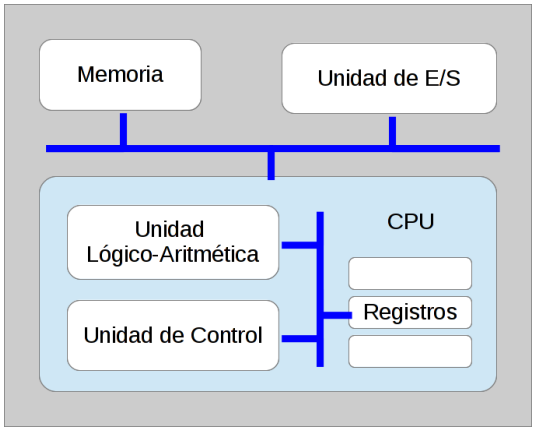
Modelo Computacional Binario Elemental

En esta unidad describiremos la arquitectura de las computadoras en general, y de una computadora hipotética en particular, que llamaremos **MCBE (Modelo Computacional Básico Elemental)**. Aunque el MCBE es sumamente rudimentario, comparte las características esenciales de casi todas las computadoras digitales, y está inspirada en máquinas que han existido realmente.

Arquitectura de una computadora

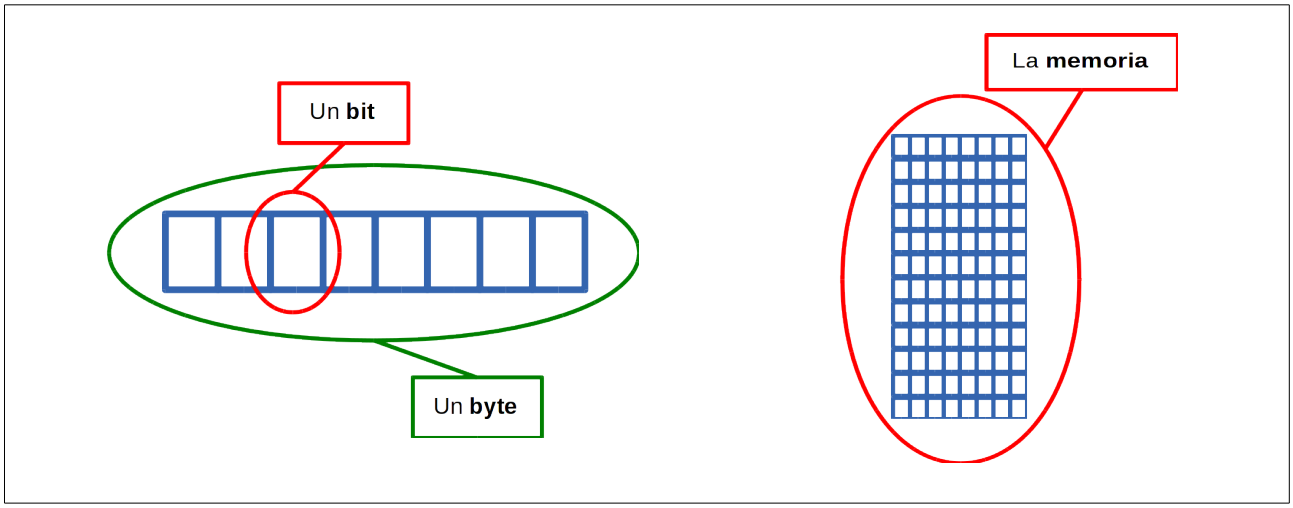
El modelo de arquitectura que usaremos para describir nuestra computadora ideal es el **modelo de Von Neumann**[1]. En él se distinguen ciertas unidades funcionales principales: **Memoria, CPU, Unidad de E/S**, conectadas mediante **buses** o canales de comunicación.



*Las tres unidades principales **Memoria, Unidad de E/S y CPU** están conectadas mediante el **bus de sistema**. Los componentes internos de la CPU, mediante un **bus interno**.*

Memoria

La **memoria** de la computadora es un conjunto de circuitos **biestables**, cada uno de los cuales puede almacenar **un bit** de información. Esos circuitos de la memoria están dispuestos en celdas de **ocho** biestables. Cada una de estas celdas ocupa una **posición** de memoria, que puede almacenar **un byte** de información.



