

Introducción a la Computación

Contenidos

| | |
|---|----|
| Introducción a la Computación..... | 1 |
| Programa de la materia..... | 3 |
| Unidad I: La información..... | 3 |
| Unidad II: Arquitectura y organización de computadoras..... | 3 |
| Unidad III: Sistemas operativos..... | 3 |
| Unidad IV: Transmisión de datos..... | 4 |
| Unidad I: La Información..... | 5 |
| Datos e información..... | 5 |
| Procesamiento de la información..... | 6 |
| Tratamiento automático de la información..... | 6 |
| Informática y Computación..... | 7 |
| La información como objeto matemático..... | 8 |
| Arquitectura de computadoras..... | 9 |
| Sistemas de numeración..... | 11 |
| Sistemas posicionales..... | 11 |
| Sistema binario..... | 15 |
| Sistema hexadecimal..... | 16 |
| Conversión de decimal a otras bases..... | 16 |
| Conversión decimal a binario..... | 17 |
| Conversión de decimal a hexadecimal..... | 18 |
| Conversión de otras bases a sistema decimal..... | 19 |
| Representación digital | 19 |
| Números negativos..... | 19 |
| Números reales..... | 19 |
| Unidad II: Arquitectura y Organización de Computadoras..... | 21 |
| Modelo Computacional Binario Elemental..... | 21 |
| Unidades funcionales del MCBE..... | 21 |
| Descripción detallada del MCBE..... | 22 |
| Ciclo de instrucción..... | 23 |
| Detalles operativos del MCBE..... | 24 |
| Diagrama estructural del MCBE..... | 25 |
| Conjunto de instrucciones..... | 25 |

Programa de la materia

Unidad I: La información

1. Concepto de Información
2. Procesamiento de la información: evolución histórica
3. Sistemas de numeración:
 - 3.1. Sistemas posicionales y no posicionales
 - 3.2. Sistema decimal
 - 3.3. Sistema binario
 - 3.4. Sistema hexadecimal
4. Unidades de información
 - 4.1. bits, bytes, nibbles, palabras
 - 4.2. Múltiplos y submúltiplos

Unidad II: Arquitectura y organización de computadoras

1. Modelo computacional binario elemental
 - 1.1. Memoria
 - 1.2. CPU
 - 1.3. Conjunto de instrucciones
 - 1.4. Ciclo de instrucción
2. Representación e interpretación de la información
 - 2.1. Instrucciones, datos, números, caracteres, ASCII, multimedia, etc.
3. Programa almacenado
4. Lenguajes
 - 4.1. Lenguaje de máquina
 - 4.2. Lenguaje de alto nivel
 - 4.3. Interpretación
 - 4.4. Compilación

Unidad III: Sistemas operativos

1. Modelo de Sistema de Computación
 - 1.1. Usuarios
 - 1.2. Programas de aplicación
 - 1.3. Sistema operativo
 - 1.4. Hardware
2. Definición de sistema operativo
 - 2.1. Por lo que es
 - 2.2. Por lo que hace
3. Componentes del SO
 - 3.1. Kernel
 - 3.2. Shell
 - 3.3. Otros componentes
 - 3.4. Procesos y recursos
4. Funciones de un SO

- 4.1. Planificación de procesos
- 4.2. Planificación de recursos (memoria, sistema de archivos, almacenamiento secundario)
- 4.3. Protección y control
- 4.4. Contabilidad
- 5. Evolución histórica de los SO
 - 5.1. Desde máquina pelada hasta sistema embebido
 - 5.2. Monoprogramación y multiprogramación
 - 5.3. Batch e interactivo
- 6. Servicios de un SO
 - 6.1. Ejecución de programas
 - 6.2. Operaciones de E/S
 - 6.3. Manejo del file system
 - 6.4. Detección de errores
 - 6.5. Comunicaciones
 - 6.6. System calls, traps

Unidad IV: Transmisión de datos

Unidad I: La Información

No es ningún secreto que la vida moderna nos da muchas herramientas para hacer las cosas más rápidas, o más fáciles; o para hacer cosas que hace tiempo no se hubieran creído posibles. A través de Internet podemos manejar nuestras cuentas bancarias, consultar los diarios, mirar videos, estudiar, comunicarnos con nuestros amigos y miles de otras cosas. Sacamos fotos con celulares o con cámaras digitales, nos comunicamos con teléfonos móviles, leemos libros electrónicos, jugamos o escuchamos música con dispositivos portátiles. Hasta los lavarropas tienen una cierta inteligencia para hacer las cosas.

En todos estos casos, hay algo en común: en todas estas situaciones, existe esencialmente algún **dispositivo** que **procesa información**.

Diariamente utilizamos información en casi todo lo que hacemos. La necesitamos para poder desarrollar nuestra vida diaria. En esta materia nos vamos a ocupar de conocer, entre otras cosas, qué es la información, cómo es que se procesa la información por medios automáticos y cómo están contruidos los dispositivos capaces de hacerlo.

Datos e información

Cuando Alicia dice "19/3/95", nos está comunicando un **dato**. Cuando además nos dice que ese dato es su fecha de nacimiento, nos está dando **información**. La información se compone de datos, pero puestos en contexto de manera que sean relevantes, es decir, importantes, y modifiquen de alguna manera nuestra visión del mundo.

Esta información puede ser producida, almacenada, recuperada, comunicada, procesada, de varias maneras.

