Universidad Andrés Bello | Unab Online

Arquitectura de los Computadores

Zuccar, I. (2021). *Arquitectura de los Computadores* [apunte]. Chile. UNAB

ARQUITECTURA DE LOS COMPUTADORES

1. Introducción

Los computadores parecen ser una máquina bastante compleja... Han cambiado rápidamente en el tiempo y han evolucionado en su tecnología para hacerlos cada vez más potentes, más pequeños, y más cercanos a las personas, en sus hogares y lugares de trabajo, como herramientas de apoyo en sus tareas. Y, pese a que los computadores están presentes en nuestra vida diaria, para muchas personas su funcionamiento sigue siendo un "misterio".

Lo interesante de todo esto es que, en estos 70 años de evolución, en los computadores aún se pueden distinguir 5 componentes imprescindibles para su funcionamiento. En los siguientes apartados de este apunte, conocerás tales componentes y el objetivo de cada uno, además, revisarás los diferentes sistemas de bases numéricas, para luego pasar a cómo se representa la información dentro del computador y, finalmente, comprenderás qué es lo que ocurre internamente en el computador cuando ejecutas algún programa.

2. Componentes de un computador

Los componentes principales de un computador son:

- Dispositivos de entrada.
- Dispositivos de salida.
- Memoria secundaria.
- Memoria principal.
- Unidad central de procesamiento.
- Buses para interconectar a estos elementos.

Tanto los dispositivos de entrada como los de salida y algunos tipos de memoria secundaria, los podemos visualizar como artefactos externos a nuestro computador[1]. El resto de los componentes, sin embargo, son internos a él. A continuación, te explicaré cada uno de ellos.

2.1 Dispositivos de entrada:

Los dispositivos de entrada son los encargados de ingresar información del mundo real al computador. Por mundo real me refiero al mundo de nosotros los humanos.

[1] Por lo mismo, los dispositivos de entrada y salida también reciben el nombre de **periféricos**.

Los dispositivos de entrada, por lo tanto, también son los responsables de que ocurra la traducción[2] de la información, a la forma en la que el computador entiende[3].

En la figura 1 aparecen ejemplos de los dispositivos de entrada clásicos y también otros menos convencionales. Dentro de los dispositivos de entrada cabe destacar al teclado, que también se le conoce, en el mundo de la programación computacional, como entrada estándar (*standard input*).

Figura 1: (a) Teclado. **(b)** *Mouse*. **(c)** Cámara *web*: cámara sin pantalla, normalmente incorporada en los computadores. **(d)** Micrófono. **(e)** Lector de código de barras: usado comúnmente en los comercios. **(f)** Escáner: capaz de digitalizar documentos (algunos poseen programas computacionales capaces de transformar la imagen del texto, a texto real editable[4]). **(g)** *Joystick*: usado normalmente en algunos juegos.

[2] Esta traducción ocurre a través de programas computacionales que se deben instalar al momento de usar el dispositivo.

[3] La forma de comprender cómo se maneja la información dentro de un computador, te la explicaré en las siguientes secciones de este apunte.

[4] Estos programas se llaman OCR del inglés *Optical Character Recognition* o Reconocimiento Óptico de Caracteres.

2.2 Dispositivos de salida:

Los dispositivos de salida son los encargados de entregar la información que está dentro de nuestro computador, al mundo real. Los dispositivos de salida, por lo tanto, también se encargan de transformar la información almacenada dentro del computador, a una forma en la que los humanos comprendamos.

En la figura 2 aparecen ejemplos de dispositivos de salida. En particular, a la pantalla o monitor se le conoce en el mundo de la programación computacional, como salida estándar (*standard output*).

