IPF

Instituto Politécnico Formosa

José María Uriburu 820 - Formosa

TECNICATURA SUPERIOR EN DESARROLLO DE SOFTWARE MULTIPLATAFORMA

ASIGNATURA: Arquitectura y Sistemas Operativos

Año 2022

Instituto Politécnico Formosa

UNIDAD III - Parte 1

Representación de la información

Índice

3.1. Introducción

En ésta unidad se describe la representación interna de los formatos de “tipos de datos" que los lenguajes de programación permiten declarar, para las distintas estructuras de dato. Por ejemplo: una variable **stríng** se almacenará en un código de representación alfanumérico, como UNICODE o ASCII; una variable *unsigned integer*, como un número entero no signado, conocido como "de coma fija” o "de punto fijo”; un número como el $7{,}5_{(10)}$ se almacenará como un número de "punto flotante” o “coma flotante”. La mayoría de ellos se ajusta a convenios estandarizados, de los cuales los consideran los más difundidos en los diseños de las computadoras.

3.2. Flujo de datos dentro de una computadora

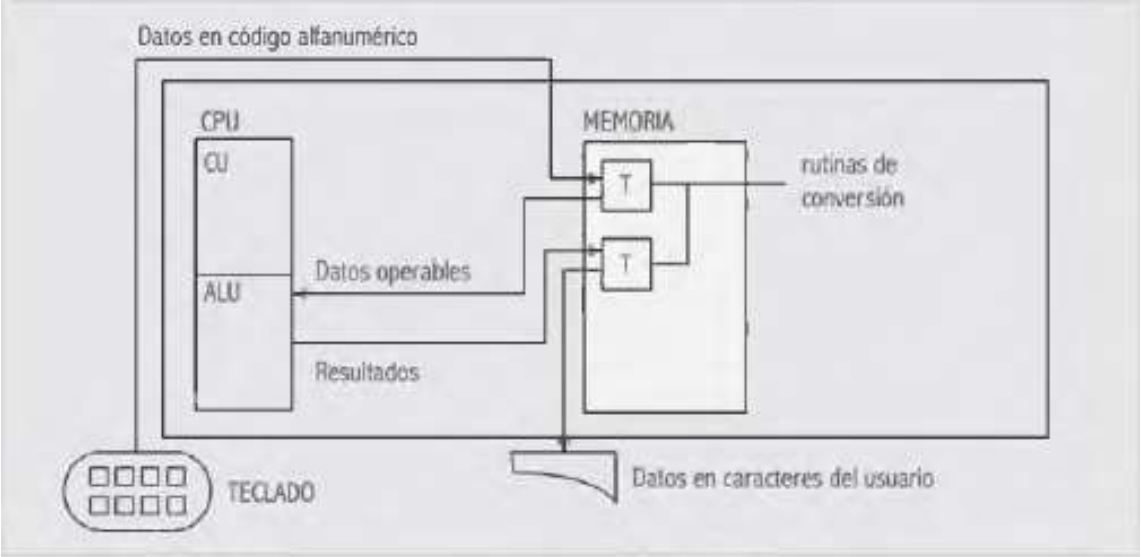
Una computadora está constituida por una memoria, que almacena programas y datos en forma temporal, y un órgano ejecutor de estos programas conocido como unidad central de proceso (CPU).

Si se la analiza desde un ejemplo más cotidiano, pero práctico, la **memoria** cumple la función de una hoja en la que se escribieron los pasos para la resolución de un problema y los **datos** que intervienen en los cálculos y donde se volcarán los resultados obtenidos. A su vez, se interpretan los pasos, se toma los datos y se los vuelca en una calculadora. Dentro de la computadora, quien realiza esta actividad es la **unidad de control** (CU) y la de la calculadora es la **unidad aritmético-lógica** (ALU). Así, el conjunto CU-ALU constituye la ya mencionada **CPU**.

Las variaciones en el diseño interno de las distintas computadoras actuales no modifican este esquema global, pero sí afectan el procesamiento de los programas y los datos que deben transformarse en el interior para adaptarse a la estructura de cada computadora. Sin embargo, el programador no es el responsable de implementar estas transformaciones; por ejemplo, la compilación de un programa en lenguaje simbólico permite adaptar su estructura y generar el programa en el código de máquina que la computadora puede entender. Entonces hay un “mediador” entre los programas y los datos que vienen "de afuera” y las necesidades internas para procesarlos. Este vínculo que pasa de lo estándar a lo particular en cada equipo es una rutina de conversión enlazada al código del programa.