- La información se **produce** a raíz de algún **evento** importante, a partir del cual podemos **tomar alguna decisión**.
 - Evento: las tostadas saltan de la tostadora. Información: las tostadas están listas. Decisión: sacarlas de la tostadora y comerlas.
 - Evento: el semáforo cambia a verde. Información: me toca el paso. Decisión: poner primera y avanzar.
 - Evento: Alicia nos dice "hoy cumplo 18 años". Información: es el cumpleaños de Alicia. Decisión: felicitarla, y quizás hacerle un regalo.
- Cuando la información se **almacena**, es porque queda escrita o registrada de alguna forma en un **soporte** o medio de almacenamiento. Puede tratarse de un papel escrito con la lista de las compras, una pantalla que muestra horarios de llegada de aviones, o un pendrive con canciones.
- La información puede ser **comunicada**. Comunicamos información al recitarle la lista de las compras al almacenero; al decirle por teléfono a alguien a qué hora vamos a llegar mientras miramos la pantalla de los horarios; al escuchar las canciones almacenadas en el pendrive.
- A veces, para poder comunicar información hace falta **copiarla** de un soporte a otro. Ocurre esto al pasar en limpio la lista de compras en otro papel; o al ingresarla en la computadora, o al sacarle una foto.

¿En qué otras situaciones se produce información? Describa situaciones de la vida diaria donde la información se almacene. ¿Qué ejemplos de soportes de información conoce? ¿Se puede perder la información? ¿Se puede destruir un soporte de información y sin embargo conservarse la información? ¿Puede existir información sin un soporte? ¿Qué otras situaciones de comunicación de

información puede describir?

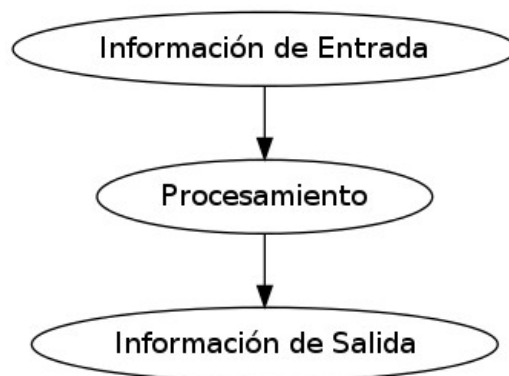
Procesamiento de la información

Cuando decimos que la información es **procesada**, queremos decir que hay alguien o algo que modifica esa información, presentándola de otra manera o combinándola con alguna otra pieza de información. Ese alguien o algo es un **agente** (agente quiere decir "que realiza una acción") de procesamiento de información. Todos procesamos información continuamente, durante todo el día, para varios fines.

El procedimiento exacto que se necesita para modificar la información depende de cada problema, de la forma como está representada la información, y de la forma como se desea obtener al final. Para lograr esa transformación se requiere utilizar la información tal como está representada, y operar con ella para crear una representación diferente.

Ese procedimiento necesario, en teoría, puede ser llevado a cabo por una o varias personas. Es lo que hacen, por ejemplo, los docentes en los colegios cuando toman la planilla de notas y calculan la nota final para cada alumno; los empleados contables, cuando toman el registro de asistencias e inasistencias de un empleado junto con su legajo y calculan el monto de lo que debe; el médico, cuando recibe el análisis de sangre de un paciente y combina esos datos con sus propias observaciones del paciente para formular un diagnóstico; el sistema de navegación automática de un avión, cuando analiza los datos de su posición y velocidad, y calcula la corrección necesaria para mantener el rumbo.

Un agente de procesamiento de información, primero que nada, debe **recuperar** esta información, es decir, leerla del soporte donde está almacenada. La información de la cual se parte para resolver un problema se llama información de entrada, o simplemente **entrada**; y la que produce el agente de procesamiento, información de salida, o simplemente **salida**.



Tratamiento automático de la información

La información es tan importante para nuestras actividades, que resulta interesante comprender y aplicar las formas de procesarla en forma **automática**, es decir, usando máquinas. Las ventajas son, por supuesto, que es posible construir máquinas que ejecuten este procesamiento sin cansarse, sin equivocarse, y en muchísimo menos tiempo. Y como consecuencia de todo esto, a mucho menor costo.

En muchos casos, el procesamiento de información podría hacerse a mano. Lamentablemente, algunos casos de procesamiento de información son tan complicados, o llevan tanto trabajo, que a mano resultan impracticables. En caso de realizarlos a mano, con lápiz y papel, el resultado tardaría tanto en estar disponible que directamente no serviría para nada.

¿De dónde sale la información que nos da la señorita del pronóstico del tiempo por la televisión?
¿Cómo se computa esta información? ¿Cuánto tiempo le llevaría a una persona realizar esos cálculos? ¿Y a más de una persona?

En realidad, el procesamiento de información viene de muy antiguo. Todos los pueblos que han desarrollado sistemas de escritura, con alfabetos u otros sistemas de signos, han sido capaces de almacenar información. Inclusive podemos decir que es antiguo el tratamiento de la información por medios más o menos automáticos, y más o menos mecánicos. Los *quipus*¹ de los incas son una forma de almacenamiento de información. Los pueblos antiguos diseñaron máquinas, a veces sumamente complicadas, para ayudarse en los cálculos del diseño de armas², o para completar cálculos astronómicos³.

—¿Y qué es una raíz cúbica, en nombre de todos los dioses?—

3 http://es.wikipedia.org/wiki/Mecanismo_de_Anticitera

preguntó.

Arquímedes lo observó, demasiado asombrado para poder hablar.

«La solución al problema délico —pensó—, la piedra angular de la arquitectura, el secreto de la dimensión, la diversión de los dioses.» ¿Cómo era posible que alguien que fabricaba catapultas no supiese lo que era una raíz cúbica?

Eudaimon lo miró con desagrado. Luego arrugó el papiro con furia, simuló limpiarse el trasero con él y lo arrojó al suelo.

Fragmento de *El Contador de Arena*, de Gillian Bradshaw.