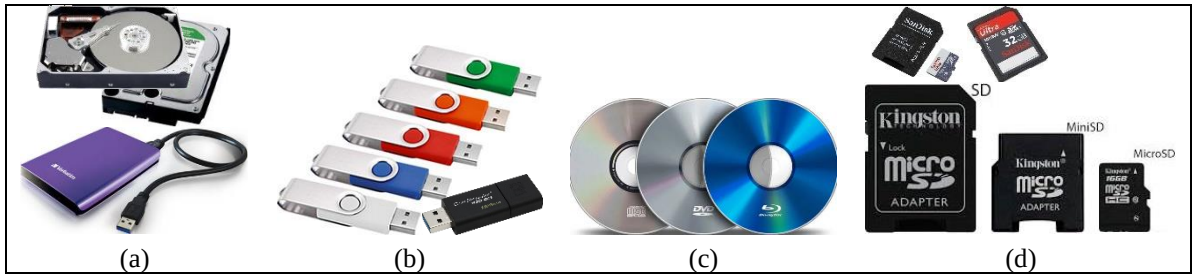
Figura 2: (a) Monitor o pantalla. **(b)** Audífonos. **(d)** Impresora. **(e)** Proyector.

2.3 La memoria secundaria:

La memoria secundaria se utiliza para almacenar en forma permanente programas computacionales y archivos. Si pensamos en los componentes internos del computador, entonces la memoria secundaria recibe el nombre de disco duro. Por otro lado, si pensamos en elementos externos, entonces la memoria secundaria corresponde a artefactos de almacenamiento que podemos transportar como discos duros externos, *pendrives*, tarjetas de memorias, por mencionar algunos. En la figura 3 puedes ver algunos ejemplos.

Almacenar en forma permanente significa que, aunque el computador esté apagado, la información seguirá allí; a no ser que tales memorias sufran de daños físicos[5].

[5] La nube apareció como una forma de evitar los "dolores de cabeza" por perder información de nuestros almacenamientos secundarios, debido a posibles daños físicos en estos elementos. Sin embargo, es importante que comprendas que la nube, sigue siendo físicamente una memoria secundaria en alguna parte del planeta, donde los computadores que alojan nuestra información poseen sistemas de respaldo y control de versiones bastante eficientes. Lo interesante de esta idea es que, a través de internet, podemos tener acceso a la información como si estuviera en nuestros computadores.



(a) (b) (c) (d)

Figura 3: (a) Disco duro interno (arriba) y externo (abajo). **(b)** *Pendrives*. **(d)** CD, DVD y *Blue-Ray*. **(f)** Tarjetas de memorias micro SD y adaptadores (de teléfonos móviles, cámaras fotográficas, etc.).

2.4 La memoria principal:

La memoria principal también es una forma de almacenamiento. Recibe el nombre de RAM[6] por su sigla en inglés *Random Access Memory* (que significa "Memoria de Acceso Aleatorio"). La diferencia fundamental con la memoria secundaria es su condición de volátil. Esto quiere decir que, cuando apagamos nuestro computador, la información que allí se almacena "**desaparece**". Entonces… ¿De qué podría servir una memoria que "olvida" la información que almacena?
La respuesta a esa pregunta es fundamental para comprender el funcionamiento del computador: la misión de la RAM es almacenar **todos los programas que se están ejecutando en nuestro computador**, y deja de almacenar, cuando cierras el programa (terminando su ejecución). La RAM también almacena los archivos que estemos usando y, en general, todos los datos que nuestros programas computacionales necesiten usar internamente (¡Y que nosotros ni nos enteramos!). En las siguientes secciones revisaremos en más detalle la RAM.

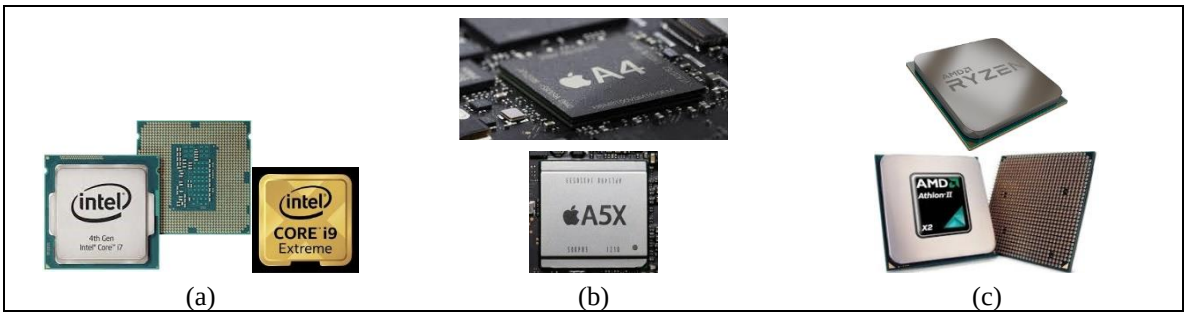
Figura 4: módulos de RAM.

