FICHA[1]

LA COMPUTADORA

Guía para docentes y familias

[1] Material extraído del Manual para la Enseñanza de las Ciencias de la Computación en el aula de la Iniciativa Program.AR. Claudia Banchoff Tzancoff; Vanessa Aybar Rosales; Silvina Justianovich; Vanina Klinkovich; Hernán Czemerinski (2019). Ciencias de la computación para el aula, 2do ciclo secundaria (1st ed.). Buenos Aires, Argentina: Fundación Sadosky.

Fundación SADOSKY <Program.AR/>

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿QUÉ SON LAS COMPUTADORAS?

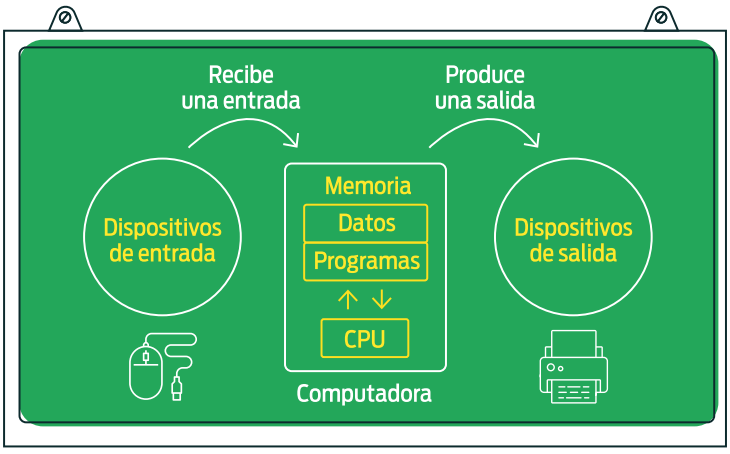
¿Qué caracteriza a las computadoras? ¿Las hay de diferentes formas y tamaños? ¿Qué cosas tienen en común todas ellas?

1. Completá la tabla con cinco computadoras que no sean ni las portátiles ni las de escritorio. Para cada una de ellas identificá, al menos, un dispositivo de entrada y uno de salida.

MÁQUINA QUE ES O QUE CONTIENE UNA COMPUTADORA	DISPOSITIVO DE ENTRADA	DISPOSITIVO DE SALIDA

ARQUITECTURA DE VON NEUMANN

En 1945, el matemático de origen austrohúngaro John von Neumann presentó un modelo conceptual de arquitectura de computadoras, cuyo diseño sigue vigente en las computadoras modernas. En este modelo, conocido como ***arquitectura de von Neumann***, una computadora está compuesta por una unidad central de procesamiento –que se encarga de ejecutar las instrucciones de los programas–, una memoria –en la que se almacenan los datos y los programas–, y dispositivos de entrada y salida que permiten el ingreso y egreso de datos.



NOMBRE Y APELLIDO:

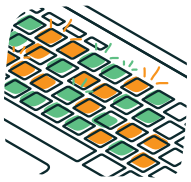
CURSO: FECHA:

2. Explorá el siguiente anuncio. Relacioná los elementos de la computadora publicitada con los componentes descritos en el modelo de von Neumann.

Bet&RobPro

La nueva computadora *Bet & Rob Pro* cuenta con un microprocesador Intel Core i7-7700HQ que maneja cargas de trabajo pesadas en minutos. Este y la RAM DDR2-667 de 32 Gigabytes le permitirán trabajar con programas complejos y múltiples pestañas, incluso ejecutando varias aplicaciones a la vez.

***TOUCHPAD* HIPERSENSIBLE**

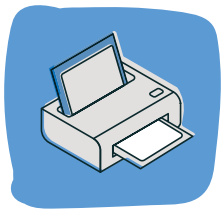


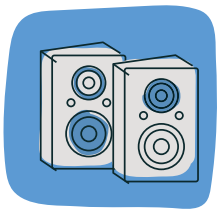
TECLADO LUMINOSO

LECTOR DE HUELLAS

Además, tiene *touchpad* hipersensible, teclado luminoso y lector de huellas dactilares.

Con la pantalla táctil *full HD* podrá comandar la máquina usando los diez dedos de la mano, y la cámara frontal de 8 megapíxeles le permitirá sacarse fotos cuando esté procrastinando.