Lo que nos diferencia de estos pueblos antiguos es que con la tecnología moderna podemos tratar cantidades muchísimo más grandes de información, en tiempos muchísimo más reducidos, y con formas de procesamiento muchísimo más complicadas.

Si bien el tratamiento de la información tiene una existencia independiente de la computación, hoy prácticamente toda la información se trata mediante **computadoras**. El Diccionario de la Lengua de la Real Academia Española dice que el término **Informática** viene del francés y designa el "conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores".

Sin embargo, hay que tener presente que la información y su tratamiento presentan muchos problemas interesantes para la ciencia aun sin necesidad de pensar en utilizar computadoras, o aunque las computadoras nunca se hubieran inventado. Es decir: **Informática** y **Computación** son dos disciplinas estrechamente relacionadas, pero diferentes.

La información como objeto matemático

No hay una única definición de información, ya que cada ciencia o disciplina utiliza el concepto en forma diferente. Un epistemólogo define la información como "una diferencia que importa"⁴. Wikipedia dice que la información es "una secuencia de símbolos que puede ser interpretada como un mensaje"⁵. Si preguntamos a una persona cualquiera qué significa para ella información, es posible que lo primero que se le venga a la cabeza sean las noticias del diario o de la TV.

Aunque todo lo anterior resulta correcto en su contexto, nosotros hablaremos de información en un sentido más preciso. Esto implica considerar la información desde un punto de vista matemático.

Siempre que los matemáticos consideran cosas del mundo real para trabajar, tratan de quedarse con lo esencial de esas cosas, aquello que puede ser reducido a cantidades y propiedades esenciales. Este procedimiento se llama abstracción. Cuando pensamos en la información en forma abstracta, no nos importa cómo está construido el mensaje (la **sintaxis**) ni lo que quiere decir (la **semántica**). Al abstraer datos, símbolos y mensajes, dejamos afuera su forma y su significado, sacando todo lo que sobra, y nos queda... *la información*. Ahora esta información puede ser medida y reducida a unidades.

Para ejemplificar, volvamos al episodio de la tostadora que eyecta sus tostadas listas. Ésta era una situación de producción de información. Si comenzamos a abstraer propiedades no esenciales de esa situación, ¿qué nos queda? Para el matemático no es importante que se trate de una tostadora, ni de tostadas, ni que el evento sea exactamente la eyección de una tostada. Lo importante es que *antes no había ocurrido el evento, y ahora sí*. En el caso del semáforo, no es importante que se trate de un semáforo, ni la altura a la cual está puesto, ni siquiera el color. Lo importante es que se ha producido el evento. Antes no había ocurrido el evento, y ahora sí. Esta diferencia en el tiempo es lo que genera la información, no la tostadora, ni la tostada, ni el semáforo.

4 http://es.wikipedia.org/wiki/Gregory_Bateson

5 <http://es.wikipedia.org/wiki/Informaci%C3%B3n>

Si alguien no pudiera ver la tostadora, o el semáforo, y necesitara tomar una decisión, podría preguntar a alguien más si el evento ya ha ocurrido, y ese alguien más podría responderle "**sí**" o "**no**". Esa persona le estaría comunicando la información necesaria para tomar la decisión. Ésta es la "diferencia que importa" y es la mínima unidad de información que se puede comunicar.

Las preguntas "de sí o no" se llaman preguntas **binarias**, porque su respuesta tiene dos posibilidades. Esa respuesta permite tomar una decisión entre dos posibles acciones (retirar la tostada/no retirar la tostada; avanzar/no avanzar). No hay una decisión más simple, ¡porque no se puede tomar una decisión entre **menos** posibilidades!

Al haber identificado esa unidad mínima de información, ahora podemos medir la "cantidad de información" de cualquier pieza de información. La cantidad de información que es aportada por la respuesta a una pregunta binaria recibe el nombre de **bit**. Un bit es, en definitiva, una pieza de información que puede tomar uno de dos valores. Esos dos valores suelen representarse con **0** y **1**, que como veremos más adelante, son los dos dígitos del **sistema de numeración binario**. El dígito binario, o, abreviadamente, "bit", es la **mínima unidad de información** en Informática, y será la que utilizaremos.

Si quisiéramos diseñar un artefacto de comunicación de información muy simple, podríamos hacerlo con una luz eléctrica provista de un interruptor. Cuando la persona interesada en comer la tostada viera la luz encendida, recibiría la información de que la tostada está lista. Nuestro dispositivo de comunicación habría acabado de comunicar **un bit de información**.

En una escena de una película muy popular de no hace mucho tiempo, hay un dispositivo similar para comunicar el inicio de una guerra... ¿Recuerdan la escena? ¿De cuál película estamos hablando?



¿Cómo se analiza esta escena en términos de información? ¿Qué operaciones tienen lugar sobre la información durante toda la secuencia? ¿Cuánta información se transmite? ¿Quiénes son los agentes de procesamiento?

Arquitectura de computadoras

¿Por qué las computadoras son capaces de procesar tantas clases diferentes de información? Los circuitos de las computadoras electrónicas son más fáciles de diseñar y construir, más económicos, y más útiles, cuando funcionan en base a sólo dos clases de señales o **estados**. Los microscópicos circuitos de las computadoras digitales modernas están en uno de dos posibles estados: activo o inactivo. Por este motivo se llaman **biestables**. Un elemento biestable de una computadora está activo cuando por él circula corriente. En esta condición, o estado, el biestable representa un 1. Cuando el mismo elemento está inactivo, representa un 0. De esta forma, las computadoras son dispositivos construidos de modo de poder manipular bits. Para poder ser almacenada, recuperada,

procesada o comunicada por una computadora, la información debe estar representada por bits.

