CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Arquitectura de ordenadores

5. El procesador

David Pinto Fernández
david.pinto@u-tad.com
Carlos M. Vallez Fernández
carlos.vallez@u-tad.com
2022-2023

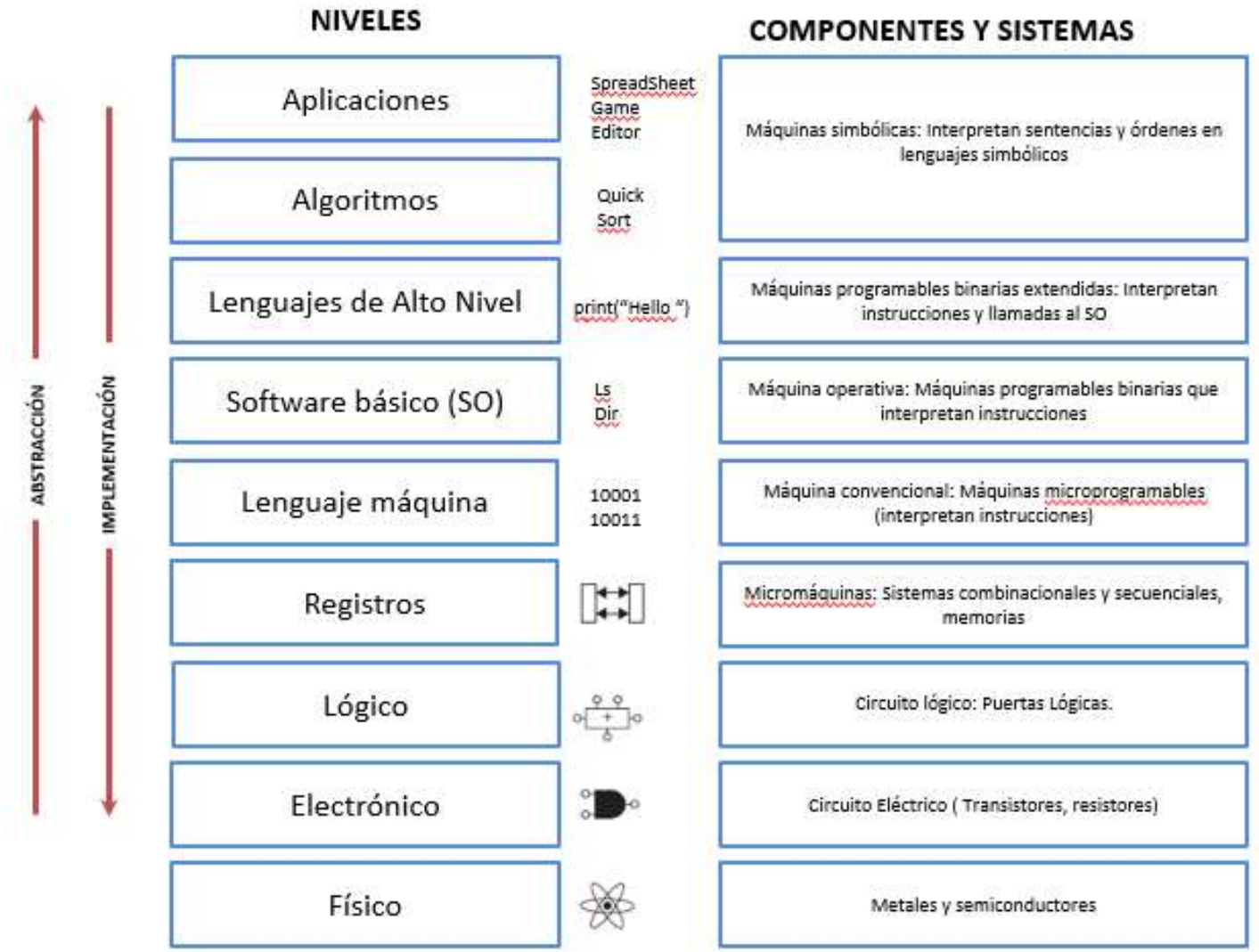
Índice

Motivación (I)

¿Cómo consigue un computador comprender lo que quiere el programador?

Será que se sabe todos los lenguajes…

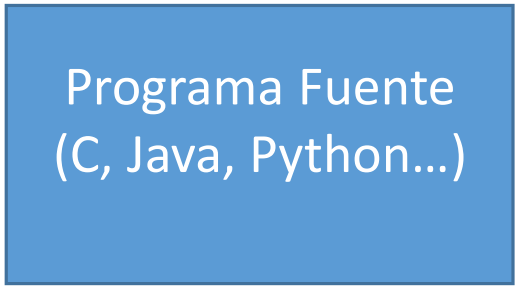
Motivación (II)



Motivación (III)