[6] En este curso se usará indistintamente ambos términos: memoria principal o RAM.

2.5 Unidad central de procesamiento:

La unidad central de procesamiento o CPU (de su sigla en inglés *Central Processing Unit*) o simplemente procesador[7], es el "cerebro" de nuestro computador. Su función principal es la ejecución de los programas que queremos usar[8]. Un programa computacional es un conjunto de instrucciones escritas en el lenguaje que el procesador comprende. Por lo tanto, para que se ejecute algún programa en el computador, lo que en realidad ocurre, es que el procesador ejecuta una a una las instrucciones de tal programa.

En la figura 5 aparecen ejemplos de procesadores de los 3 principales fabricantes. Las instrucciones que un procesador puede ejecutar se establecen a nivel de circuitos y está definido desde su fabricación. Esto quiere decir que un procesador – y, por lo tanto, un computador - no puede ejecutar otras instrucciones que no sean las establecidas por su fabricante. A este conjunto de instrucciones se les conoce como **lenguaje de máquina**.



(a) (b) (c)

Figura 5: (a) Procesadores Intel. **(b)** Procesadores Apple. **(c)** Procesadores AMD.

[7] En este curso se usará indistintamente estos términos: unidad central de procesamiento, CPU o procesador.

[8] Para que no quede espacio a dudas, "ejecutar un programa del computador", es simplemente lo que haces cuando seleccionas algún programa de tu computador y "lo abres para usarlo" (por ejemplo, haciendo doble clic con el *mouse* sobre su ícono).

2.6 Buses:

Los buses corresponden al cableado para interconectar los componentes del computador. En cuanto a la función que cumplen, se distinguen los buses de datos, que transportan instrucciones y datos; los buses de direcciones, que transportan las direcciones que se deben acceder (para leer o para escribir información), y los buses de control que transportan las señales de lectura y escritura y ayudan en la sincronización. Los buses están construidos con diferentes tecnologías según su objetivo, su ancho (cantidad de bits que puede transmitir en paralelo), su frecuencia (velocidad en que los datos son transmitidos) y su velocidad (cantidad de datos que transportan por unidad de tiempo); varían desde las pistas en un circuito impreso hasta cables propiamente tal (en la figura 6 puedes ver algunos ejemplos de buses).