Al ser capaces de tratar con bits y bytes, las computadoras pueden procesar cualquier clase de información, porque cualquier pieza de información puede descomponerse en partes y ser representada por bits. A la hora de representar datos de la vida real, para tratarlos por medios automáticos (es decir, al momento de procesar información con computadoras), esos datos pueden ser traducidos a bits de muchas maneras. Una forma natural de representar los números enteros, por ejemplo, es utilizar el sistema binario de numeración. Esta representación se traduce a bits de manera inmediata: un 0 o 1 binario será un bit con valor respectivamente 0 o 1 en la circuitería de la computadora.

Veremos más sobre la importancia de los bits cuando hablemos de la arquitectura de las computadoras.

Sistemas de numeración

Para representar números, habitualmente utilizamos el sistema de numeración decimal, que tiene diez **dígitos** (llamados así porque representan a los diez dedos con los que contamos). Como todos sabemos, éstos son el 0, el 1, el 2, el 3, el 4, el 5, el 6, el 7, el 8 y el 9.

¿Cómo podemos contar objetos en cantidades mayores que diez, con solamente diez dígitos? El truco es crear un sistema **posicional**, donde los dígitos tienen diferente valor según en qué lugar del número se encuentran.

Contando con el cero y los naturales

Asignamos dígitos a las cosas que queremos contar, y cuando se agota la secuencia de los dígitos, agregamos un nuevo dígito **1** a la izquierda para representar el hecho de que se nos acabó **una vez** la secuencia, y volvemos a empezar:

0... Cero...

1... uno...

... (varios dígitos...)

7... siete...

8... ocho...

9... nueve... ¡se acabaron los dígitos!

No importa, tomemos un nuevo dígito para la posición siguiente a la izquierda...

1 0... y repetimos la misma secuencia de diez dígitos a la derecha volviendo a empezar... diez...

1 1... once...

1 2... doce...

... y seguimos...

1 8... dieciocho...

1 9... diecinueve... ¡se acabaron **2 veces** los dígitos!

Tomemos el siguiente dígito en la secuencia para la posición de la izquierda...

2 0... y repetimos la secuencia a la derecha de nuevo...

y sigamos contando... **2** 1... veintiuno...

2 2... veintidós...

¿Qué pasa cuando se agota la secuencia en la posición de la izquierda? Es decir, ¿qué pasa cuando llegamos al **99**? ¿Y al **999**? Tenemos que tomar todavía una posición más a la izquierda y volver a empezar la secuencia con todas las posiciones de la derecha (luego del **99** escribimos **100**). Esto ocurre en una posición diferente, cada vez más a la izquierda, cada vez que agotamos la secuencia.

Como vemos, este procedimiento puede seguir indefinidamente, permitiendo escribir números de cualquier cantidad de dígitos.

Sistemas posicionales

Nuestro sistema es posicional: cuando escribimos un número, el **valor absoluto** de cada dígito será

siempre el mismo, pero su significado o **valor relativo** depende de la posición donde se encuentra. El **2** es un dígito cuyo valor absoluto es, claro, **2**. Pero el dígito **2** de la línea donde contamos **veintiuno (21)** no tiene el mismo valor relativo que el **2** de la línea **doce (12)**. Los símbolos que se encuentran más a la izquierda tienen mayor valor relativo: el **2** de **veintiuno** vale **veinte**, o sea **diez veces más** que el **2** de **doce**, porque representa el hecho de que la secuencia de **diez** dígitos se agotó dos veces.

En el número **21**, el **2** está desplazado a la izquierda una posición; por lo tanto, su valor se multiplica por **10**, valiendo **20**. En el número **215**, el **2** está desplazado a la izquierda dos posiciones; por lo tanto su valor se multiplica **dos veces** por **10** (o sea, $10 * 10 = 10^2 = 100$), dando **200**.

La cantidad de dígitos de un sistema numérico se llama la **base** del sistema. En cualquier sistema posicional, cada vez que un dígito se desplaza a la izquierda una posición, para obtener su valor relativo hay que multiplicarlo por una potencia de la base del sistema (cualquiera sea dicha base). En este caso, la base es **10**, porque estamos utilizando el sistema decimal. Cuando escribimos **215**, en realidad estamos expresando su desarrollo como suma de potencias de la base:

$$2 * 10^2 + 1 * 10^1 + 5 * 10^0.$$

Cuando vemos el “número” **215**, lo que estamos viendo en realidad son los coeficientes de las potencias de la base en el desarrollo del número **215**.

Símbolo y significado de los dígitos

Ya que estamos, tratemos de aclarar lo siguiente, que es un poco difícil de explicar: los símbolos que escribimos, y que llamamos “números”, no son más que una representación de los verdaderos **números**. ¿Se entiende esto? A ver con ejemplos:

- La figura siguiente no es música.



Es solamente una notación para los sonidos musicales. La verdadera música aparece cuando alguien toca esta notación en un instrumento y nosotros la escuchamos, o la imaginamos en nuestra cabeza.

- Éstas no son palabras: *CASA*, *SOL*, *PERRO*. Son una notación para las palabras. Las verdaderas palabras aparecen cuando alguien las dice, o cuando su significado está en nuestra cabeza.
- ¡Ésta no es una pipa!



Es una pintura que representa una pipa.

- De la misma manera, **12**, **21** y **3.1416** ¡no son los verdaderos números, sino una notación para los números! Y esta notación se hace eligiendo una base, y esta base puede ser cualquiera.








Así que podríamos elegir cualquier otra representación que quisiéramos para los números (igual que para las palabras, o para la música). Y como veremos, el mismo **número** podrá tener dos, o varias, **representaciones**.