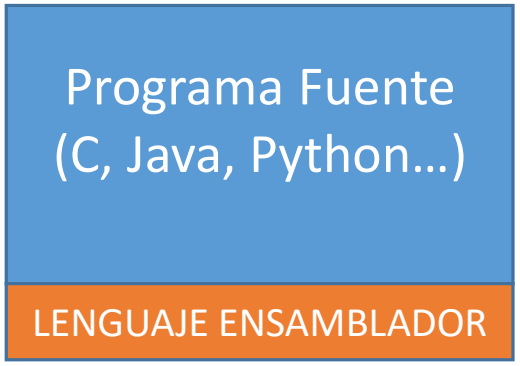
Índice

Lenguaje máquina (I). Definiciones básicas



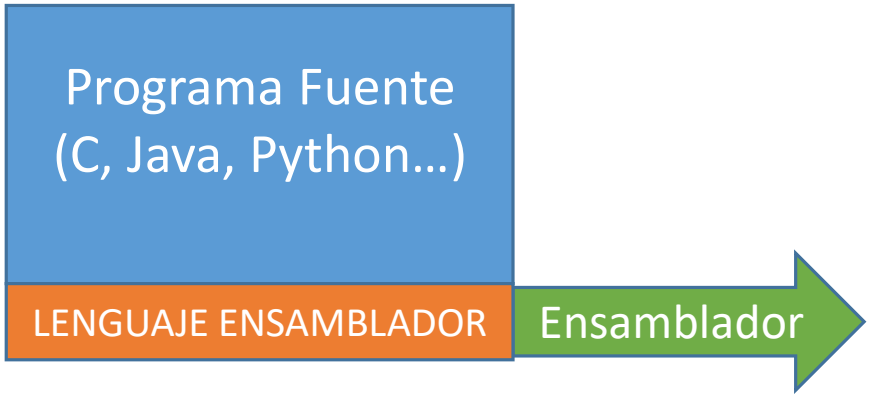
- **Programa fuente:** Es un archivo de texto que contiene el código escrito en un lenguaje de programación, tal como C, Python, Java, etc. El programa fuente no puede ser ejecutado directamente por la computadora, sino que debe ser compilado o interpretado antes de ser ejecutado.

Lenguaje máquina (I). Definiciones básicas



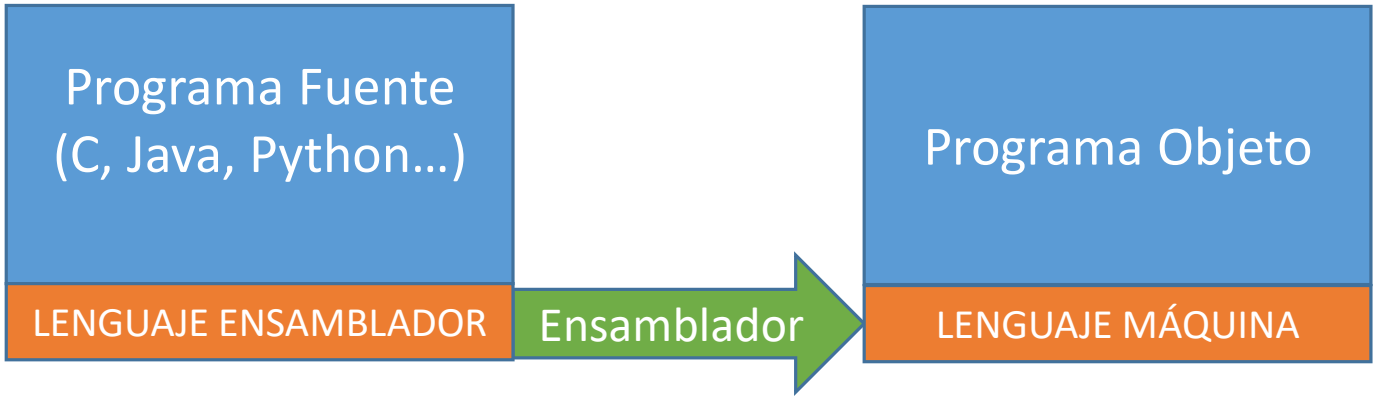
- **Lenguaje Ensamblador:** Es un lenguaje de programación de bajo nivel que permite escribir código que sea fácilmente traducido a lenguaje de máquina. Un ensamblador traduce cada instrucción escrita en lenguaje ensamblador a un código binario que puede ser entendido y ejecutado por la CPU.

Lenguaje máquina (I). Definiciones básicas



- **Ensamblador:** Aplicación encargada de traducir el lenguaje ensamblador al lenguaje máquina

Lenguaje máquina (I). Definiciones básicas



- **Programa objeto:** Es un archivo generado a partir del programa fuente que contiene el código traducido a un formato ejecutable por la CPU. Este archivo es generado por un compilador o un intérprete y contiene instrucciones en lenguaje de máquina o ensamblador. El programa objeto puede ser ejecutado directamente por la computadora.
- **Lenguaje máquina:** Instrucciones que ejecuta el procesador. Son instrucciones nativas del procesador, es decir, que son diferentes por cada familia/fabricante.