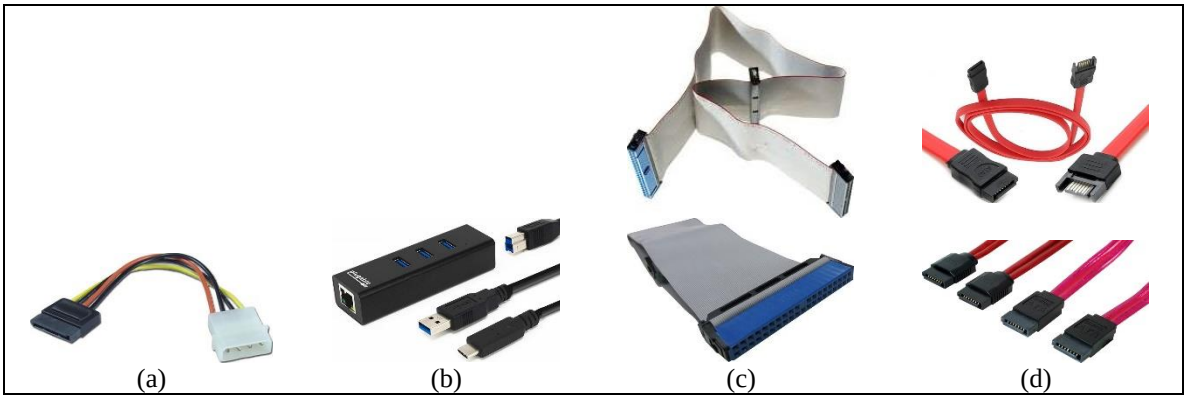


Figura 6: Ejemplos de buses. **(a)** Bus de control. **(b)** Bus USB (*Universal Serial Bus*) usado para conectar distintos periféricos. **(c)** Bus ATA o IDE usado para conectar discos duros y unidades de CD-ROM. **(d)** Bus Serial ATA (SATA) es una evolución del bus ATA.

3. Bases numéricas

En una asignatura como esta, debemos estar dispuestos a abrir nuestra mente para cambiar la forma en la que hemos hecho las cosas hasta ahora. Uno de estos cambios de *paradigmas* corresponde a la forma en la que se representan los números. Si lo piensas bien, los números aparecieron en la humanidad por la necesidad de contar. Usando marcas, nudos, o piedrecillas, nuestros antepasados pudieron conocer la cantidad de bienes que poseían. Sin embargo, el hecho de que se ocupen 10 dígitos en un sistema de posiciones[9] para representar cualquier número, el *dibujo* con el que se representa cada dígito y el nombre que posee, fue inventado por el hombre basado en una idea muy simple: tenemos 10 dedos *a la mano* para contar. Pero te invito a

[9] Las posiciones corresponden a la posición de las unidades, decenas, centenas, unidades de mil, etc.

considerar lo siguiente: ¿Cómo hubieran sido nuestros números si hubiésemos tenido solo 2 dedos en total? O ¿Cómo hubieran sido nuestros números si hubiéramos tenido 8 dedos en cada mano? Lo que hubiese ocurrido es que nuestro sistema de numeración hubiese sido **binario** (que usa 2 dígitos) o **hexadecimal** (que usa 16 dígitos), en vez del sistema **decimal**. Para saber cómo se transforma un número de una base a otra, existen ciertos *algoritmos*[10] que no ahondaremos en este curso. Sin embargo, haremos un pequeño ejercicio a continuación, que te permitirá tener una idea de cómo se construyen los números en una base numérica distinta a la decimal. Considera los números en nuestra base cotidiana, hasta el 17: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17. ¿Cómo se representarían estos mismos números en base numérica binaria? Para responder esto, deberás anotar todos los números en base decimal hasta el 10001 (*diez mil uno*)[11]: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, …, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, …, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, ..., 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, …, 9999, 10000, 10001. Ahora, destacaré en rojo a aquellos números que se escriben usando **solo** a los dígitos 0 o 1. Recuerda que el sistema binario solo utiliza 2 dígitos, y estos justamente corresponden al 0 y al 1: **0**, **1**, 2, 3, 4, 5, 6, 7, 8, 9, **10**, **11**, 12, 13, 14, 15, 16, 17, 18, 19, 20, …, 98, 99, **100**, **101**, 102, 103, 104, 105, 106, 107, 108, 109, **110**, **111**, …, 999, **1000**, **1001**, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, **1010**, **1011**, ..., 1099, **1100**, **1101**, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, **1110**, **1111**, …, 9999, **10000**, **10001**. De esta manera, el **0** y el **1**, corresponden al 0 y al 1 en base decimal; luego el **10**, corresponde al 2 en base decimal, el **11** al 3 en base decimal, el **100** al 4 en base decimal, y así sucesivamente (ver tabla 1).

Por otro lado, la representación hexadecimal considera el uso de 16 dígitos. Para esto, se consideran desde el 0 al 9, y luego se agregan 6 dígitos más, que corresponden a las 6 primeras letras del abecedario. De esta manera, los dígitos que contemplan son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **A**, **B**, **C**, **D**, **E**, **F**. Los *dígitos* A, B, C, D, E y F corresponden, respectivamente, a los números en base decimal 10, 11, 12, 13, 14 y 15 (ver tabla 1).