En resumen, tanto las instrucciones de un programa como los datos que ingresan en la memoria desde el exterior lo hacen en un código alfanumérico de representación de caracteres, por ejemplo, el ASCII. Las instrucciones son transformadas por un programa de traducción (que puede ser un ensamblador, un traductor o un compilador) a código de máquina, que es el que entiende el procesador, para luego ejecutarse. Los encargados de la transformación de los datos, respetando el formato definido en el programa, suelen ser rutinas previstas por bibliotecas estándar del lenguaje e incluidas en el programa. Por lo tanto, aunque los datos ingresen desde el teclado en un código alfanumérico de caracteres, éstos se almacenarán en la memoria de distintas formas, según los requerimientos determinados por la declaración de los datos del programa; éste es el único que puede reconocer si el byte de memoria accedido es, por ejemplo, un operando o un carácter alfanumérico como vemos en la figura.

Por otra parte, una computadora necesita comunicarse con el medio externo por medio de dispositivos de entrada o salida, y éstos se diseñan para conectarlos a una amplia gama de computadoras diferentes entre sí, lo que induce a pensar que los periféricos utilizan códigos estándar para la representación de información. La forma de ingreso o egreso de datos y programas en una computadora respeta estos códigos de representación de caracteres que abastecen las necesidades de intercambio de información a nivel general. Esto es independiente de las transformaciones que se realizan para su tratamiento interno.



Código alfanumérico

Un **código alfanumérico** establece la relación necesaria para que una computadora digital, que procesa solamente dígitos binarios, interprete el lenguaje que utiliza el usuario (caracteres numéricos, alfabéticos o especiales). Un **código** es una convención que determina una única combinación binaria para cada símbolo que se ha de representar. Por ejemplo, con 7 bits se obtienen $2^7 = 128$ combinaciones distintas, que son suficientes para asignar un símbolo alfabético, numérico o especial a cada una de ellas.

Si bien se puede establecer infinidad de códigos alfanuméricos de "entendimiento" entre el usuario y las distintas computadoras, se desarrollaron convenios de uso internacional, cuya finalidad es concretar la compatibilidad entre dispositivos de distinto origen.

Cada caracter emitirá una serie de señales binarias al "digitarse" sobre el teclado, que quedará alojada internamente como una combinación de bits, de acuerdo con alguna representación de caracteres alfanuméricos preestablecida.

Supóngase que se ingresa el dato alfanumérico "+105", el proceso de codificación hará que éste quede registrado en la memoria, por ejemplo, en código ASCII, como se muestra en la figura.

0010 1011	+
0000 0001	1
0000 0000	0
0000 0101	5

Símbolos del usuario y su representación binaria en ASCII

Se denomina **formato** de una entidad binaria a la estructura y la cantidad de bits de un determinado tipo de dato para su tratamiento en la computadora. Los **formatos** pueden ser de longitud fija o variable, según la categoría de dato que represente. Es importante destacar que el que interpreta un grupo de bits bajo un formato determinado es un programa. Esto significa que una cadena de bits en un registro de la ALU puede ser interpretada por el programa que la utiliza como operando, como un binario sin signo o con signo, o simplemente puede representar una parte del operando en sí. Del mismo modo, un byte alojado en la memoria puede ser un carácter alfanumérico, un operando o incluso una instrucción, según quién lo "interprete".

En la tabla de abajo se detallan distintos tipos de representaciones de datos que una computadora puede manejar y que más adelante veremos. Debe quedar claro que, en general, del conjunto de posibilidades para cada una de las representaciones, sólo se utilizará una de ellas. Por ejemplo, para una computadora las representaciones alfanuméricas siempre se interpretarán en código ASCII, las representaciones decimales se manejarán en BCD puro, los números enteros se expresarán en complemento a la base y los números reales utilizarán una representación en IEEE P754. De manera independiente, la misma computadora puede utilizar formatos de números enteros en punto fijo sin signo o con signo, ya que ello depende de la definición del tipo de dato que el programa de usuario necesite.