1 http://es.wikipedia.org/wiki/Arquitectura_de_von_Neumann

Las posiciones de la memoria se encuentran numeradas consecutivamente **a partir de 0**, por lo cual podemos imaginarnos que la memoria es algo así como una alta estantería vertical, de muchos estantes numerados. Cada uno de esos estantes, de ocho casilleros, será capaz de guardar un determinado contenido.

Como cada biestable representa un bit y cada posición de memoria representa un byte, a veces esos circuitos y celdas de circuitos se llaman directamente **bits y bytes de la memoria**. La posición relativa de cada byte se llama su **dirección**. Al acceder a un dato contenido en una posición de memoria, ya sea para leerlo o para modificarlo, necesariamente tenemos que mencionar su dirección; cuando hacemos esto decimos que **direccionamos** esa posición de la memoria.

Es costumbre representar las direcciones de memoria con la posición inicial (la dirección 0) en la base del diagrama.

CPU

Las siglas **CPU** se refieren a *Central Processing Unit*, **Unidad Central de Procesamiento**. La CPU es un dispositivo complejo, formado por varios componentes, que al activarse es capaz de ejecutar **instrucciones** que transformarán la información almacenada en la memoria.

La CPU, a su vez, contiene sus propias unidades funcionales: la **Unidad de Control (UC)** y la **Unidad Lógico-aritmética (**sigla en inglés: **ALU)**. Las unidades cuentan con **registros** especiales, que son espacios de almacenamiento, similares a los de la memoria, pero situados en otro lugar de la circuitería.

- Unidad de Control
  - Su función es gobernar la actividad de la CPU, indicando cuál es la próxima instrucción a ejecutar y de qué modo debe cumplirse.
- Unidad Lógico-aritmética
  - Contiene la circuitería necesaria para ejecutar operaciones matemáticas y lógicas.

Unidad de Entrada/Salida

La **Unidad de Entrada/Salida** (**UE/S**) conecta a la computadora con dispositivos como teclados, pantallas o impresoras. La Unidad de Entrada/Salida se requiere para poder comunicar la máquina con el resto del mundo. Si no existiera la UE/S, la máquina no podría recibir los datos con los que tiene que trabajar, ni podría hacer saber al usuario de la máquina los resultados de sus cálculos.

Buses

En las computadoras, las unidades funcionales comparten datos, y para eso están relacionadas mediante **buses**, que son canales de comunicación que permiten transferir datos entre las unidades.

El MCBE, una computadora elemental

Repitamos que esta computadora **no tiene existencia real**: es tan poco potente que hoy ya no sería razonable implementarla, salvo por motivos de enseñanza. Pero, aun tan simple como es, puede ejecutar tareas de complejidad bastante interesante y nos servirá para mostrar muchos de los problemas relacionados con la arquitectura y la organización de las computadoras reales. El MCBE

es un ejemplo muy sencillo de **computador de programa almacenado**[2].

Descripción detallada del MCBE

Recordemos que las computadoras **no toman decisiones por sí solas**. Todo lo que hacen **está determinado por el programa almacenado**, cuya escritura es responsabilidad del usuario. Especificaremos entonces con todo detalle, en el diseño de esta máquina elemental, cuál será la respuesta de la computadora a cada instrucción de un programa.