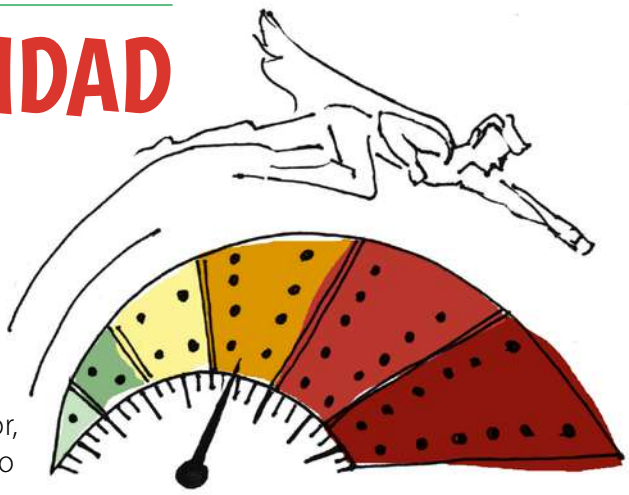
De regalo, unos parlantes holofónicos y una impresora láser color.

CPU	
MEMORIA	
DISPOSITIVOS DE ENTRADA	
DISPOSITIVOS DE SALIDA	

NOMBRE Y APELLIDO:

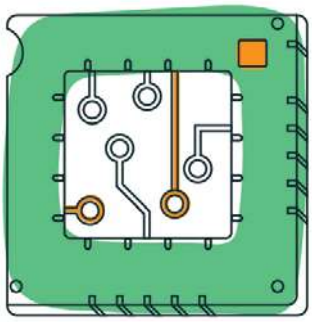
CURSO: FECHA:

ZOOM IN A LA UNIDAD CENTRAL DE PROCESAMIENTO



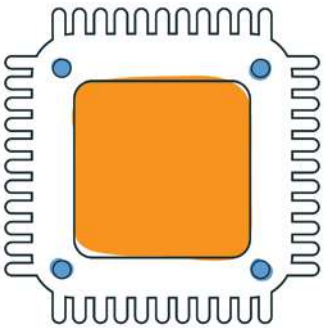
Cuándo miramos el anuncio de venta de un procesador, ¿qué quieren decir los datos que se mencionan? ¿Cómo inciden en el rendimiento de nuestras computadoras?

1. Mirá los anuncios publicitarios y completá las tres primeras filas de la tabla que se encuentra en la siguiente página.



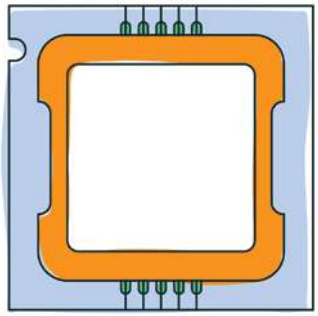
Intel® Core™ i7

· 6 núcleos 4.5 GHz
· Memoria compatible DDR3/DDR4.



AMD Ryzen 7 / 2700

La verdadera inteligencia, compuesta por 8 núcleos, una frecuencia de reloj de 4.1 GHz, compatible con memorias DDR4.



Qualcomm Snapdragon 835

Un 30% más fino y eficiente, para un presente de dobles cámaras y realidad virtual. Los 8 núcleos de 2.45 GHz se integran con memorias LPDDR4, lo que genera máxima eficiencia en dispositivos móviles.

NOMBRE Y APELLIDO:

CURSO: FECHA:

MARCA	MODELO	CANTIDAD DE NÚCLEOS	FRECUENCIA	COMPATIBILIDAD CON MEMORIA

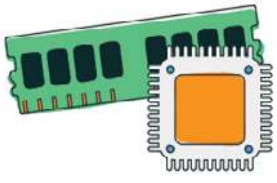
2. Buscá en Internet dos anuncios más y completá las últimas dos filas.

3. ¿Cuál o cuáles de los procesadores de la tabla tiene capacidad de ejecutar más instrucciones en forma simultánea? ¿Por qué, de qué depende?

4. ¿Qué indica la frecuencia de un procesador? ¿Tiene alguna incidencia en la velocidad a la que funciona una computadora?

COMPATIBILIDAD CON LA MEMORIA

Para poder funcionar juntos, el procesador y la memoria deben ser compatibles. Como sucede con casi cualquier componente electrónico, también entre las memorias existen diferentes generaciones tecnológicas. Un procesador, en general, admitirá memorias de una generación. Si querés ampliar la cantidad de memoria de tu computadora, asegurate de comprar una compatible.



NOMBRE Y APELLIDO:

CURSO: FECHA:

ANEXO

¿CUÁNTO CABE EN LA MEMORIA?

Cuando hablamos de memoria, nos referimos a una serie de componentes físicos –de *hardware*– que tienen la capacidad de representar información. Esto incluye tanto a la memoria RAM, como a los discos rígidos, los dispositivos portátiles de almacenamiento, etc.

¿CÓMO SE ALMACENA INFORMACIÓN EN UN DISPOSITIVO DE MEMORIA?