Lenguaje máquina (II). Diferencias

- **El lenguaje de alto nivel** es un lenguaje de programación que utiliza una **sintaxis más cercana** al lenguaje humano y se basa en conceptos abstractos, como variables, estructuras de control de flujo y funciones. Ejemplos incluyen Python, Java y C++.

- **El lenguaje ensamblador** es un lenguaje de programación que se utiliza para **programar directamente en lenguaje máquina**. El ensamblador **permite un control más fino de la CPU** y el sistema, pero requiere un conocimiento profundo de la arquitectura de la CPU y de la memoria.

- **El lenguaje máquina** es un lenguaje de programación que se utiliza para **programar directamente en código binario**, en forma de 0s y 1s. Es el **lenguaje más bajo nivel** y no es legible por los humanos, por lo que **se requiere de un ensamblador** o un compilador para escribir programas en él.

Instrucciones máquina (I). Definición

- **Las instrucciones de máquina** son las órdenes que la CPU puede ejecutar directamente.
- Estas instrucciones están **escritas en un código binario** que la CPU puede entender y procesar.
- Las instrucciones de máquina pueden realizar una amplia variedad de tareas, como:
  - Cargar y almacenar valores en la memoria
  - Realizar cálculos y comparaciones
  - Saltar a diferentes secciones del código
  - , etc.

Instrucciones máquina (II). Estructura

- Cada instrucción de máquina **está compuesta por una serie de bits** que representan un **"código op" o "codop"**, que identifica la operación a realizar, y por otros bits que representan los **operandos** necesarios para llevar a cabo la operación. Por ejemplo, para 16 bits tendríamos:

  - El **codop** es el código binario que define la operación a realizar por la CPU.
  - Cada instrucción se divide en los siguientes elementos (ejemplo con 16 bits):
  - Los **operandos** pueden ser de origen y de destino. Pueden estar localizados en:
    - La memoria principal.
    - Los registros.
    - Un dispositivo de E/S.

Instrucciones máquina. Estructura

- Una instrucción de máquina para sumar dos números podría tener la siguiente estructura:

Opcode: 100000

Operando 1: 00101

Operando 2: 11011

- Esta instrucción indica a la CPU que realice una operación de suma utilizando los valores almacenados en los operandos 1 y 2.

Instrucciones máquina. Repertorio

- Los repertorios de instrucciones son **dependientes de la CPU**. Suelen agruparse en los siguientes tipos:
  - **Procesamiento de datos:** operaciones aritméticas y lógicas.
  - **Almacenamiento de datos:** instrucciones de memoria.
  - **Transferencia de datos:** instrucciones de E/S. Teclado, pantalla, altavoz, etc
  - **Control de flujo:** instrucciones condicionales, de bifurcación, de iteración…

Instrucciones máquina. Repertorio

- Cada instrucción tiene asociado su propio codop, así como su propio nemotécnico o **nemónico.** Por ejemplo:

Codop	Nemónico	Operación
0001	ADD	Sumar
0010	SUB	Restar
0011	LD	Cargar

- El **número y tipo de instrucciones** que soporta una CPU puede **variar dependiendo del tipo de CPU**.
  - Las CPUs modernas suelen tener un repertorio de instrucciones más amplio y complejo.
- El repertorio de instrucciones de una CPU **es una parte fundamental de su arquitectura**, ya que **influye en la velocidad y eficiencia** con la que la CPU puede ejecutar tareas, y también en la capacidad de la CPU para ejecutar diferentes tipos de aplicaciones y sistemas informáticos.