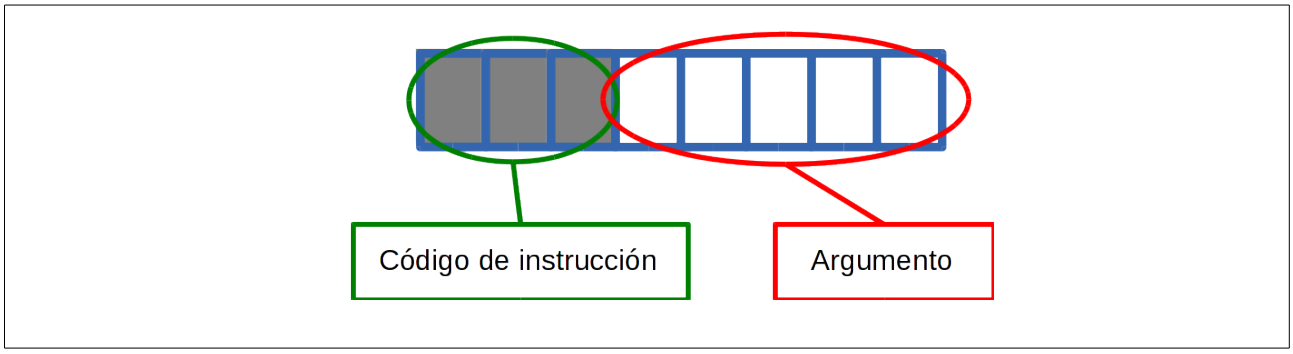
Instrucciones

Hay tan sólo **ocho diferentes instrucciones** que puede seguir esta máquina. Algunas sirven para realizar cálculos; otras, para mover datos de un lugar a otro; otras, para modificar el curso de las acciones a seguir por el programa. En cuanto a operaciones aritméticas, el MCBE sólo sabe **sumar y restar** datos. Sin embargo, basándose en esas únicas dos operaciones, puede seguir un **programa** que implemente otras operaciones más complejas.

Para ayudar a los programadores, las instrucciones reciben **nombres mnemotécnicos**, derivados del inglés (**LD, ST, ADD, SUB, JMP, JZ, HLT, NOP**). Se acostumbra usar estos nombres, u otros muy similares, en la programación de máquinas parecidas de la realidad. Sin embargo, estos nombres únicamente sirven para que los humanos comprendan mejor el modelo y su programación. El MCBE los ignora completamente y **sólo utiliza la expresión binaria** de esas instrucciones, residente en la memoria. La Unidad de Control de la CPU es quien interpretará cada una de las posiciones de memoria, ya sea como un dato numérico, o como una instrucción.

Interpretación de instrucciones

Cuando el byte contenido en una posición de memoria represente una instrucción, los **tres bits de orden más alto** (los tres bits situados **más a la izquierda**) indicarán el **código** de la operación. En el caso de ciertas instrucciones, los restantes bits en el byte (los **cinco bits de orden más bajo**) representarán un **argumento** para la instrucción, es decir, un dato para que esa instrucción trabaje.



Argumentos

Cuando la instrucción utilice argumentos, éstos pueden ser de una de dos clases: **direcciones y desplazamientos**.

- Cuando la instrucción sea de transferencia entre el acumulador y la memoria (LD, ST, ADD, SUB) el argumento será una **dirección**, y los cinco bits de orden bajo codificarán esa dirección. La dirección servirá para ir a buscar un dato a la memoria, o para acceder a una

2 http://es.wikipedia.org/wiki/Computador_de_programa_almacenado

posición y dejar allí el resultado de un cálculo.

- Normalmente, luego de cumplir una instrucción, el MCBE continúa con la que se encuentre en la posición siguiente en la memoria. Sin embargo, ciertas instrucciones pueden alterar esa rutina. Las **instrucciones de salto** (JMP, JZ) sirven para desviar el curso de la ejecución. En estos casos el argumento representará un **desplazamiento**, y será interpretado como un entero con signo, en representación **complemento a dos**. Un desplazamiento es una cantidad de bytes que deben sumarse o restarse al PC, para **transferir el control** a una posición diferente a la siguiente.

Ciclo de instrucción

El **ciclo de instrucción** es la rutina que continuamente ejecuta el MCBE, leyendo y ejecutando las instrucciones del programa almacenado.

Al inicio de la operación, la máquina comenzará leyendo la posición 0, interpretándola como una instrucción y ejecutándola, según la especificación del ciclo de instrucción. El resto del comportamiento de la máquina depende de qué secuencia particular de instrucciones y datos (es decir, qué **programa**) haya preparado el usuario en la memoria.

