemplo de ejecución de un sencillo programa

- Consideremos que el microprocesador debe sumar dos números, antes de que el programa inicie los números y el programa deben cargarse en memoria.
- Para nuestro ejemplo vamos a sumar los números 2 y 3, de forma arbitraria les asignaremos las direcciones de memoria 80 y 81 respectivamente, también la localidad 82 para almacenar el resultado.
- El programa o grupo de instrucciones para sumar ambos números es:
 - LOAD | primera
 - 80 | instrucción
 - ADD | segunda
 - 81 | instrucción
 - STORE | tercera
 - 82 | instrucción

Ejemplo de ejecución de un sencillo programa, continuación

- La primera instrucción, LOAD, carga el contenido de la localidad 80, el número 2, y lo guarda en el Acumulador.
- La segunda instrucción lee el contenido de la localidad 81, el número 3 y lo suma con el Acumulador, que contiene el número 2, la suma, 5, se queda en el registro Acumulador.
- La tercera instrucción hace que el contenido del Acumulador se escriba en la memoria, en el registro 82, completando el programa.

Ejecución del programa ejemplo desde el hardware

- Describiremos como los registros y la ALU dentro del microprocesador se coordinan para ejecutar nuestro programa de ejemplo.
- Las instrucciones deben estar alojadas en localidades específicas de la memoria, asumamos que el inicio del programa esta en la dirección 10, así la localidad 10 contiene el *opcode* LOAD, la localidad 11 contiene el número 80, que se interpretará como una dirección, la localidad 12 contiene el opcode ADD, así hasta completar nuestro programa.
- Antes de comenzar el programa, también asumamos que la dirección de inicio para el PC ya ha sido escrita con el valor de 10.
- Con estas condiciones iniciamos la ejecución del programa.

Ejecución del programa ejemplo desde el hardware, continuación

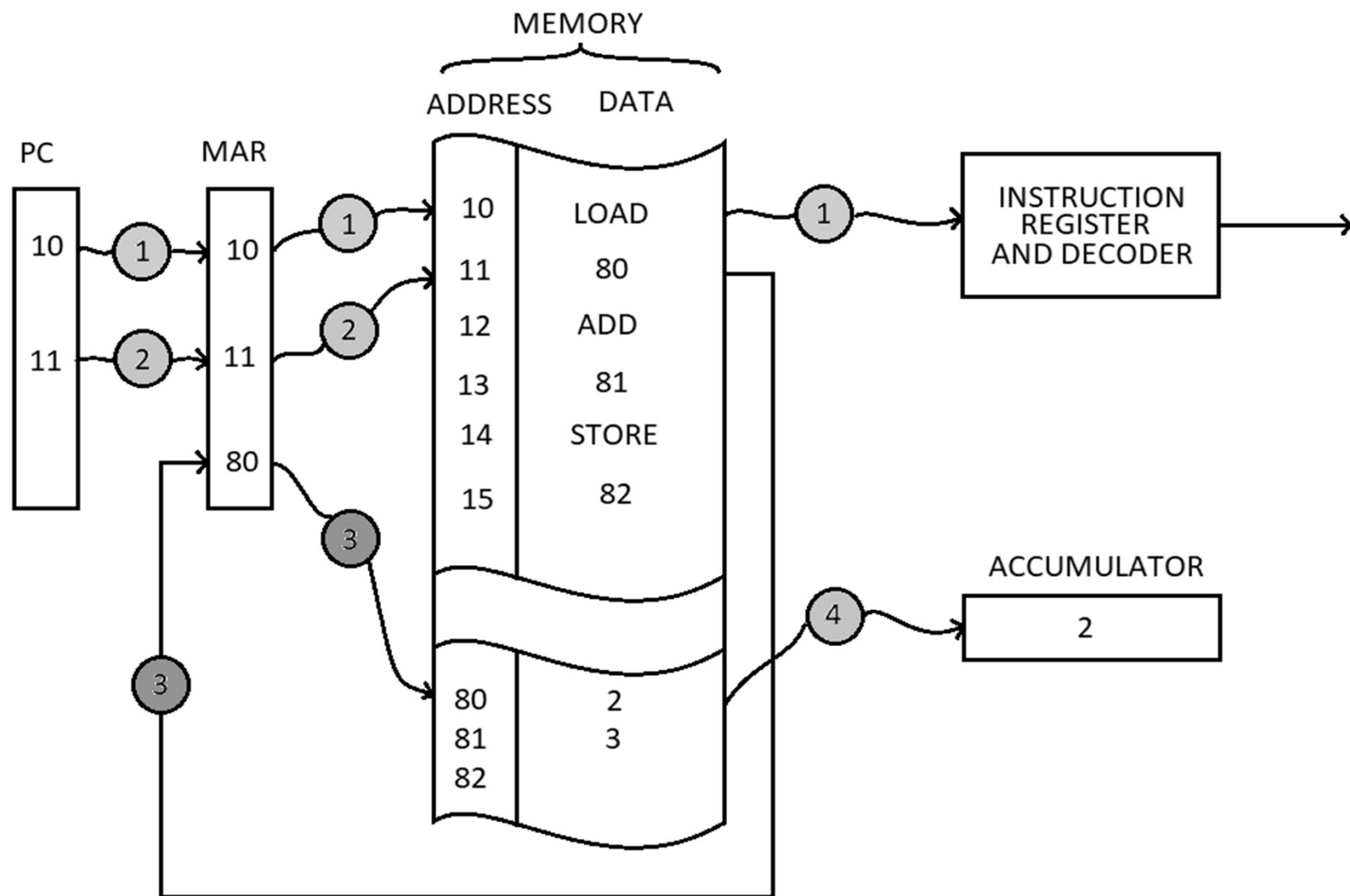
- 1. El PC es transferido al MAR y la memoria es leída en la localidad 10.
- 2. Como esta es la primera parte del ciclo *fetch*, el dato leído es colocado en el *Instruction Register* y el *Decoder Instruction* determina que se trata de una instrucción LOAD.
- 3. El microprocesador incrementa el PC, este se valor se transfiere a MAR y lee el contenido del registro 11 de memoria, el número 80, el que coloca en MAR.