Instrucciones Máquina. Operandos

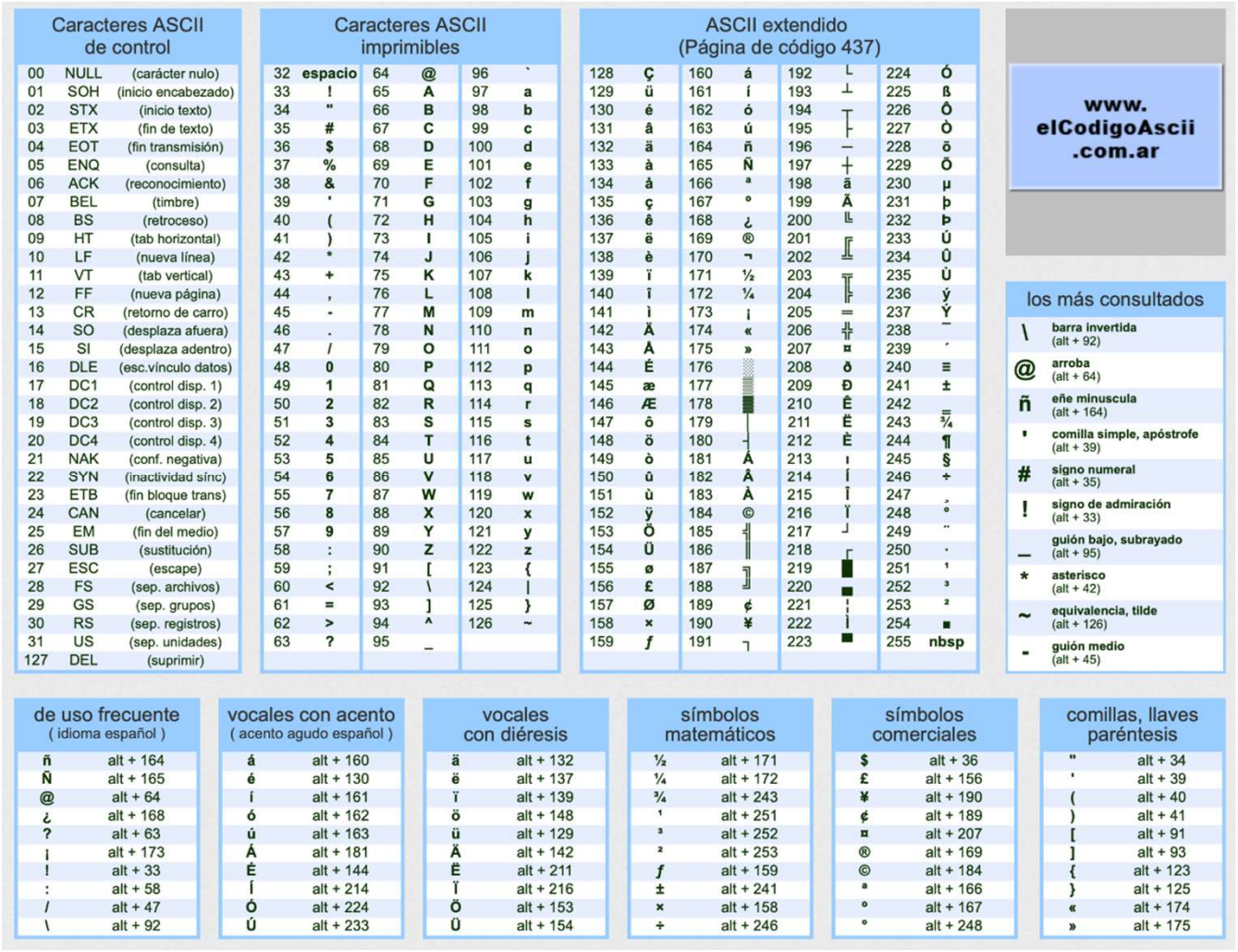
- Los operandos son los valores o direcciones de memoria que se utilizan en las operaciones realizadas por las instrucciones de máquina.
- Los operandos pueden ser valores constantes almacenados en el código de la instrucción, o bien pueden ser direcciones de memoria donde se almacenan los valores.
- Cada instrucción de máquina puede tener uno o más operandos, dependiendo de la naturaleza de la operación que se realiza. Los más frecuentes son:
  - **Direcciones de memoria**: Reflejan una dirección en memoria donde, por ejemplo, está almacenado un dato de interés (8800H). Están contenidas entre paréntesis y las direcciones se suele expresar en hexadecimal.
  - **Números**: Como parte del cómputo a realizar o de una estructura de control. Aunque la ALU trabaja en binario (011B), se pueden ver representados en hexadecimal (34H, 0A34H) y decimal (34 ó 34D).
  - **Caracteres**: Utilizados en secuencia para representar texto. Para representar caracteres, se establece una codificación que relaciona un código binario determinado con un único carácter: ASCII

Instrucciones Máquina. ASCII

- **American Standard Code for Information Interchange**
- Define un **conjunto de caracteres y símbolos que se pueden utilizar para representar texto,** incluyendo letras mayúsculas y minúsculas, números, signos de puntuación y caracteres de control como el salto de línea y la tabulación.
- Cada **carácter** está asociado con un **número binario de 7 bits**.
- Es un estándar ampliamente utilizado.
- Muchos otros códigos de representación de caracteres, como Unicode, han evolucionado a partir de ASCII.

Instrucciones Máquina. ASCII

- https://elcodigoascii.com.ar



Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	▒	208	ð	240	≡
145	æ	177	▓	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	▉	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

www.elCodigoAscii.com.ar

los más consultados

\	barra invertida	(alt + 92)
@	arroba	(alt + 64)
ñ	eñe minuscula	(alt + 164)
'	comilla simple, apóstrofe	(alt + 39)
#	signo numeral	(alt + 35)
!	signo de admiración	(alt + 33)
_	guión bajo, subrayado	(alt + 95)
*	asterisco	(alt + 42)
~	equivalencia, tilde	(alt + 126)
-	guión medio	(alt + 45)

de uso frecuente (idioma español)

ñ	alt + 164
Ñ	alt + 165
@	alt + 64
¿	alt + 168
?	alt + 63
¡	alt + 173
!	alt + 33
:	alt + 58
/	alt + 47
\	alt + 92

vocales con acento (acento agudo español)