El ciclo de instrucción se realiza continuamente hasta encontrar una instrucción **HLT**, y siempre de la misma manera:

1. Se carga en el registro IR la instrucción cuya dirección está en el registro PC.
2. Se decodifica la instrucción.
    - La máquina examina los tres primeros bits del IR, identificando de **qué instrucción** del conjunto de instrucciones se trata.
    - El resto de los bits, cuando corresponda, se utilizan como argumento de la instrucción, representando **una dirección o un desplazamiento** según se trate.
3. Se **ejecuta** la instrucción. Cada instrucción tiene un efecto determinado sobre los registros o la memoria, que se detalla en la tabla adjunta.

Luego de la ejecución de la instrucción, y según cuál haya sido esa instrucción, los registros tienen posiblemente otros valores y ha ocurrido, posiblemente, algún efecto sobre la memoria. Con ese nuevo estado de la máquina, el MCBE se dirige a la siguiente instrucción a ejecutar.

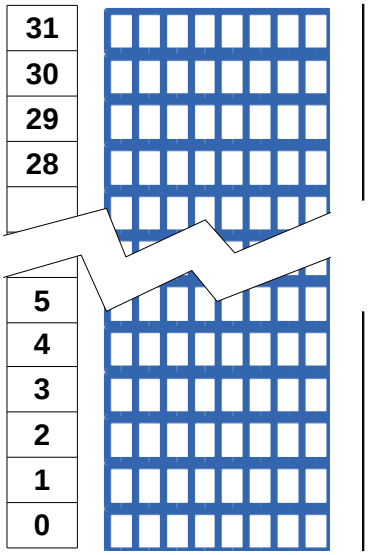
Detalles operativos del MCBE

- La Unidad de Control de la máquina MCBE posee dos registros especiales, llamados **PC** (por *Program Counter*, Contador de Programa[3]) e **IR** (por *Instruction Register*, Registro de Instrucciones[4]).
    - La función del PC es contener la dirección de la próxima instrucción a ejecutar.
    - El IR contiene el valor de la última instrucción que se ha leído de la memoria.
- La Unidad Lógico-Aritmética de la máquina dispone de un registro especial llamado **A** (por *Acumulador*).

3 http://es.wikipedia.org/wiki/Contador_de_programa
4 http://es.wikipedia.org/wiki/Registro_de_instrucción

- El acumulador es un lugar de trabajo para efectuar aritmética binaria, y sirve de zona de comunicación entre los registros y la memoria.

- La máquina tiene **32 posiciones** de memoria. Cada posición aloja un byte de información. La UE/S utiliza dos de estas posiciones (ver Figura).
- Cada vez que un valor se copia del acumulador A a una posición de memoria B cualquiera, **el valor de A no se altera**. Sin embargo, el valor anterior de B **se pierde** y la posición B pasa a contener un valor igual al de A.
- Inversamente cuando se copia un valor desde una posición de memoria B al acumulador (el valor de B no se altera, pero A cambia su valor por el de B).
- La máquina puede cargarse con un programa escrito por el usuario, y a continuación este programa se ejecuta.
- Al momento previo a la ejecución de un programa, todos los registros están inicialmente en 0.