[10] Un algoritmo es una secuencia de pasos finitos y precisos, que cumplen con una tarea particular. Para comprender su definición, se suele hacer la analogía entre un algoritmo y una receta de cocina.

[11] Por espacio y tiempo, aparecen puntos suspensivos para aclarar que allí están los números intermedios, pero que no son importantes para este ejercicio.

Tabla 1: Escritura de las mismas cantidades usando diferentes bases de numeración: decima, binaria y hexadecimal.

Base numérica decimal	Base numérica binaria	Base numérica hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

4. RAM: Almacenamiento y Representación de la información.

La figura 4 muestra una RAM físicamente. Sin embargo, cuando construimos nuestros propios programas computacionales, es importante que imagines a la RAM como se muestra en la figura 7: esta imagen corresponde a la representación lógica de la RAM. Como puedes observar, la RAM se representa como una gran cuadrícula de *celditas*: organizadas con 8 *celditas* hacia el lado y *muuuuchas*, hacia abajo.

Como los sistemas computacionales son con base en la electrónica, en cada una de esas *celditas* hay cierto nivel de voltaje almacenado. Si la *celdita* tiene más de 5 volts, se dice que hay almacenado un 1. Si tiene menos de 5 volts, se dice que hay almacenado un 0. Esta *celdita* es la unidad más pequeña de almacenamiento de información, y se llama **bit** (cuadrados rojos de la figura 7.b). La combinación de 8 bits recibe el nombre de **byte** (rectángulos azules de la figura 7.b). Medidas más grandes de almacenamiento son: kilobyte (KB) = 1024 bytes; megabyte (MB) = 1024 KB; gigabyte (GB) = 1024 MB y un terabyte (TB) = 1024 GB. Recuerda que todas estas son unidades de medidas de almacenamiento y son usadas para las memorias primarias y secundarias, y para el ancho de los buses.

En la RAM dibujada, también puedes observar que cada byte tiene una ubicación respecto de la misma. Este valor es llamado **dirección**, y corresponde al valor exacto donde se encuentra cada byte (rectángulo verde de la figura 7.b). Las direcciones de memoria se suelen representar en con números en base hexadecimal.