Un sistema “frutal”

¿Cómo sería nuestro sistema de numeración si tuviéramos otra cantidad de dedos en nuestras manos? Depende de cuál fuera esa otra cantidad de dedos, pero muy probablemente seguiría siendo un sistema posicional. Y los dígitos, ¿necesariamente tienen que seguir siendo como los conocemos?

Imaginemos un sistema numérico con sólo dos símbolos, **naranja** y **banana**, sabiendo que **naranja** equivale a nuestro **0** y **banana** equivale a nuestro **1**. ¿Podremos usar estos símbolos para contar objetos?

Escribamos una secuencia de números para contar en nuestro sistema *frutal* y anotemos la correspondencia con nuestro sistema habitual.

-  **0**
-  **1** Se acabó la secuencia...
-  **2** Entonces agregamos un 1 a la izquierda y volvemos a iniciar la secuencia.
-  **3** ¡Se volvió a agotar la secuencia!
-  **4** Tendríamos que cambiar el dígito de la izquierda por el siguiente en la secuencia... pero no hay más dígitos. Entonces tomamos una posición más a la izquierda. Agregamos un dígito **1** y volvemos a iniciar la secuencia en todas las demás posiciones.
-  **5** Se agotó la secuencia en la posición de más a la derecha, paso al siguiente dígito en la posición siguiente.
-  **6** Y vuelvo a empezar la secuencia.

Nuestra secuencia de símbolos ahora tiene sólo dos “dígitos”, en lugar de los diez del sistema habitual. Para contar hasta 6, los números con la secuencia de naranjas y bananas se forman exactamente como hicimos antes: mientras contamos objetos, escribimos los símbolos de la secuencia (pasos **0** y **1**). Cuando se agota la secuencia, tomamos el siguiente símbolo de la secuencia y lo colocamos a la izquierda, porque nuestro sistema es posicional. Siempre con ese símbolo a la izquierda, volvemos a repetir la secuencia de dos símbolos mientras seguimos contando (**2** y **3**). Se acabó de nuevo la secuencia, entonces tenemos que agregar un nuevo “dígito” a la izquierda (la banana que aparece a la izquierda, en el **4**).

Preguntas

- ¿Cómo se escriben los números hasta el 10 con este sistema?
- Así como el **2** de **12** vale **2** pero el **2** de **21** vale **20**, las bananas y las naranjas tienen diferente valor relativo según en qué posición se encuentran. ¿Cuánto valen la banana de 1, la de 2 y la de 4? ¿Y cada una de las dos bananas de 5?
- Un náufrago anota con marcas en una palmera de su isla los días que van pasando. Cada día hace una nueva rayita en la palmera. ¿Cuántos dígitos tiene el sistema numérico que utiliza? ¿Éste es un sistema posicional?
- ¿Qué valores tienen, en el sistema decimal, los dígitos **naranja**, **banana** y **uva**, de un sistema posicional de base 3? ¿Cómo se escriben los primeros diez números en este sistema?
- ¿Cuánto vale la suma **naranja + banana**? ¿Y **banana + banana**?
- Sabiendo que la base de un sistema numérico posicional es **n**, ¿cuál es el valor absoluto de su dígito más grande?

Sistema binario

El sistema decimal tiene base 10, pero el sistema *frutal* tiene base 2. En realidad, el sistema *frutal* no es otra cosa que el **sistema binario**, de dos dígitos, donde en lugar de **0** y **1** quisimos escribir **naranja** y **banana** para mostrar que lo importante de los dígitos no son los símbolos, sino su significado.

Escribamos algunos pocos números en el sistema binario.

| | | | | | | | | | | | |
|---------|---|---|----|----|-----|-----|-----|-----|------|------|------|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Binario | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 | 1010 |

Como vemos, los dos símbolos **0** y **1** del sistema **binario** son los mismos que el **0** y el **1**, símbolos que ya conocemos, del sistema **decimal**. Esto puede llevar a confusión: cuando escribo **101**, ¿me refiero a la cantidad de los dálmatas de la película, o estoy hablando del número **5**, sólo que escrito en sistema binario? Para evitar estos problemas conviene aclarar en qué base estamos escribiendo los números, indicándola con un número subscripto, así: **101₂**. Esto quiere decir que nos referimos al número 101 en base 2, que equivale al 5 en base 10 (o sea **101₂ = 5₁₀**). Si quisiéramos escribir el título de la película, sin ambigüedades, escribiríamos “Los **101₁₀** Dálmatas”.

Como ocurre con cualquier sistema posicional, cada número expresado en el sistema binario es en realidad el resultado de un desarrollo, o cuenta, donde utilizamos las potencias de la base para calcular el valor relativo de cada dígito. Así, por ejemplo,

$$101_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 4 + 0 + 1 = 5_{10}.$$

Sistema hexadecimal

Así como hemos desarrollado un sistema numérico posicional con sólo dos dígitos, también es posible crear uno con más dígitos que en el sistema decimal. En el sistema **hexadecimal** tenemos **16** símbolos. Los primeros 10 símbolos se copian de los del sistema decimal (y valen lo mismo). La base del sistema es **16**, ¡así que nos faltan 6 símbolos!. Pero, como habíamos visto, los símbolos pueden elegirse a gusto, así que para el sistema hexadecimal se toman las letras **A** a la **F** como “dígitos” que toman los valores entre 10 y 15.

| | | | | | | | | | | | | | | | | |
|-------------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Hexadecimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

El sistema hexadecimal es muy usado en computación porque aporta importantes ventajas: la expresión de los números será en general más corta, y es bastante más fácil convertir entre los sistemas binario y hexadecimal que entre binario y decimal.

Conversión de decimal a otras bases

El procedimiento general para convertir un número expresado en sistema **decimal** a otra base, es **dividir** sucesivamente el número a convertir, y los sucesivos **cocientes**, por la base deseada. La expresión final se forma tomando **el último cociente y la sucesión de los restos en orden inverso**.