á	alt + 160
é	alt + 130
í	alt + 161
ó	alt + 162
ú	alt + 163
Á	alt + 181
É	alt + 144
Í	alt + 214
Ó	alt + 224
Ú	alt + 233

vocales con diéresis

ä	alt + 132
ë	alt + 137
ï	alt + 139
ö	alt + 148
ü	alt + 129
Ä	alt + 142
Ë	alt + 211
Ï	alt + 216
Ö	alt + 153
Ü	alt + 154

símbolos matemáticos

½	alt + 171
¼	alt + 172
¾	alt + 243
¹	alt + 251
³	alt + 252
²	alt + 253
ƒ	alt + 159
±	alt + 241
×	alt + 158
÷	alt + 246

símbolos comerciales

$	alt + 36
£	alt + 156
¥	alt + 190
¢	alt + 189
¤	alt + 207
®	alt + 169
©	alt + 184
ª	alt + 166
º	alt + 167
°	alt + 248

comillas, llaves paréntesis

"	alt + 34
'	alt + 39
(	alt + 40
)	alt + 41
[	alt + 91
]	alt + 93
{	alt + 123
}	alt + 125
«	alt + 174
»	alt + 175

Índice

El micro Z80

- El microprocesador **Zilog Z80** (1976), se usó en la década de 1980 en computadores como el Sinclair **Spectrum** (izquierda) y el **Amstrad** CPC 64 (derecha). Compatible a nivel de código máquina con el Intel 8080, con ciertas mejoras: más registros, más instrucciones, menor coste…

- El Z80 ha sido **utilizado en una amplia variedad de sistemas**, ordenadores, consolas, instrumentos musicales electrónicos y equipos industriales. A pesar de que **ha sido reemplazado por microprocesadores más avanzados** y potentes, **todavía es una pieza de hardware de culto**.

El micro Z80

- Es un procesador con arquitectura de 8 bits (1 byte u octeto).
- Los microprocesadores modernos son de 32 o 64 bits. Los microprocesadores modernos ofrecen una gran cantidad de ventajas comparadas con el Z80:
  - Mayor velocidad, mayor capacidad de procesamiento, mayor memoria (acceso a cantidades mayores de memoria RAM y memoria flash) y mayor compatibilidad con periféricos.

Entonces, ¿Por qué aprendemos a trabajar con el Z80 y no con uno más moderno?

El micro Z80. Registros

- Un **registro** es una pequeña **área de memoria interna** en un procesador donde se almacenan valores temporales y se realizan operaciones.
- Los registros **son mucho más rápidos que la memoria RAM** externa, por lo que el procesador los utiliza para almacenar y manipular los datos más críticos.
- Los registros son un **elemento clave en la arquitectura** de un procesador.
- Cada procesador tiene un número diferente de registros dependiendo de su arquitectura, cada uno con **diferentes usos y propósitos**.
- Cada registro tiene una función específica y su **contenido puede ser leído y escrito** por el procesador **en una operación de un ciclo de reloj**.
- El Z80 tiene **16 registros de 8 bits** (1 byte) y **4 registros de 16 bits** (2 bytes). A su vez, estos registros se dividen en dos grupos: **12 registros de propósito general** y **8 dedicados** (4 de 16 bytes).

El micro Z80. R. propósito general.

- Los registros de **propósito general sirven para almacenar datos** sobre los que operar sin tener que referenciar a la memoria. Están divididos en **dos juegos**, **normales** y **'**. Son:
  - **B, C, D, E, H, L, B', C', D', E', H' y L'**.
  - El procesador **sólo puede acceder a uno de los juegos al mismo tiempo**.
- Son todos de **8 bits**, aunque **se pueden emparejar para trabajar con datos de 16 bits** de la siguiente forma:
  - BC, DE, HL, B'C', D'E' y H'L'.
- **Existe una instrucción específica (EXX) para cambiar el juego** de registros con el que se trabaja en un instante dado.

El micro Z80. Registros dedicados.

- Los registros dedicados son: **A, A', F, F', PC, IX, IY y SP.**
  - **A, A', F y F':**
    - Todos de 8 bits.
    - A es el registro **acumulador**.
    - Se utiliza para la mayoría de instrucciones aritméticas, lógicas y de carga.
    - El registro de indicadores o F (**Flag**), está asociado a A. Su valor cambia dependiendo de la última operación realizada sobre A, indicando datos interesantes sobre la misma: overflow, carry, cero, …
    - A y F jamás pueden asociarse para trabajar con datos de 16 bits.
    - A' y F' son los registros correspondientes a A y F en el juego de registros '.