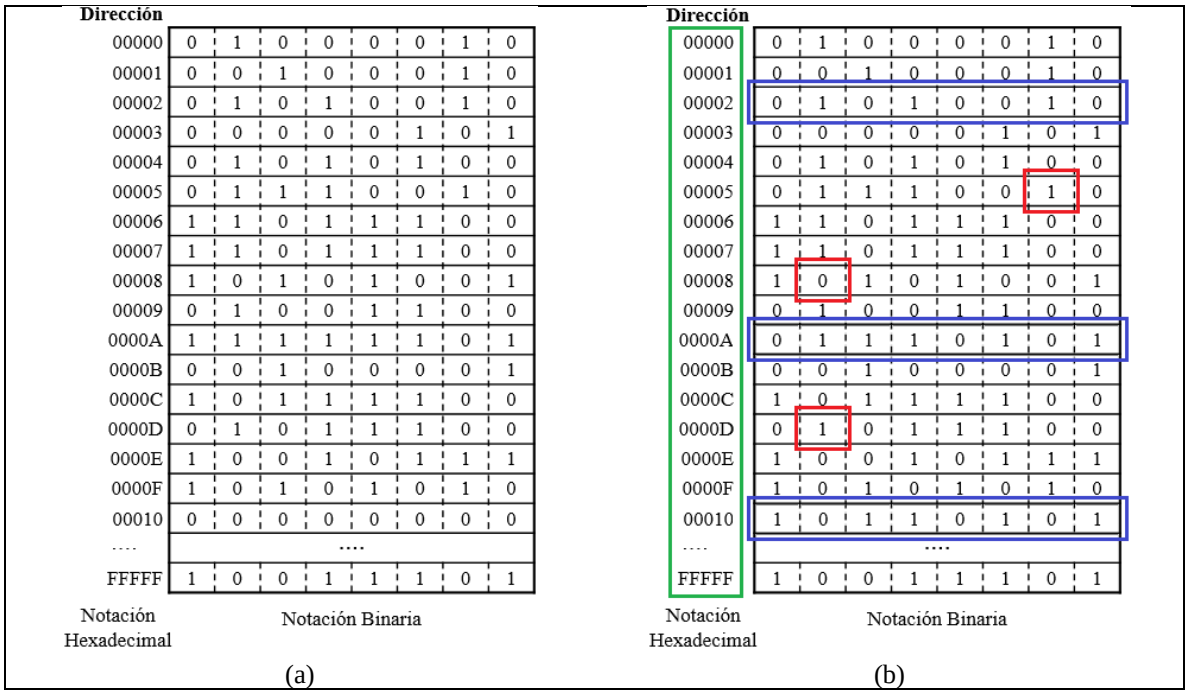


Figura 7: (a) Esquema de la RAM en términos lógicos. **(b)** Los marcos rojos destacan a 3 bits distintos, los marcos azules, a 3 bytes diferentes y, el marco verde de la izquierda, a las direcciones de memoria donde se ubica cada byte.

Como podrás imaginar, **todo lo que se almacena en la RAM (y dentro de nuestro computador) está representado en números binarios**. Aunque es el sistema operativo[12] de nuestro computador el encargado de administrar la RAM es importante, cuando se aprende a programar, comprender y recordar que en términos lógicos la RAM se divide en 3 partes o segmentos: el segmento de datos, el segmento de programas y el segmento de pila o *stack*. Por simplificación para este curso, solo consideraremos los segmentos de datos y de programas de la RAM. El segmento de programas es el que efectivamente almacena los programas en ejecución. Por otro lado, el segmento de datos es el que almacena la información que los programas necesitan para funcionar correctamente (en la figura 8 aparece una posible partición para estos segmentos).

[12] El sistema operativo es el programa fundamental que debe poseer nuestro computador para que inicie su correcto funcionamiento, administrando sus recursos. Cuando compras un computador ya viene instalado el sistema operativo (normalmente Windows) para que esté listo para su uso.

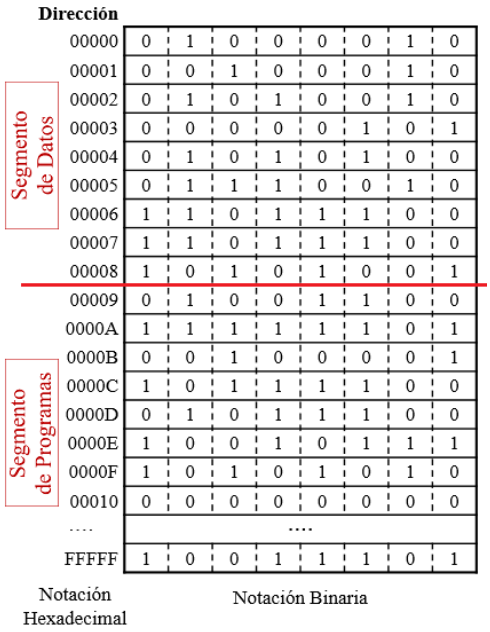


Figura 8: Esquema de la RAM en términos lógicos donde se representa una posible partición en los segmentos de datos y de programas. Lo interesante de esto es que los 0s y 1s del segmento de datos se interpretarán como información (números, letras, una imagen, una canción, etc.), mientras que los 0s y 1s del segmento de programas se interpretarán como instrucciones en lenguaje de máquina.

4.1 Segmento de Datos:

Toda la información que está guardada en el segmento de datos son números escritos en base binaria. Veremos algunos ejemplos para comprenderlo mejor.

Ejemplo 1: Si una aplicación necesita almacenar el valor **57**, se hará la transformación internamente a base binaria, y lo que en realidad se almacenará será **111001**.

Ejemplo 2: Si una aplicación necesita almacenar el valor **10437**, se hará la transformación internamente a base binaria, y lo que en realidad se almacenará será **10100011000101** (observa que este número requiere 2 bytes para ser almacenado, es decir, 16 bits).