Una memoria se puede representar como una gran tira de celdas contiguas. En cada una se puede almacenar uno de dos valores: o bien un cero o bien un uno. Esta es la mínima unidad de información que una computadora puede representar y se llama ***bit***. En general, no se piensa en términos de bits, sino que se usan unidades de medida más grandes. Por ejemplo, un ***byte*** equivale a 8 bits. Como cada bit puede tener un cero o un uno, al considerar todas las posibles combinaciones de ellos agrupados de a 8, podemos ver que en un byte se pueden almacenar 256 valores distintos.

0	0	0	0	0	0	0	0	→	0
0	0	0	0	0	0	0	1	→	1
0	0	0	0	0	0	1	0	→	2
0	0	0	0	0	0	1	1	→	3
...									
1	1	1	1	1	1	1	0	→	254
1	1	1	1	1	1	1	1	→	255

BITS, BYTES Y LO DE MÁS ALLÁ

Así como para medir longitudes hay distintas unidades de medida, como por ejemplo milímetros, centímetros y metros, lo mismo sucede cuando queremos medir cantidades de información. Estas son algunas de ellas.

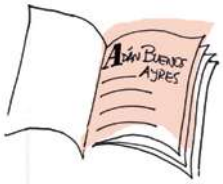
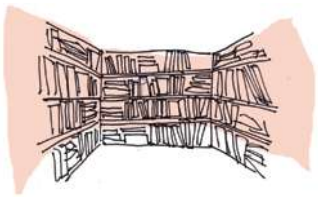
UNIDAD	Byte	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)
TAMAÑO	8 bits	1024 bytes	1024 KB	1024 MB	1024 GB

El uso del número 1024 como factor de multiplicación entre distintas unidades de medida se debe a que, por ser una potencia de 2 ($2^{10} = 1024$), es de fácil manipulación para una computadora.

NOMBRE Y APELLIDO:

CURSO: FECHA:

Para dimensionar cuánta información cabe en un dispositivo de memoria, mirá la siguiente analogía.

CANTIDAD DE MEMORIA	PUEDE CONTENER...	
1 byte		Una letra de un libro
1 KB		Una página de un libro
1 MB		Un libro de 1024 páginas
1 GB		Una habitación con 1024 libros
1 TB		Los libros contenidos en 6 edificios de 10 pisos y uno de 5 pisos con cuatro departamentos de cuatro habitaciones por piso

Esto entra en un disco de un Terabyte.

NOBLEZA OBLIGA

El espacio que se utiliza para codificar un carácter depende del sistema de codificación usado. Por ejemplo, algunas versiones de UNICODE llegan a usar cuatro bytes. Pero la codificación ASCII, que permite codificar el alfabeto latino, utiliza un solo byte para cada carácter. O sea que, si todos los libros están en castellano, ¡la analogía es precisa!

NOMBRE Y APELLIDO:

CURSO: FECHA:

LA CACHÉ

¿Alguna vez tu teléfono inteligente te sugirió "borrar la *caché*"? Todo muy lindo, pero… ¿qué es la *caché*?

1. Frecuentemente utilizamos aplicaciones de mapas en los teléfonos inteligentes. Cuando buscamos algún lugar, ¿cómo hace el teléfono para mostrarnos el mapa? ¿De dónde saca la información?

a. Deshabilitá de tu teléfono la conexión wifi y el acceso a la red de datos de telefonía celular. Después abrí la aplicación de mapas que uses habitualmente y buscá la dirección de tu casa. ¿Apareció el mapa en la pantalla? ¿Dónde están guardados esos mapas?

b. Ahora buscá una ciudad distante, que nunca hayas visitado. ¿Apareció el mapa en la pantalla? ¿Cuál es la diferencia de esta búsqueda con la del punto anterior? ¿Dónde está guardado el mapa de esta ciudad?

CACHING

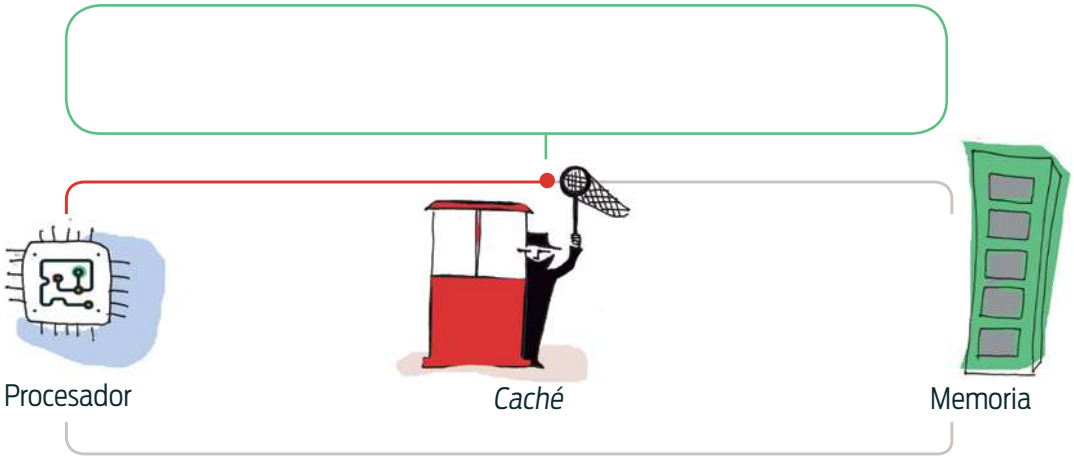
La idea de *caching* es que, entre el procesador y un dispositivo de memoria, haya otra memoria más pequeña a la que se pueda acceder a mucha mayor velocidad. En ella se conservan los datos más frecuentemente solicitados por los programas y, de esta manera, se reduce significativamente el tiempo que espera el procesador entre que pide esos datos y los obtiene. Algunos ejemplos son un disco para algo almacenado en la nube, la memoria RAM para algo almacenado en el disco, etc.