Ensamblador Z80. Registro F - Flags

- El **registro F contiene 8 indicadores binarios** (8 bits), también llamados flags, que **actúan como alertas** que proporcionan **información adicional sobre ciertas operaciones** realizadas por el procesador.
- Al contrario que el resto de registros, el registro F **no se lee como un todo**, sino que **cada indicador provee información por separado**.
- Su estructura es la siguiente:

F	7	6	5	4	3	2	1	0
	S	Z	-	H	-	P/V	N	C

Ensamblador Z80. Flags

F	7	6	5	4	3	2	1	0
	S	Z	-	H	-	P/V	N	C

- **Bit C:** Es el indicador de **acarreo**.
  - Por defecto está a 0. Se puede poner a 1 como consecuencia de operaciones aritméticas, de desplazamiento y de comparación.
- **Bit N:** Indica si la última operación efectuada ha sido una **resta (1).**
- **Bit P/V:**
  - Tras una operación aritmética, indica si ha habido **overflow (V = 1).**
  - Tras una operación lógica o de desplazamiento, indica la **paridad de unos** en el byte (P = 1).

Ensamblador Z80. Flags

F	7	6	5	4	3	2	1	0
	S	Z	-	H	-	P/V	N	C

- **Bit H:** Es el **indicador de acarreo de nivel 2**.
  - Es decir, funciona como C, pero indica si hay acarreo en los cuatro primeros bits del byte resultante de la operación.
- **Bit Z:** Indica si el resultado de la última **operación realizada es nulo.** En ese caso, Z = 1.
- **Bit S:**
  - Es el **bit de signo del número almacenado en el acumulador** (registro A)
  - Toma el valor del octavo bit del byte almacenado en A.
- **Bits 5 y 3: No se usan.**

El micro Z80. Registros dedicados.

- **PC:**
  - Es el contador de programa.
  - Es de 16 bits, porque se relaciona con las direcciones de memoria.
  - Contiene la dirección en memoria (16 bits) de la siguiente instrucción a ejecutar.
  - Su contenido se modifica automáticamente cuando se termina de ejecutar una instrucción, aunque también se puede modificar manualmente.

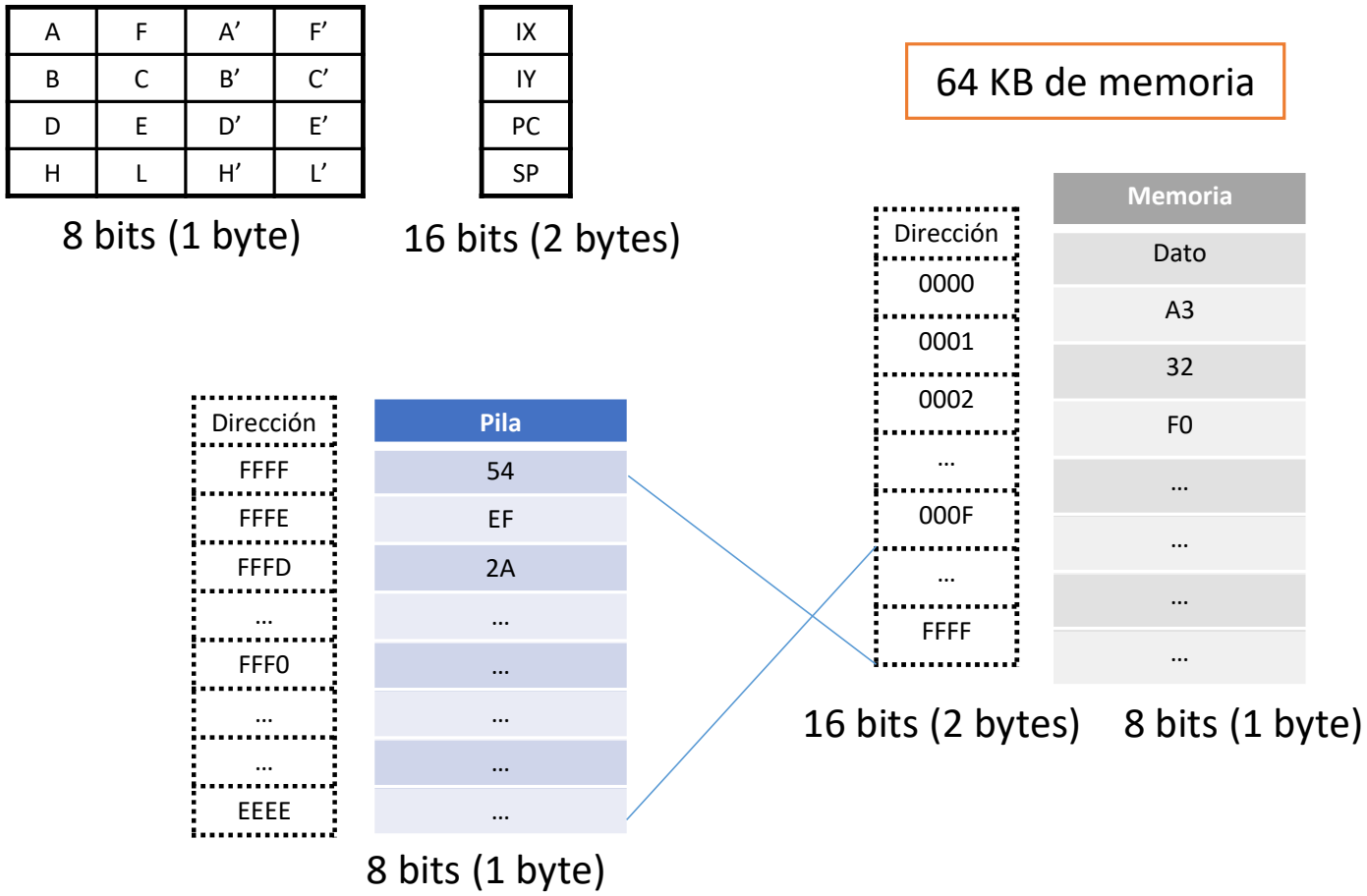
El micro Z80. Registros dedicados.