Ejemplo 3: Si una aplicación necesita almacenar el valor **-1,82**, se hará la transformación internamente a base binaria, y lo que en realidad se almacenará será

00111111000000000000000000000000. Observa que este número requiere 4 bytes (32 bits) para ser almacenado (estándar IEEE 754[13]).

Cada caracter (símbolo, letra, dígito) que se puede almacenar en el computador tiene un número asociado y estandarizado. Esto es un estándar a nivel mundial y nace a partir de la codificación ASCII. La figura 9 muestra la tabla de códigos ASCII, y la figura 10, la tabla de códigos ASCII extendida. También existe la codificación UNICODE que además de contener a la tabla ASCII agrega otros caracteres necesarios para la escritura de otros idiomas[14].

Ejemplo 4: Si una aplicación necesita almacenar la frase **Amor!**, se calculan los códigos ASCII de cada caracter, y luego se almacenan como números binarios. De esta forma la **A** se almacenará como **01000001** (ya que le corresponde el código **65** en base decimal); la **m** se almacenará como **01101101** (le corresponde el código **109** en base decimal); la **o** se almacenará como **01101111** (código **111** en base decimal); la **r** se almacenará como **01110010** (código **114** en base decimal) y, finalmente, el símbolo **!**, se almacenará como **00100001** (código **33** en base decimal). Observa que, para almacenar toda la frase, se ocuparían 5 bytes.

ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo
0	NUL	16	DLE	32	(espacio)	48	0
1	SOH	17	DC1	33	!	49	1
2	STX	18	DC2	34	"	50	2
3	ETX	19	DC3	35	#	51	3
4	EOT	20	DC4	36	$	52	4
5	ENQ	21	NAK	37	%	53	5
6	ACK	22	SYN	38	&	54	6
7	BEL	23	ETB	39	'	55	7
8	BS	24	CAN	40	(	56	8
9	TAB	25	EM	41	)	57	9
10	LF	26	SUB	42	*	58	:
11	VT	27	ESC	43	+	59	;
12	FF	28	FS	44	,	60	<
13	CR	29	GS	45	-	61	=
14	SO	30	RS	46	.	62	>
15	SI	31	US	47	/	63	?

ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo
64	@	80	P	96	`	112	p
65	A	81	Q	97	a	113	q
66	B	82	R	98	b	114	r
67	C	83	S	99	c	115	s
68	D	84	T	100	d	116	t
69	E	85	U	101	e	117	u
70	F	86	V	102	f	118	v
71	G	87	W	103	g	119	w
72	H	88	X	104	h	120	x
73	I	89	Y	105	i	121	y
74	J	90	Z	106	j	122	z
75	K	91	[	107	k	123	{
76	L	92	\	108	l	124	|
77	M	93	]	109	m	125	}
78	N	94	^	110	n	126	~
79	O	95	_	111	o	127	

Figura 9: Tabla de códigos ASCII (del inglés *American Standard Code for Information Interchange*, que se puede traducir como: "Código Americano Estandarizado para el Intercambio de Información". Esta tabla codifica los caracteres básicos utilizando para aquello los valores desde el 0 al 127. Inicialmente se usó 7 bits para esta representación, lo que permitía 128 posibles combinaciones.

[13] Si quieres conocer la forma de representar números negativos y con decimales en base binaria, puedes revisarlo en el siguiente enlace: https://es.wikipedia.org/wiki/IEEE_754

[14] Puedes revisar más detalle de la historia y esfuerzos para la estandarización de la codificación en el computador en el enlace: https://es.wikipedia.org/wiki/ASCII

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Figura 10: Tabla de códigos ASCII extendida. Codifica entre otros, símbolos no usados en el inglés como son las letras con diferentes tipos de tildes, la **ñ** y la **Ñ**. Esta tabla utiliza los códigos entre el 128 y el 255, considerando 1 bit más en la codificación original. Esto amplió las combinaciones de 128 a 256.