NOMBRE Y APELLIDO:

CURSO: FECHA:

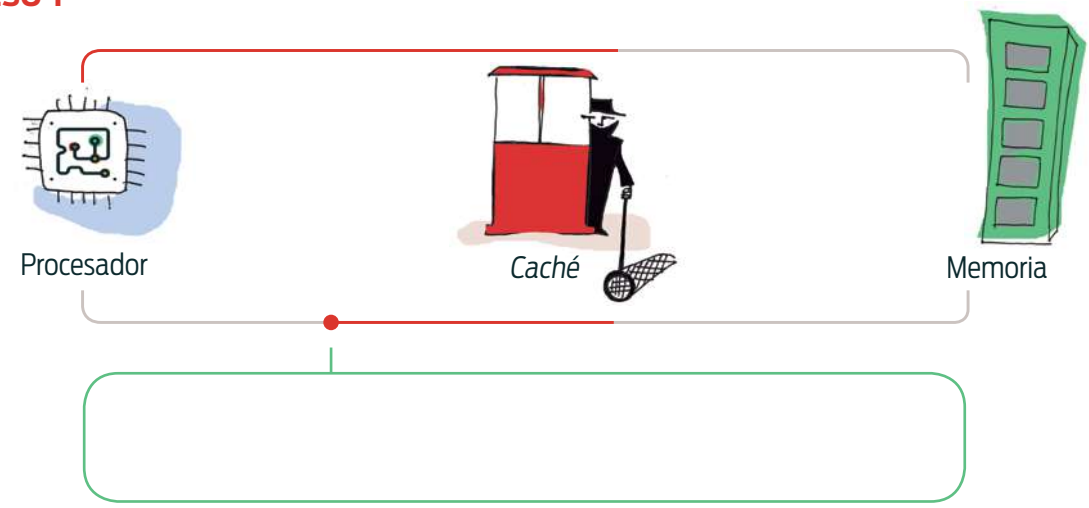
2. Cuando un programa solicita un dato, pueden suceder dos cosas: o bien el dato está en la *caché* o bien no lo está.

a. Completá en el siguiente esquema lo que va sucediendo desde que el procesador requiere un dato de la memoria hasta que lo obtiene.



En esta instancia pueden suceder dos cosas: o bien el dato está en la *caché* o bien no lo está.