Un esquema útil

Para exponer nuestros cálculos en el presente material nos resultará práctico construir un esquema como el siguiente:

| Número | Cociente (base) | Resto |
|--------|-----------------|-------|
| | | |

Los pasos para utilizar el esquema son:

1. En el primer renglón de la columna *Número* escribimos el número a convertir.
2. Dividimos *Número* por la base, llenando *Cociente* y *Resto*.
3. Copiamos el *Cociente* en la columna *Número*. Éste es nuestro nuevo *Número*.
4. Repetimos los pasos 2 y 3 hasta que ya no se pueda seguir dividiendo. Notemos tres cosas importantes:
 - Los *Restos*, por el algoritmo de división, deben necesariamente ser menores que el divisor (la base). De lo contrario, se podría seguir dividiendo.
 - Lo mismo ocurre con el último *Cociente*. De lo contrario, se podría seguir dividiendo.
 - Por lo dicho anteriormente, como son menores que la base, tanto los *Restos* como el último *Cociente* se pueden escribir con **un único dígito** en el sistema deseado.
5. Finalmente escribimos el último *Cociente* y los *Restos*, de abajo hacia arriba, en la base deseada, siguiendo las flechas. Ésta es la expresión que buscamos.

Ejemplo: Convertir **66** a base **3**.

| Número | Cociente (3) | Resto |
|--------|--------------|-------|
| 66 | 22 | 0 |
| 22 | 7 | 1 |
| 7 | 2 | 1 |

Finalmente, el número 66 expresado en base 3 se escribe $2110_{(3)}$.

Conversión decimal a binario

Como se explicó, el procedimiento es dividir sucesivamente el número a convertir por la base. La expresión en el sistema binario se forma tomando el último cociente y la sucesión de los restos en orden inverso. Todos estos números serán menores que la base y por lo tanto son dígitos 0 o 1. Convirtamos, por ejemplo, $531_{(10)}$ a sistema binario:

| Número | Cociente (2) | Resto |
|--------|--------------|-------|
| 531 | 265 | 1 |
| 265 | 132 | 1 |
| 132 | 66 | 0 |
| 66 | 33 | 0 |
| 33 | 16 | 1 |
| 16 | 8 | 0 |
| 8 | 4 | 0 |
| 4 | 2 | 0 |
| 2 | 1 | 0 |

Es decir, $531_{(10)} = 1000010011_{(2)}$.

Truco mnemotécnico

Un truco útil para convertir rápidamente números decimales (pequeños) al sistema binario es memorizar los valores de algunas potencias de 2 y utilizarlos en las cuentas. Por ejemplo:

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Entonces, si me preguntan cómo se escribe en el sistema binario, por ejemplo, el número $78_{(10)}$, sólo necesito ver de qué manera se descompone 78 como suma de estos valores. Hay una sola manera:

$$78_{(10)} = 64 + 8 + 4 + 2 = 2^6 + 2^3 + 2^2 + 2^1$$

Si completamos las potencias:

$$78_{(10)} = 64 + 8 + 4 + 2 = 2^6 + 2^3 + 2^2 + 2^1 = 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0.$$

Por lo tanto, cuando quiera escribir $78_{(10)}$ en binario, utilizaré los coeficientes de la expresión que acabamos de escribir: $78_{(10)} = 1001110_{(2)}$.

Comprobemos nuestro resultado con el esquema:

| Número | Cociente (2) | Resto |
|--------|--------------|-------|
| 78 | 39 | 0 |
| 39 | 19 | 1 |
| 19 | 9 | 1 |
| 9 | 4 | 1 |
| 4 | 2 | 0 |
| 2 | 1 | 0 |

O sea, $78_{(10)} = 1001110_{(2)}$.

Conversión de decimal a hexadecimal

Como con el sistema binario, el procedimiento es dividir sucesivamente el número a convertir por la base. La expresión en hexadecimal se forma tomando el último cociente y la sucesión de los restos en orden inverso⁶. Ahora tenemos que tener cuidado con cómo escribimos los sucesivos *Restos*: tienen que estar en el sistema hexadecimal.

Ejemplo

Convertir $531_{(10)}$ a hexadecimal:

| Número | Cociente (16) | Resto |
|--------|---------------|-------|
| 531 | 33 | 3 |
| 33 | 2 | 1 |

O sea, $531_{(10)} = 213_{(16)}$.

Ahora veamos el caso donde algún resultado intermedio es mayor que 10:

6 http://es.wikipedia.org/wiki/Sistema_hexadecimal

Convertir $7158_{(10)}$ a base 16:

| Número | Cociente (16) | Resto |
|--------|---------------|-------|
| 7158 | 447 | 6 |
| 447 | 27 | 15 |
| 27 | 1 | 11 |

Aquí hay que tener cuidado de expresar todos los restos en dígitos pertenecientes al sistema. Al construir la representación en hexadecimal de 7158, no puedo escribir los restos **15** u **11** (que están escritos en decimal) si no los expreso en hexadecimal. Se tiene que $11_{(10)} = B_{(16)}$ y que $15_{(10)} = F_{(16)}$. O sea, $7158_{(10)} = 1BF6_{(16)}$.

Hay más detalles sobre conversiones numéricas en varias páginas online⁷.

Conversión de otras bases a sistema decimal

Para convertir de otras bases al sistema decimal usamos lo que explicamos al hablar de la **base** de los sistemas numéricos posicionales y en el ejemplo del truco mnemotécnico anterior. Todo lo que hay que hacer es multiplicar los dígitos de la expresión en la base actual por potencias crecientes de la base, comenzando por el exponente 0. En el truco mnemotécnico vimos el desarrollo de un número decimal en base 2; leyendo el ejemplo al revés, tenemos el mecanismo para volver a convertir a sistema decimal. Si quisiéramos convertir un número en sistema hexadecimal a decimal, consideraremos las potencias de 16.