Representaciones alfanuméricas	ASCII UNICODE	
Representaciones decimales (BCD)	BCD puro o natural (8421) BCD exceso tres BCD 2421 o AIKEN	
Representaciones binarias	Números enteros	Coma o punto fijo sin signo (enteros-positivos) Coma o punto fijo con signo (enteros) Coma o punto fijo con complemento a la base (enteros) Coma o punto fijo con complemento restringido (enteros)
	Números reales	Coma o punto flotante (entera y fraccionaria) Coma o punto flotante con mantisa normalizada Coma o punto flotante IEEE P754 Coma o punto flotante "exceso 64"
Representaciones redundantes (detección de errores)	Códigos de paridad	Paridad vertical o a nivel carácter Paridad vertical o a nivel bloque Paridad entrelazada Código de Hamming

3.3. Códigos de representación de caracteres alfanuméricos

Si bien una computadora es una “calculadora aritmética” por excelencia, gran parte de sus aplicaciones consiste en el tratamiento de cadenas de caracteres llamadas stríngs. Un stríng es una unidad formada por caracteres representados según un **código alfanumérico** preestablecido para esa computadora. Uno de los códigos de representación de caracteres más usado es el "ASCII".

3.3.1 Código ASCII

El código **ASCII** (American Standard Code for Information Interchange o Código Estándar Americano para Intercambio de Información), es un código de 7 bits, representa cada caracter con 7 bits, que permite determinar 2^7= 128 combinaciones distintas, suficientes para establecer una única relación carácter-combinación binaria, que se representa en la siguiente tabla.

	HEX. N°	0	1	2	3	4	5	6	7
HEX. N°	BITS 654 3210	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DCI	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	¦
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

Tabla de código ASCII (de 7 bits).

Significado en inglés de los caracteres de control de las columnas 0 y 1		
NUL, *null*	VT, *vertical tabulation*	SYN, *synchronous idle*
SOH, *start of heading*	FF, *form feed*	ETB, *end of transmission block*
STX, *start of text*	CR, *carriage return*	CAN, *cancel*
ETX, *end of text*	SO, *shift out*	EM, *end of medium*
EOT, *end of transmission*	SI, *shift in*	SUB, *substitute*
ENQ, *enquiry*	DEL, *data link escape*	ESC, *escape*
ACK, *acknowledge*	DC1, *device control 1*	FS, *file separator*
BEL, bell (*audible signal*)	DC2, *device control 2*	GS, *group separator*
BS, *backspace*	DC3, *device control 3*	RS, *record separator*
HT, *horizontal tabulation (punched card skip)*	DC4, *device control 4*	US, *unit separator*
LF, *line feed*	NAK, *negative acknowledge*	DEL, *delete*

Este patrón se puede definir como “oficial' y en todos los textos en los que se incluya esta tabla no habrá variaciones que den lugar a pensar que “hay varios ASCII”.

Los primeros 3 bits (6, 5 y 4) de mayor significación en cada combinación se **denominan bits de zona** y en la tabla de doble entrada numeran las columnas 0 a 7.

Los últimos 4 bits (3, 2, 1 y 0) en cada combinación se denominan **bits de dígito** y en la tabla numeran las filas de 0 a 15, o de 0 a F en hexadecimal.

Los primeros 32caracteres pueden parecer “raros", sin embargo, son de gran utilidad porque permiten **cierto control** de las actividades de la computadora. Por ejemplo, para tabular hay una tecla en el teclado que permiten “ocupar" un lugar en la pantalla con la combinación 0000 1001; aunque no se visualice, "marcan” ese lugar para que el cursor salte allí. El carácter **BEL** permite emitir un sonido que puede utilizarse como aviso al operador de la computadora. Los códigos **SO** y **SI**, shift activo o inactivo, se utilizan para ordenar a la impresora que imprima caracteres estándar o condensados. El resto de los caracteres es familiar para el usuario: Letras mayúsculas y minúsculas, dígitos, signos de puntuación, etcétera. Sin embargo, hay ciertas cuestiones para tener en cuenta. Por un lado, las mayúsculas se consideran distintas a las minúsculas, aun cuando se trate de la misma letra, debido a que su forma de representación simbólica es diferente por completo (A vs. a). Las combinaciones que representan letras están numéricamente ordenadas, esto es, la combinación de “A” es menor que la de "B”, lo que permite ordenar alfabéticamente un conjunto de datos. Por último, el espacio en blanco está definido por la combinación 0010 0000 porque el blanco o el espacio "ocupan lugar, aunque no se vean”.