Caso 1

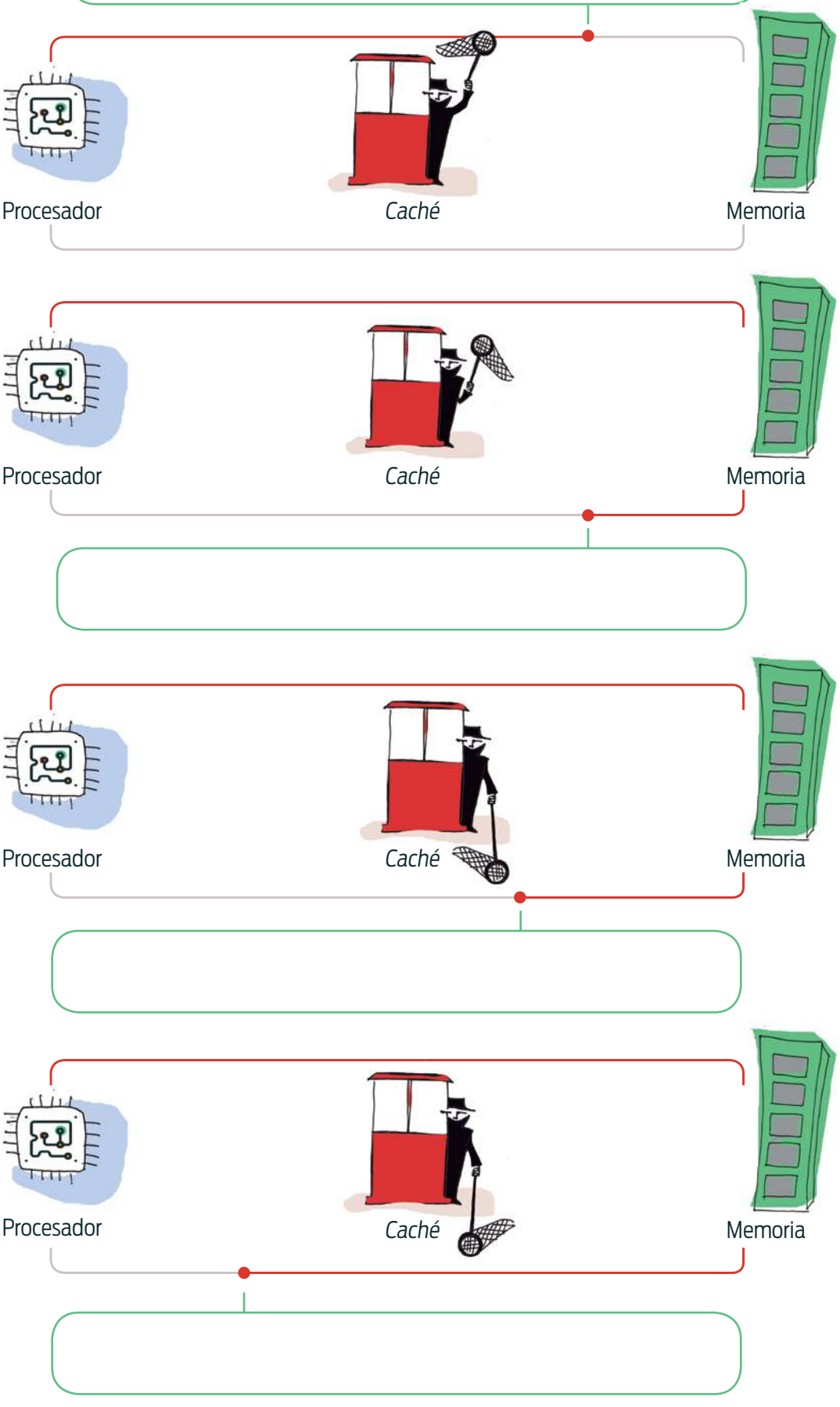


NOMBRE Y APELLIDO:

CURSO:

FECHA:

Caso 2



NOMBRE Y APELLIDO:

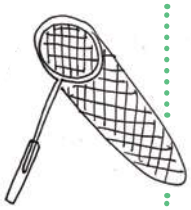
CURSO: FECHA:

b. Cuando un programa escribe un dato que es interceptado y almacenado en la *caché*, ¿debe también copiarse en la memoria? ¿Por qué?

¿LA PEGA O LA PIFIA?

Cuando un dato solicitado se encuentra en la *caché* se dice que la *caché la pegó* (o más comúnmente *caché hit*, por su forma en inglés). En cambio, cuando esto no sucede y hay que ir a buscarlo a la memoria, que la *caché la pifió* (o *caché miss*).

¿SABÍAS QUÉ...



…los procesadores tienen (muy) pequeños módulos de memoria internos llamados *registros* a los que puede acceder a altísima velocidad? En la actualidad, un procesador no suele tener más de 32, cada uno de 32 bytes. ¡En ellos se "cachea" información de la memoria RAM!

BAJAMOS HASTA EL LENGUAJE DE MÁQUINA

ANEXO

Cada modelo de computadora tiene componentes electrónicos que son capaces de llevar a cabo operaciones muy simples. Por ejemplo, operaciones aritméticas (como sumar y multiplicar), operaciones lógicas, leer un dato en la memoria, escribir un dato en la memoria, etc. Al programar, ¡por suerte no nos hace falta pensar en esto! En su lugar, usamos lenguajes que son mucho más expresivos, llamados ***lenguajes de alto nivel***, que nos permiten razonar en términos del problema que queremos resolver. Ahora bien, ¿cómo se ejecutan estos programas si la computadora hace tan pocas y rudimentarias operaciones?

COMPILADOR

Un **compilador** es un programa que toma como entrada un programa escrito en un lenguaje de programación de alto nivel y genera como salida otro programa en lenguaje ensamblador que es semánticamente equivalente; es decir, que hace exactamente lo mismo.