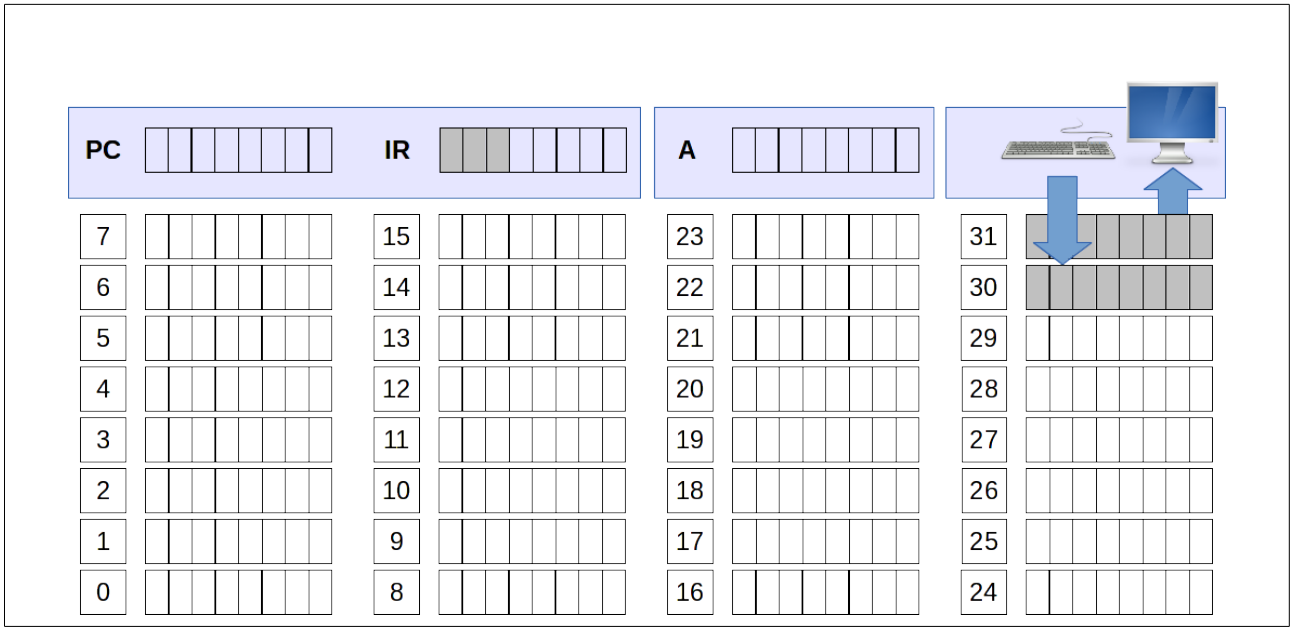
En la memoria se distinguen dos posiciones especiales, con direcciones 30 y 31. Estas posiciones sirven para Entrada/Salida, es decir, para comunicación de la máquina con otros dispositivos.

- La posición 30 es de sólo lectura, y sirve para ingresar datos (Entrada) a los programas. Cuando la máquina ejecuta una instrucción de lectura de la dirección 30, el programa se detiene hasta que el usuario de la máquina ingrese un dato.
- Inversamente, la posición 31 es de sólo escritura. Cuando se escribe un dato en la posición 31, el programa se detiene hasta que el dato sea recogido por un dispositivo de visualización. Ese dispositivo se encargará de emitir el dato (Salida) para que pueda verlo el usuario.

Preguntas

- ¿Cuál es la dirección de la primera instrucción que ejecutará la máquina?
- El MCBE, ¿puede encontrar una instrucción que no sea capaz de decodificar?
- Supongamos que hemos almacenado en la posición 14 un dato numérico que representa la edad de una persona. ¿Qué pasa si en algún momento de la ejecución el PC contiene el número 14? ¿Qué pasará si esa persona tiene 33 años? ¿Qué pasará si tiene 65? ¿Y si tiene menos de 20?
- ¿Qué pasa si el programa no contiene una instrucción HLT?
- ¿Podría aumentarse la capacidad de memoria del MCBE? ¿Esto requeriría algún cambio adicional a la máquina?
- ¿Cómo se podría aumentar la cantidad de instrucciones diferentes del MCBE? ¿Esto tendría algún efecto sobre la longitud de los programas que puede correr la máquina?