4.2 Segmento de Programas:

Todas las instrucciones que poseen nuestros programas están escritas en base binaria. ¡Este es el único leguaje que nuestro computador entiende! Como te mencioné en la sección 2.5 de este apunte, este lenguaje se llama **lenguaje de máquina** y el set de instrucciones que lo conforma, lo establece el fabricante de nuestro procesador. Qué secuencia de 0s y 1s corresponde a qué instrucción, está documentado por el fabricante. En los primeros años de los computadores, los programas computacionales se escribían directamente en este lenguaje. En la figura 11 se muestra lo que era un programa computacional (o parte de un programa) en las décadas de los ’60 y ’70.

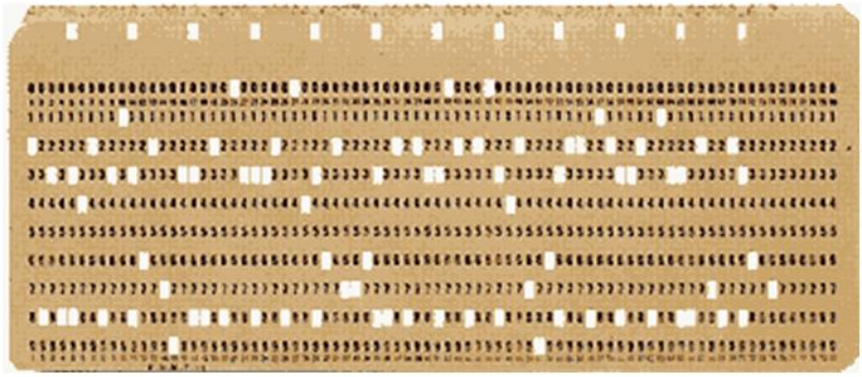


Figura 11: Programa computacional escrito directamente en lenguaje de máquina. Se le conocía como **tarjeta perforada**. La programación consistía en perforar la tarjeta para representar un 1; si la tarjeta no se perforaba, significaba que se almacenaba allí un 0. Cada instrucción es una columna (se *lee* verticalmente) de esta tarjeta.

5. Interacción entre los componentes de un computador: Ciclo de ejecución

Para finalizar este apunte, te explicaré los pasos que sigue nuestro computador para ejecutar cualquier programa que queramos usar. Este proceso recibe el nombre de **ciclo de ejecución**.

Cuando haces doble clic sobre el programa que deseas usar, el programa empieza a ejecutarse. Lo que internamente ocurre es que todas las instrucciones de tal programa, que están almacenadas en algún dispositivo de memoria secundaria, normalmente en el disco duro, se copian a la RAM, concretamente al segmento de programas. Luego la CPU, copia la primera de las instrucciones a sus registros internos y, si es que esa instrucción requiere datos, copia también los datos del segmento de datos, y luego ejecuta la instrucción. Si obtuvo algún resultado, lo escribe en la RAM en el segmento de datos, y continúa con la siguiente instrucción. De esta manera, el procesador ejecuta una a una las instrucciones hasta que termina la ejecución. Ahora bien, si el programa que deseas ejecutar tiene una interfaz interactiva con el usuario, entonces, el procesador espera a que tú interactúes[15], y el procesador ejecuta las instrucciones que corresponde a la acción que tú estás realizando.

6. Bibliografía

1. **Downey, A. (2013). Think Python. Green Tea Press.** Accesado el 2 de diciembre de 2019 en https://greenteapress.com/wp/think-python-2e/

2. **Python software foundation, Python v3 Documentation.** Accesado el 2 de diciembre de 2019 en https://docs.python.org/3/

3. Marzal Varó, A., Gracia Luengo, I., & García Sevilla, P. (2014). Introducción a la programación con Python 3. Universitat Jaume I. https://doi.org/10.6035/sapientia93 (Capítulo 1, secciones 1.1 y 1.2, páginas: 11 - 16)

[15] Por ejemplo, que pintes alguna zona de una imagen en el *Paint*, que escribas algún texto en un documento de *Word*, que ingreses un dato en una planilla de *Excel*, o que hagas alguna operación en la *calculadora de Windows*, por mencionar algunos.