Para terminar, es importante saber que la combinación binaria que representa un carácter también se expresa en sus formas hexadecimal y decimal de la siguiente forma:

“A" =100 0001 = 41h = 65d

3.3.2 Código ASCII ampliado

HEX. N°	BITS 7654 3210	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111
0	0000	NUL		SP	0	@	P	`	p
1	0001		DCI	!	1	A	Q	a	q
2	0010		DC2	"	2	B	R	b	r
3	0011		DC3	#	3	C	S	c	s
4	0100		DC4	$	4	D	T	d	t
5	0101			%	5	E	U	e	u
6	0110			&	6	F	V	f	v
7	0111	BEL		'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF		*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF		,	<	L	\	l	|
D	1101	CR		-	=	M	]	m	}
E	1110	SO		.	>	N	^	n	~
F	1111	SI		/	?	O	_	o	DEL

HEX. N°	BITS 7654 3210	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0	0000	Ç	É	á	▒	À	Ð	a	=
1	0001	ü	æ	í	▒	Á	Ñ	ß	±
2	0010	é	Æ	ó	▒	Â	Ò	G	‗
3	0011	â	ô	ú	³	Ã	Ó	p	¾
4	0100	ä	ö	ñ	´	Ä	Ô	S	¶
5	0101	à	ò	Ñ	µ	Å	Õ	s	§
6	0110	å	û	ª	¶	Æ	Ö		÷
7	0111	ç	ù	º	·	Ç	×	t	¸
8	1000	ê	ÿ	¿	¸	È	Ø	F	°
9	1001	ë	Ö	⌐	¹	É	Ù	Θ	¨
A	1010	è	Ü	¬	º	Ê	Ú	O	·
B	1011	ï	¢	½	»	Ë	Û	d	√
C	1100	î	£	¼	¼	Ì	—	8	n
D	1101	ì	¥	¡	½	Í	¦	f	²
E	1110	Ä	₧	«	¾	Î	¦	e	■
F	1111	Å	ƒ	»	¿	Ï	¯	n	□

Si bien el ASCII se concibió como código de 7 bits, su uso generalizado dio lugar a una ampliación a causa de la necesidad de agregar caracteres. Ésta es la razón por la que se incluye la tabla ASCII de 8 bits, 4 bits de zona y 4 de dígito. Los primeros 128 caracteres son los originales. Los segundos 128 son los agregados y aquí se puede hablar de caracteres "no oficiales", dado que las tablas no guardan uniformidad; en su mayoría, representan caracteres gráficos y no son

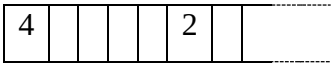
esenciales. Algunos ejemplos son el símbolo del Yen japonés -D9h-, el franco -F9h-, el signo de interrogación de apertura que utiliza el idioma español - 8Ah-, caracteres de contorno (doble línea, línea simple), caracteres de brillo o intensidad que se superponen en una pantalla sobre otros caracteres, etcétera.

3.3.3 Delimitación de strings

Las cadenas de caracteres alfanuméricas (strings), procesadas y almacenadas en la computadora, pueden ser:

- **De longitud fija:** cada dato ocupa un número determinado de bytes fijo. Por lo tanto, el comienzo y el final de cada dato puede conocerse con rapidez.
- **De longitud variable:** para determinar el final de cada dato hay dos métodos:

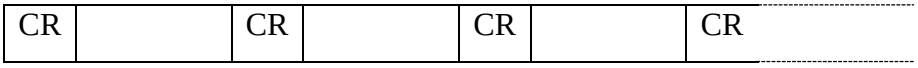
  - Cada dato tiene en su inicio un campo en el que se indica la longitud en bytes, como lo muestra la figura.



  - Cada dato se separa de los colindantes mediante un símbolo específico o separador, que se utiliza para marcar el final de la cadena de caracteres, pero no se considera parte de ésta. Hay dos separadores muy usados:

  1) El “byte 0" o NUL.

  2) El byte con un código identificado en la tabla ASCII como de "retorno de carro" (CR), que por lo general se utiliza para el final de una línea de texto, que en el teclado equivale a la tecla <ENTER>. En la figura se observa ese caso.