Ejemplo

Convertir $1C8A09_{(16)}$ a decimal.

$$\begin{aligned} 1C8A09_{(16)} &= \\ 1 * 16^5 + 12 * 16^4 + 8 * 16^3 + 10 * 16^2 + 0 * 16^1 + 9 * 16^0 &= \\ 1048576 + 12 * 65536 + 8 * 4096 + 10 * 256 + 0 + 9 * 1 &= \\ 1048576 + 786432 + 32768 + 2560 + 9 &= 1870345_{(10)} \end{aligned}$$

Representación digital

Con lo que hemos visto hasta ahora, hemos podido representar los números naturales y el 0. Pero existen otros conjuntos numéricos (como los racionales, o los enteros negativos, o los irracionales) que aparecen frecuentemente en los problemas de la Informática.

Cabe preguntarse cómo podemos representar otros conjuntos numéricos con las mismas herramientas, es decir, sólo con una cantidad finita de dígitos binarios.

Números negativos

Números reales

⁷ http://es.wikipedia.org/wiki/Sistema_binario#Conversi.C3.B3n_entre_binario_y_decimal

Unidad II: Arquitectura y Organización de Computadoras

Modelo Computacional Binario Elemental

En esta unidad describiremos la arquitectura de una computadora hipotética, sumamente elemental. La utilizaremos como ejemplo para explicar ciertas cuestiones básicas sobre el funcionamiento de las computadoras. Comparte las características esenciales de casi todas las computadoras digitales, y está inspirada en máquinas que han existido realmente.

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.

No se trata, entonces, más que de un **modelo**, y como utiliza aritmética binaria, lo hemos bautizado **MCBE (Modelo Computacional Binario Elemental)**. El MCBE es un ejemplo muy sencillo de **computador de programa almacenado**⁸. Nos servirá para mostrar muchos de los problemas relacionados con la arquitectura y la organización de las computadoras reales.

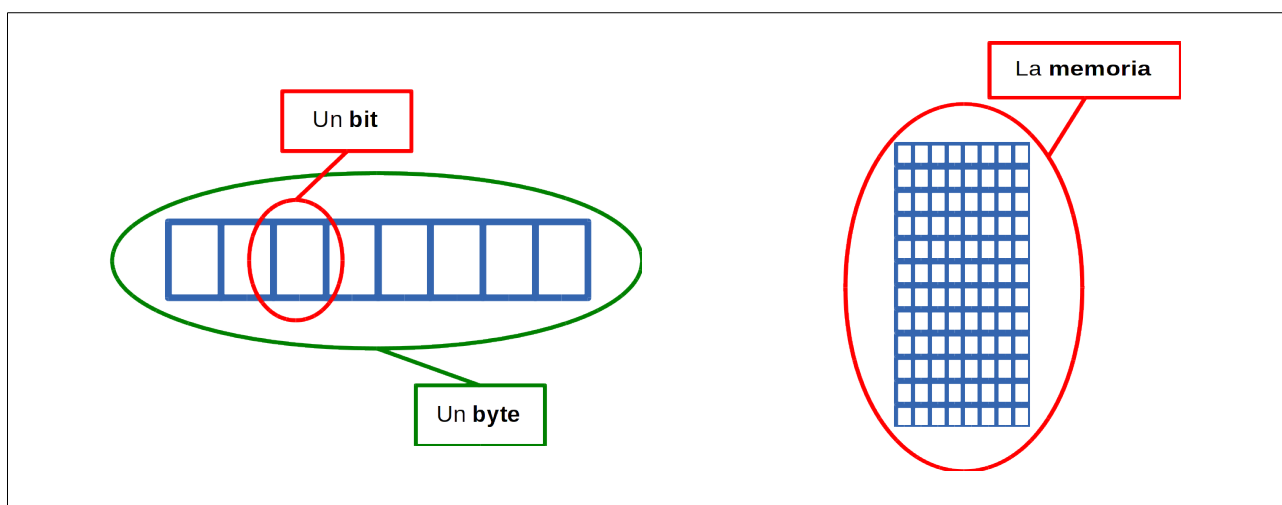
Recordemos, antes que nada, que las computadoras **no toman decisiones por sí solas**. Todo lo que hagan **está determinado por el programa almacenado**, y el diseño de este programa es responsabilidad del usuario. Entonces, es necesario que especifiquemos en nuestro diseño, con todo detalle, cuál será la respuesta de la computadora a cada alternativa de este programa.

Unidades funcionales del MCBE

Esta computadora elemental se compone de dos unidades funcionales principales: **memoria y CPU**.

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.



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 será capaz de guardar un determinado contenido. Como cada biestable representa un bit y cada posición de memoria representa un byte, a veces esos

⁸ http://es.wikipedia.org/wiki/Computador_de_programa_almacenado

circuitos y celdas de circuitos se llaman directamente **bits y bytes de la memoria**. La posición relativa de cada byte se llama su **dirección**. Para acceder a un dato contenido en una posición de memoria, para leerlo o para modificarlo, necesariamente tenemos que mencionar su dirección; al hacerlo así 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 tres unidades funcionales propias: la **Unidad de Control**, la **Unidad Lógico-aritmética**, y la **Unidad de Entrada/Salida**. Dos de estas unidades cuentan con **registros** especiales, que son espacios de almacenamiento, similares a los de la memoria, pero situados en otro lugar de la circuitería, y por lo tanto no tienen direcciones como tienen los bytes de la memoria.