Ejecución del programa ejemplo desde el hardware, continuación

- 4. El microprocesador esta listo para ejecutar la instrucción LOAD, lo hace mediante leer el contenido de la localidad 80, que ya esta en MAR y colocándolo en el Acumulador, que ahora contienen el valor 2.
- 5. La ejecución de la instrucción LOAD se ha completado, el PC se incrementa de nuevo, contiene ahora el valor 12 y es colocado en MAR.
- 6. El código ADD es traído de la localidad 12.
- 7. El PC se incrementa nuevamente y la localidad 81 de memoria es leída y colocada en MAR.

Ejecución del programa ejemplo desde el hardware, continuación

- 8. Se ejecuta ahora la operación ADD, el contenido de 81, el número 3 se lee y es sumado al contenido del Acumulador . Se usa la ALU en este proceso. El resultado, 5, se escribe en el Acumulador y así se completa la instrucción ADD.
- 9. Se incrementa de nuevo el PC y se transfiere al MAR, el *opcode* de la instrucción STORE es traída y decodificada.
- 10. La dirección para la instrucción STORE, 82, es traída desde la localidad 15 y colocada en MAR.
- 11. Ahora se ejecuta la instrucción STORE, mediante tomar el contenido del registro Acumulador, 5, y escribiéndolo en la localidad 82 de memoria.



EJEMPLO, TABLA DE CAMBIOS EN REGISTROS

PC	MAR	MEMORIA	ACUMULADOR	
10	10	LOAD	X	
11	11	80	X	INSTRUCCIÓN LOAD
11	80	2	2	
12	12	ADD	2	INSTRUCCIÓN
13	13	81	2	ADD
13	81	3	5	
14	14	STORE	5	
15	15	82	5	INSTRUCCIÓN STORE
15	82	5	5	

GLOSARIO DE TERMINOS SOBRE MICROCOMPUTADORAS

- ACCES TIME
- ACCUMULATOR
- ADDRESS
- ARITHMETIC LOGIC UNIT (ALU)
- BUS
- CONTROL UNIT