ENSAMBLADOR

Un programa en lenguaje ensamblador sigue siendo un archivo con texto. Para ejecutar un programa, lo primero que hay que hacer es cargarlo en la memoria, de donde el procesador irá leyendo sus instrucciones una por una. Sin embargo, en la memoria solo se guardan bits. El **ensamblador** es un programa que toma como entrada un programa en lenguaje ensamblador y genera como salida otro equivalente en **lenguaje de máquina**.

```
void saludarEstudiantes() {
        printf("¡Hola, estudiantes!");
}
```

COMPILADOR

```
movl $0xFF001122, %eax
addl %ecx, %edx
xorl %esi, %esi
push1 %ebx
movl 4(%esp), %ebx
leal (%eax, %ecx, 2), %esi
cmpl %eax, %ebx
jnae foo
retl
```

ENSAMBLADOR

1	0	1	1	1
0	0	0	1	0
1	1	1	1	1
1	1	0	0	1

LENGUAJE DE ALTO NIVEL

Habitualmente, al construir un programa se utiliza un **lenguaje de alto nivel**. Estos lenguajes permiten que un programador razone sobre el problema que quiere resolver, abstrayendo por completo el funcionamiento interno de los componentes de *hardware*.

LENGUAJE ENSAMBLADOR

Cada computadora tiene componentes que realizan algunas operaciones simples: aritméticas, lógicas, de lectura y escritura en la memoria, y algunas más. El **lenguaje ensamblador** de una computadora tiene instrucciones que se corresponden con lo que sus componentes de *hardware* son capaces de llevar a cabo.

LENGUAJE DE MÁQUINA

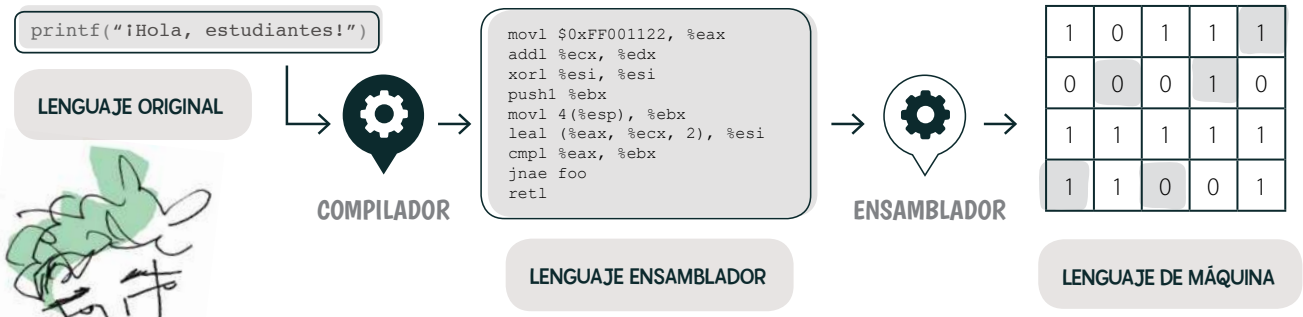
El **lenguaje de máquina** está formado por una serie de instrucciones codificadas con números binarios que pueden tanto cargarse en la memoria como ser interpretadas por el *hardware* de la computadora.

BIEN ADENTRO DE LA COMPUTADORA

¿Cómo hace una computadora para ejecutar cada instrucción de un programa? ¿Qué componentes internos intervienen? Metámonos bien adentro para ver qué es lo que pasa por allá abajo.

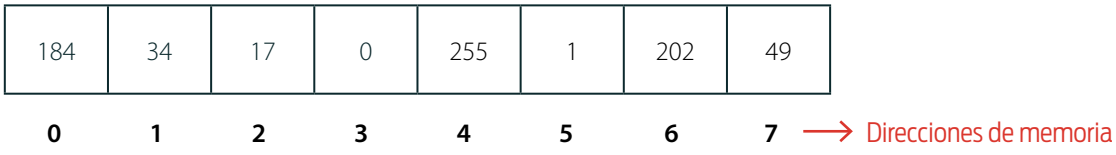
RECORDATORIO

Para que un programa pueda ejecutarse, en primer lugar se compila y se obtiene un programa semánticamente equivalente en lenguaje ensamblador, es decir, con instrucciones que se corresponden con las operaciones que los circuitos del *hardware* de la máquina pueden realizar; en segundo lugar, el programa se ensambla para obtener código de máquina –instrucciones codificadas en un sistema binario que pueden almacenarse en la memoria de la computadora–.



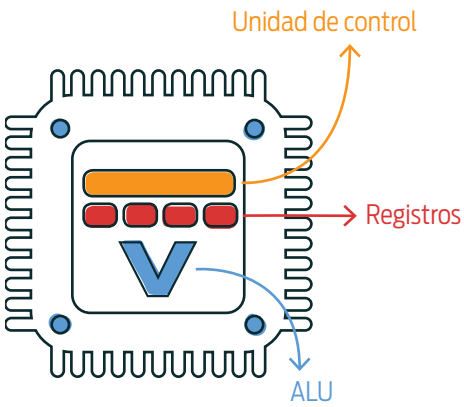
DIRECCIONES DE MEMORIA

En la memoria, los bits se agrupan de a 8 y, por lo tanto, se la concibe como una secuencia de bytes. Cada byte de la memoria puede identificarse con un número, que se corresponde con la ubicación en la que se encuentra, comenzando a contar desde 0. A este número se lo conoce como **dirección de memoria**.