CR (carriage return).

Para dar un ejemplo, el *string* “PRIMER AÑO” se representa en ASCII de 8 bits (en hexadecimal) como:

P	**R**	**I**	**M**	**E**	**R**		**A**	**Ñ**	**O**	**CR**
50	**52**	**49**	**4D**	**45**	**52**	**20**	**41**	**A5**	**4F**	**0D**

Ejercicios:

Expresar las siguientes frases en por medio del su código ASCII en binario, en hexadecimal y comprobar cada carácter convirtiéndolo en decimal.

a) ARQUITECTURA Y SISTEMAS OPERATIVOS
b) BASE DE DATOS I
c) DESARROLLO DE SOFTWARE

3.4. BCD (Binary Coded Decimal)

El código BCD se utiliza en las computadoras para representar los números decimales 0 a 9 empleando el sistema de numeración binario. Los números representados en código BCD se escriben utilizando ceros y unos.

Son códigos de representación de números y se los denomina códigos ponderados, porque adjudican cierto peso a los 1 binarios, según la posición que ocupan en el bloque, por lo que se debe verificar que la suma de los pesos de cada combinación sea igual al número decimal representado. La tabla Código BCD especifica la codificación de caracteres numéricos.

Decimal	Decimal Codificado en Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

A partir de la tabla Código BCD, se observa que este código requiere el empleo de un carácter binario de cuatro posiciones (cuatro bits) para especificar el carácter de un dígito decimal.

Evidentemente, este código es mucho menos eficiente que el sistema decimal, pero presenta la ventaja de especificar los caracteres mediante las cifras 0 y 1, que constituyen el lenguaje del computador, por lo que el código BCD puede ser utilizado en una computadora. Algunos ejemplos de representación de números decimales en este código son:

Decimal	BCD
22	0010 0010
35	0011 0101
671	0110 0111 0001
3256	0010 0101 0111 1001

Puede verse que cada cifra decimal requiere un conjunto binario equivalente de cuatro bits. Para especificar un número, el código BCD requiere más posiciones que el sistema decimal. Otro punto que debe tenerse presente es que la posición de cada bit, dentro de los cuatro bits de cada cifra, es muy importante (como sucede en todo sistema de numeración posicional). Puede especificarse la ponderación de cada una de las posiciones y algunas veces se emplea para indicar la forma de codificación. El peso de la primera posición (situada a la derecha es 2^0=1, el de la segunda, 2^1=2; el de la tercera, 2^2= 4 y el de la cuarta, 2^3=8. Leyendo el número de la izquierda a derecha, la ponderación es 8-4-2-1.

Cabe aclarar que esta codificación en BCD no es lo mismo que los números binarios, consideremos los casos siguientes:

Decimal	Binario	BCD
10		
18		

La confusión entre los códigos BCD y Binario se origina debido a que son exactamente iguales las nueve primeras cifras en BCD y en Binario. Después, los números son completamente diferentes.

Por ejemplo, compárense las representaciones Binaria y BCD leyendo los números en cada una de sus formas.

Decimal	Binario	BCD
141	10001101	0001 0100 0001
2179	100010000011	0010 0001 0111 1001
254		
87		
1234		

Sin embargo, cuando se utiliza esta forma de codificación en operaciones aritméticas se presentan dificultades adiciónales. Veamos lo que sucede cuando se suman 8 y 7 en ambas formas (Binario y BCD).

8	1000	1000	
+ 7	+ 0111	+ 0111	
15	1111	1111	→ No es un carácter aceptable en BCD (15 es 0001 0101)

Para realizar operaciones aritméticas con el código BCD se necesitan sumadores especiales. Cuando se desea la propiedad de fácil reconocimiento y la manipulación aritmética, puede utilizarse un código modificado. Esto es sumando en binario el 6 en caso de que el resultado sea >9.

Quedando el ejemplo anterior de la siguiente manera:

1111
+ 0110
0001 0101 → 15_{10}

Completar:

a) 0110 + 0011=

b) 1001 + 1000 =

c) 0011 0110 + 0101 0010=

d) 0001 0100 0111 + 0011 1100 0111 =

Los conceptos anteriores también son aplicables a números decimales con fracciones. Por lo tanto:

a) Facilita la representación de un número decimal,

b) Permite trabajar en forma binaria con un mínimo de espacio.