Diagrama estructural del MCBE



Conjunto de instrucciones

Instrucción	*Cód.*	*Efecto sobre memoria y registros*	*Efecto sobre el PC*
LD <dirección>	010	El argumento se trata como una dirección. El contenido de esa dirección se copia en el acumulador.	Se incrementa en 1.
ST <dirección>	011	El argumento se trata como una dirección. El contenido del acumulador se copia en esa dirección.	Se incrementa en 1.
ADD <dirección>	100	El argumento se trata como la dirección de un dato, que será sumado al acumulador.	Se incrementa en 1.
SUB <dirección>	101	El argumento se trata como la dirección de un dato, que será restado al acumulador.	Se incrementa en 1.
JMP <desplazamiento>	110	Salta <desplazamiento> bytes. El argumento se trata como un desplazamiento, es decir, un entero con signo.	El desplazamiento será sumado al PC.
JZ <desplazamiento>	111	Salta <desplazamiento> bytes en forma condicional. El argumento se trata como un desplazamiento, es decir, un entero con signo.	Si el acumulador contiene un valor 0, el desplazamiento será sumado al PC. En caso contrario el PC se incrementa en 1.
HLT	001	Detiene la máquina. Los registros y la memoria quedan con el último valor que recibieron.	
NOP	000	No ejecuta ninguna acción. La instrucción no tiene ningún efecto sobre el acumulador ni sobre la memoria.	Se incrementa en 1.

Ejemplos de programas MCBE

Los ejemplos siguientes se dan en la notación **posición / mnemotécnico o dato / argumento / contenido binario.**

Ejemplo 1. Leer un dato en la posición 4, sumarle el contenido de la posición 5 y escribirlo en la celda 6.

0	LD	4	01000100
1	ADD	5	10000101
2	ST	6	01100110
3	HLT		00100000
4	99		01100011
5	2		00000010

El efecto sobre la memoria de este programa será:

6	101		01100101

¿Qué pasaría si no estuviera la instrucción HLT de línea 3?

Ejemplo 2. Leer un dato del teclado, sumarle el contenido de la posición 5, restarle el contenido de la posición 6 y escribir el resultado por pantalla.

0	LD	30	01011110
1	ADD	5	10000101
2	SUB	6	10100110
3	ST	31	01111111
4	HLT		00100000
5	18		00010010
6	3		00000011

Ejemplo 3. Leer dos datos del teclado y escribir su suma por pantalla.

0	LD	30	01011110
1	ST	6	11100110
2	LD	30	01011110
3	ADD	6	10000110
4	ST	31	01111111
5	HLT		00100000
6	0		00000000

Ejemplo 4. Implementar la función y = 3x-2.

0	LD	30	01011110
1	ST	7	10000111
2	ADD	7	10000111
3	ADD	7	10000111
4	SUB	8	10101000
5	ST	31	01111111
6	HLT		00100000
7	0		00000000
8	2		00000010

Ejemplo 5. Leer un dato del teclado, restarle cuatro veces el contenido de la posición 7, y escribir el resultado por pantalla.

0	LD	30	01011110
1	SUB	7	10000111
2	SUB	7	10000111
3	SUB	7	10000111
4	SUB	7	10000111
5	ST	31	01111111
6	HLT		00100000
7	18		00010010

Ejemplo 5. Imprimir 10 veces el dato situado en la posición 9.

0	LD	11	01001011
1	JZ	7	11100111
2	LD	9	01001001
3	ST	31	01111111
4	LD	11	01001011
5	SUB	10	10101010
6	ST	11	01101011
7	JMP	-7	11010111
8	HLT		00100000
9	2		00000010
10	1		00000001
11	10		00001010

Ejemplo 6. Leer un dato del teclado, restarle seis veces el contenido de la posición 16, y escribir el

resultado por pantalla. Objetivo similar a un ejemplo anterior, pero diferente programa. La ventaja de este programa es que la operación de resta se puede hacer una cantidad cualquiera de veces, con sólo modificar el valor de la posición 14.

0	LD	30	01011110
1	ST	13	01101101
2	LD	14	01001110
3	JZ	7	11100111
4	SUB	15	10101111
5	ST	14	01101110
6	LD	13	01001101
7	SUB	16	10110000
8	ST	13	01101101
9	JMP	2	11000010
10	LD	13	01001101
11	ST	31	01111111
12	HLT		00100000
13	0		00000000
14	6		00000110
15	1		00000001
16	7		00000111

Ejemplo 7. ¿Qué hace este programa? ¿Qué problema presenta?

0	LD	30	01011110
2	ADD	6	10000110
3	ST	31	01111111
4	JMP	-2	11010010
5	HLT		00100000
6	1		00000001