- 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 UE/S conecta a la MCBE 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.

Descripción detallada del MCBE

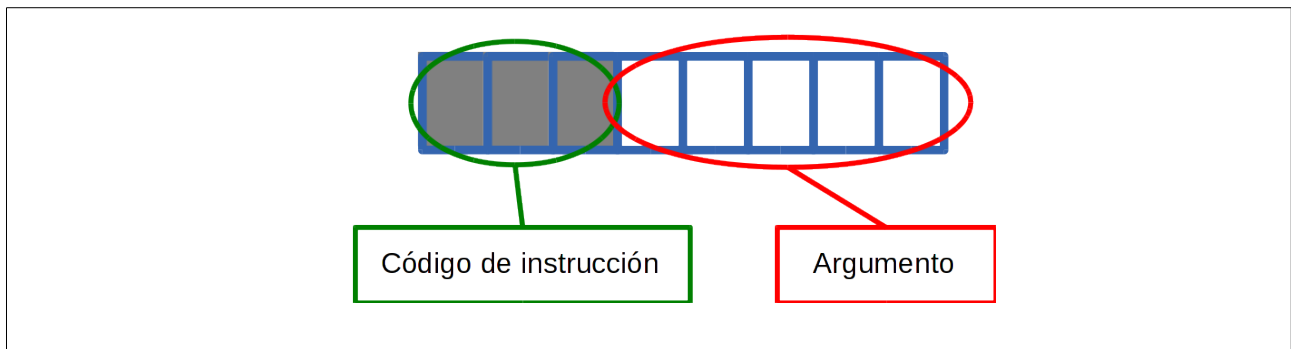
Instrucciones

Hay tan sólo **ocho diferentes instrucciones** que puede seguir esta máquina. Algunas sirven para realizar cálculos aritméticos; otras, para modificar el curso de las acciones a seguir por el programa. 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 le permitirá interpretar 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, el resto de los 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

Los argumentos pueden ser 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 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 **signo-magnitud**. 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 Programa⁹) e **IR** (por *Instruction Register*, Registro de Instrucciones¹⁰).
 - 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 posee un registro especial llamado **A** (por *Acumulador*).
 - 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.
- 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 un programa 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.
- Cada vez que un valor se copia del acumulador A a una posición de memoria B, el valor de A no cambia. 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 al acumulador.
- 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.

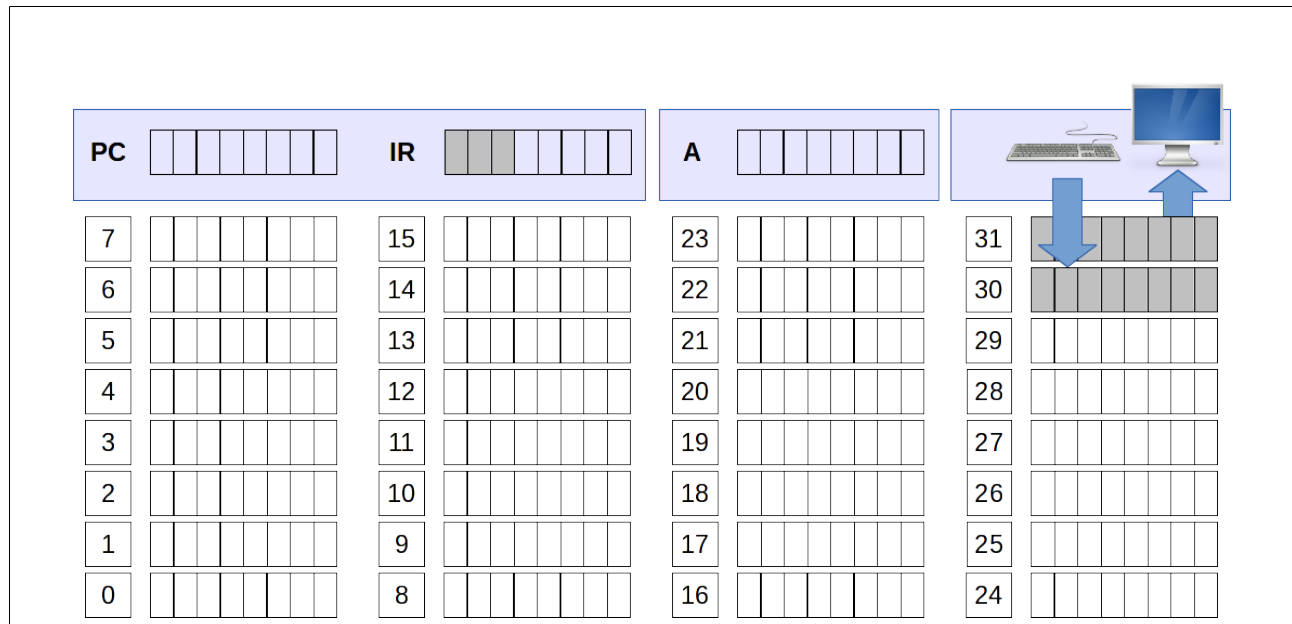
Preguntas

- ¿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?

⁹ http://es.wikipedia.org/wiki/Contador_de_programa

¹⁰ http://es.wikipedia.org/wiki/Registro_de_instrucci%C3%B3n

Diagrama estructural del MCBE



Conjunto de instrucciones

| <i>Instrucción</i> | <i>Cód.</i> | <i>Efecto sobre memoria y registros</i> | <i>Efecto sobre el PC</i> |
|-----------------------------|-------------|---|--|
| 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. | El desplazamiento será sumado al PC únicamente en caso de que el acumulador contenga un valor igual a 0. |
| 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 $3x-2$.

| | | | |
|---|-----|----|----------|
| 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 |