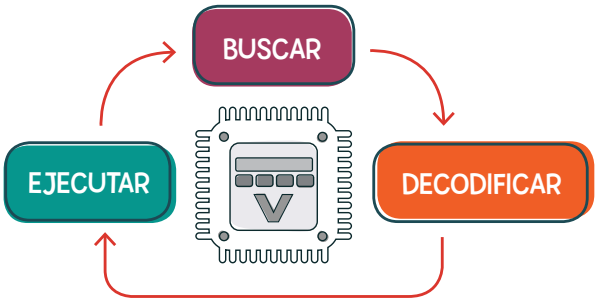
LA CPU POR DENTRO

La unidad central de procesamiento, internamente, tiene: *(i)* una unidad aritmética lógica (ALU), que cuenta con circuitos para realizar operaciones aritméticas y lógicas simples; *(ii)* registros, que son pequeñas unidades de memoria, en los que almacena datos; y *(iii)* una unidad de control, que dirige el proceso de ejecución de las instrucciones de un programa.



CICLO DE INSTRUCCIÓN

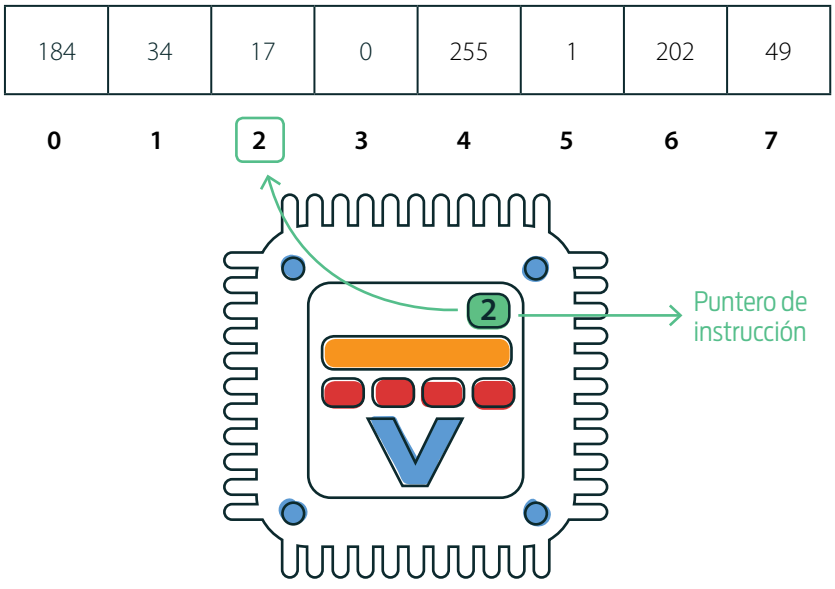
Un ciclo de instrucción comprende las tres etapas que lleva a cabo la unidad central de procesamiento para ejecutar una instrucción del lenguaje de máquina. La unidad de control se encarga de *(i)* **buscar** la instrucción en la memoria; *(ii)* **decodificar** la instrucción; y *(iii)* enviar las señales necesarias para que se pueda **ejecutar** la instrucción (dependiendo de cuál sea la instrucción, esto puede ser activar la ALU, cargar datos en registros, enviar señales a los circuitos que comunican la CPU con la memoria, etc.).



La unidad central de procesamiento repite este ciclo para cada instrucción que tiene que ejecutarse, hasta que el programa finalice su ejecución.

¿CÓMO SABE LA CPU CUÁL ES "LA PRÓXIMA INSTRUCCIÓN"?

Hay un registro interno de la unidad central de procesamiento, que se llama **puntero de instrucción**, que siempre tiene almacenada la dirección de memoria a la que hay que ir a buscar la próxima instrucción que debe ejecutarse. Antes de comenzar un nuevo ciclo de instrucción, la unidad de control actualiza su valor de modo que pase a contener la dirección de la próxima instrucción del programa que hay que ir a buscar a la memoria. En general, este valor será la siguiente posición de la memoria, aunque esto no siempre es así; por ejemplo, cuando se está en presencia de una alternativa condicional o una repetición.