- **IX e IY:**
  - Son registros de 16 bits
  - Se usan habitualmente para operaciones con memoria
  - Sirven como apuntadores o índices a direcciones de memoria, frecuentemente almacenes de datos.
  - En programación de alto nivel, tienen un símil en los clásicos contadores i y j.

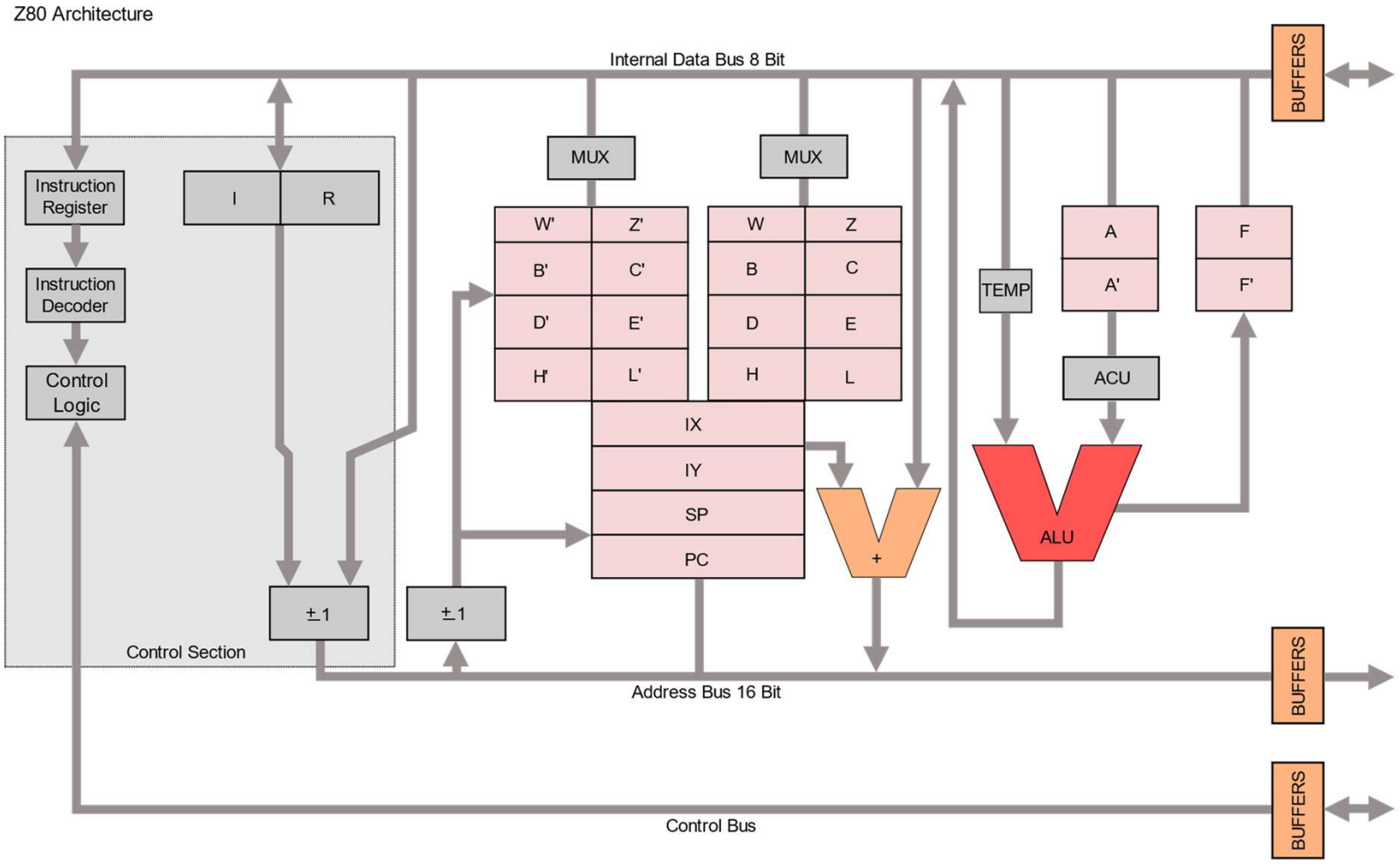
El micro Z80. Pila.

- Una **pila** es una estructura de datos **LIFO** (*Last Input First Output*):
  - Cuando se introduce un elemento nuevo, se pone en la cima de la pila.
  - Cuando se saca un elemento, siempre se saca el que ocupa la cima de la pila.
- Físicamente, la pila es una parte de la memoria principal, pero la diferencia es que no se puede acceder a cualquier dirección de la pila en cualquier instante.
  - Únicamente se puede acceder al lugar donde apunta el registro SP.
- En el Z80, la pila es una serie de celdas contiguas de 8 bits de ancho con dirección propia, al igual que la memoria.
- El registro **SP** contiene la dirección del último dato almacenado de la pila:
  - Cuando se **almacena** un dato en la pila, se decrementa el valor del SP en 1 y se escribe en la dirección a la que apunta ahora.
  - Cuando se **lee** un dato de la pila, se lee la dirección a la que apunta el SP y se incrementa su valor en 1.

El micro Z80. Registros - Memoria - Pila.



El micro Z80


Z80 Architecture
Internal Data Bus 8 Bit
BUFFERS
Instruction Register
Instruction Decoder
Control Logic
I
R
MUX
MUX
W'
Z'
B'
C'
D'
E'
H'
L'
W
Z
B
C
D
E
H
L
IX
IY
SP
PC
TEMP
A
A'
F
F'
ACU
ALU
+
±1
±1
Control Section
Address Bus 16 Bit
Control Bus

Índice

Ensamblador Z80. Estructura Instrucciones

- Las instrucciones en el ensamblador del Z80 tienen la siguiente estructura:

- Donde:
  - **Etiqueta:** Una nombre único que identifica de manera unívoca una línea del programa. Es opcional.
  - **Operación:** Acoge el nemónico de la operación ensamblador que se desea realizar.
  - **Operando(s):** Corresponde al operando de la operación. Puede ser simple o doble.
    - Si es doble, el primer operando es el destino y el segundo es el origen.
  - **Comentarios:** Debe comenzar por el símbolo ‘;’. Es un espacio para incluir comentarios sobre el programa. Es opcional.

Ensamblador Z80. Ejemplo

- Los **operandos pueden ser registros, direcciones de memoria, valores inmediatos, etiquetas,** etc., dependiendo del tipo de instrucción.
- Algunos ejemplos de instrucciones en ensamblador Z80 son:
  - LD A, 42 ; carga el valor 42 en el acumulador
  - LD B, C ; copia el contenido del registro C en el registro B
  - ADD A, B ; suma el contenido de A y B, almacenando el resultado en A
  - JP 1234h ; salta a la dirección de memoria 1234h
  - CALL mi_subrutina ; llama a la subrutina identificada por la etiqueta "mi_subrutina”
- Ya veremos más adelante con detenimiento cada una de estas instrucciones…

Ensamblador Z80. Estructura Programa

Directiva de comienzo: Indica en qué dirección de memoria está la primera instrucción del programa.

```
                org 200H

program         ld A,37H
                ld (210H),A
                ld C,A

                end program
```

Instrucciones del programa

Etiqueta de inicio: Marca la dirección de memoria del inicio.

Directiva de fin: Indica que el programa se ha acabado.

Ensamblador Z80. Representación.

- **Los números se pueden representar en diferentes bases** en el ensamblador. Algunas instrucciones admiten diferentes bases, otros son dependientes de una base concreta.
  - En **decimal**: 34D o, símplemente, 34. (Ojo 34D no funciona en ZEUS)
  - En **hexadecimal**: 34H o, si comienza por letra, 0A4H
  - En **binario**: 00110100B

```
          org 200H

Program   ld A,37H
          ld (210H),A ;source only A
          ld C,A

          end program
```

Ensamblador Z80. LD

- La instrucción que más utilizaremos en nuestros programas en ensamblador será sin duda la **operación de carga o instrucción LD**. Sirve para:
  - Meter un valor en un registro.
  - Copiar el valor de un registro en otro registro.
  - Escribir en memoria (en una dirección determinada) un valor.
  - Escribir en memoria (en una dirección determinada) el contenido de un registro.
  - Asignarle a un registro el contenido de una dirección de memoria.
- La **sintaxis de LD** en lenguaje ensamblador es:

LD DESTINO, ORIGEN

- La instrucción **LD** copia el valor contenido en el operando origen en el operando destino.
- Esta operación no es aritmética, ni lógica ni de desplazamiento, por lo que no afecta a ningún *flag* (F).

Ensamblador Z80. Direccionamiento

- **Direccionamiento** es la manera en la que se indica a una instrucción dónde están los operandos sobre los que debe operar.
- El ensamblador **soporta diferentes modos de direccionamiento** para acceder a datos y operar con ellos. Los principales modos son:
  - Direccionamiento **extendido**.
  - Direccionamiento **indirecto.**
  - Direccionamiento **indexado.**
  - Direccionamiento **a través de registro.**
  - Direccionamiento **implícito.**
  - Direccionamiento **inmediato.**
  - Direccionamiento **inmediato extendido.**