NOMBRE Y APELLIDO:

CURSO: FECHA:

LA COMPUTADORA EN ACCIÓN

Cada vez que interactuamos con una computadora hacemos que se ejecuten programas: usamos una aplicación de mensajería instantánea, una que muestra fotos, un reproductor de música, etc. ¿Cómo hace la computadora para ejecutarlos? En esta actividad iremos a fondo para ver cómo se ejecutan los programas en lenguaje de máquina. ¡Salió la ultra fina *Bet & Rob air*! Mirá las especificaciones de *hardware*:

Memoria

Tiene una memoria de 18 bytes; los primeros 9 están reservados para programas y los restantes se usan para almacenar datos.

Registros de la unidad central de procesamiento

Posee 2 registros internos, `REG1` y `REG2`, que los programas usan para escribir y leer datos. Además, como cualquier otra computadora, también tiene un registro que actúa de puntero de instrucción: `PI`. Este, sin embargo, no puede ser mencionado en los programas.

CPU	
REG_1	PI
REG_2	→

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0		9	
1		10	
2		11	
3		12	
4		13	
5		14	
6		15	
7		16	
8		17	

NOMBRE Y APELLIDO:

CURSO: FECHA:

Lenguaje ensamblador

A continuación está el conjunto de instrucciones de su lenguaje ensamblador:

`ASIGNAR [r] [n]`: escribe en el registro *r* el número *n*.

`INCREMENTAR [r]`: incrementa en 1 el valor contenido en el registro *r*.

`DECREMENTAR [r]`: decrementa en 1 el valor contenido en el registro *r*.

`MOVER_DE_MEMORIA_A_REGISTRO [r] [n]`: escribe en el registro *r* el contenido de la celda de memoria cuya dirección es *n*.

`MOVER_DE_REGISTRO_A_MEMORIA [r] [n]`: lee el contenido del registro *r* y lo escribe en la posición *n* de la memoria.

`SALTO_CONDICIONAL [r] [n] [i]`: lee el contenido del registro *r*, lo compara con el número *n* y, si son iguales, establece que la siguiente instrucción a ejecutar es la *i*-ésima (es decir, la que está en la posición *i* en el programa); si no, el flujo de ejecución continúa con la instrucción que se encuentra inmediatamente a continuación.

`SALTO_INCONDICIONAL [i]`: establece que la siguiente instrucción a ejecutar es la *i*-ésima (es decir, la que está en la posición *i* en el programa).

`DETENER`: detiene la ejecución del programa.

1. Seguí paso a paso cómo evoluciona el estado del sistema a medida que se ejecuta el siguiente programa. Los valores iniciales de la memoria y los registros se observan a continuación.

CPU

PI
→

REG_1
0

REG_2
5

MEMORIA

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	MOVER_DE_MEMORIA_A_REGISTRO REG_1 9	**9**	2
1	ASIGNAR REG_2 0	**10**	0
2	SALTO_CONDICIONAL REG_1 0 7	**11**	0
3	INCREMENTAR REG_2	**12**	0
4	INCREMENTAR REG_2	**13**	0
5	DECREMENTAR REG_1	**14**	0
6	SALTO_INCONDICIONAL 2	**15**	0
7	MOVER_DE_REGISTRO_A_MEMORIA REG_2 10	**16**	0
8	DETENER	**17**	0

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿Qué hubiese sucedido si, inicialmente, en la posición 9 de la memoria hubiese habido un 3? ¿Y si hubiera un 10?

¿Qué hace el programa?

2. Realizá el seguimiento de la ejecución de este otro programa.

CPU

PI
→

REG_1
16

REG_2
12

MEMORIA

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	MOVER_DE_MEMORIA_A_REGISTRO REG_1 9	9	2
1	MOVER_DE_MEMORIA_A_REGISTRO REG_2 10	10	4
2	MOVER_DE_REGISTRO_A_MEMORIA REG_1 10	11	0
3	MOVER_DE_REGISTRO_A_MEMORIA REG_2 9	12	0
4	DETENER	13	0
5		14	0
6		15	0
7		16	0
8		17	0

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿Qué hace el programa?

¿Tiene alguna importancia el valor inicial de los registros? ¿Qué pasaría si al iniciar la ejecución del programa, el contenido de las direcciones de memoria 9 y 10 fuese otro?

3. Ahora fijate qué pasa con este.

CPU

PI
→

REG_1
0

REG_2
5

MEMORIA

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ASIGNAR REG_1 9	**9**	2
1	SALTO_CONDICIONAL REG_1 18 5	**10**	4
2	MOVER_DE_REGISTRO_A_MEMORIA REG_1 REG_1	**11**	0
3	INCREMENTAR REG_1	**12**	0
4	SALTO_INCONDICIONAL 1	**13**	0
5	DETENER	**14**	0
6		**15**	0
7		**16**	0
8		**17**	0

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿Cómo queda la memoria al finalizar la ejecución del programa?

¿Tiene alguna incidencia el contenido inicial de la memoria? ¿Y el valor de los registros?