



UTPL
La Universidad Católica de Loja

Modalidad Abierta y a Distancia

Arquitectura de Computadoras y Sistemas Operativos

Guía didáctica



Índice

**Primer
bimestre**

**Segundo
bimestre**

Solucionario

**Referencias
bibliográficas**



Departamento de Ciencias de la Computación y Electrónica

Sección departamental de Electrónica y Telecomunicaciones

Arquitectura de Computadoras y Sistemas Operativos

Guía didáctica

Autores:

Castillo Calvas Tuesman Daniel
Cueva Carrión Samanta Patricia



D S O F _ 2 0 3 9

Asesoría virtual
www.utpl.edu.ec

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

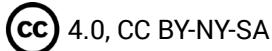
Arquitectura de Computadoras y Sistemas Operativos

Guía didáctica

Castillo Calvas Tuesman Daniel

Cueva Carrión Samanta Patricia

Universidad Técnica Particular de Loja



Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojainfo@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-25-634-8



La versión digital ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite: copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

22 de abril, 2020

1. Datos de información.....	9
1.1. Presentación de la asignatura.....	9
1.2. Competencias genéricas de la UTPL	9
1.3. Competencias específicas de la carrera.....	9
1.4. Problemática que aborda la asignatura en el marco del proyecto	10
1.5. Proyecto integrador de saberes	10
2. Metodología de aprendizaje.....	11
3. Orientaciones didácticas por resultados de aprendizaje.....	12
 Primer bimestre	 12
Resultados de aprendizaje 1 y 2	12
Contenidos, recursos y actividades de aprendizaje	12
 Semana 1	 12
 Unidad 1. Introducción a la arquitectura de computadoras.....	 13
1.1. Introducción.....	13
1.2. Organización y arquitectura de Computadoras.....	14
1.3. Funcionamiento de una computadora	15
1.4. Representación de información en una computadora.....	17
1.5. Funcionamiento básico de una computadora	19
Actividades de aprendizaje recomendadas	21
Autoevaluación 1	26
Resultados de aprendizaje 3 y 4	29
Contenidos, recursos y actividades de aprendizaje	29
 Semana 2	 29
 Unidad 2. Unidad Aritmético-Lógica	 29
2.1. Descripción del hardware de una ALU	31
2.2. Circuitos aritméticos	33
Actividades de aprendizaje recomendadas	39

Autoevaluación 2	43
Resultados de aprendizaje 5 y 6	45
Contenidos, recursos y actividades recomendadas.....	45
Semana 3	45
Unidad 3. Sistema de memoria	45
3.1. Concepto de memoria.....	46
3.2. Funciones de correspondencia	49
3.3. Algoritmos de sustitución.....	56
3.4. Memoria principal.....	57
Actividades de aprendizaje recomendadas	66
Autoevaluación 3	68
Resultados de aprendizaje 7 y 8	71
Contenidos, recursos y actividades recomendadas.....	71
Semana 4	71
Unidad 4. Set de instrucciones y direccionamiento	71
4.1. Modos de direccionamiento	71
4.2. Instrucciones máquina.....	76
Actividades de aprendizaje recomendadas	88
Autoevaluación 4	92
Resultados de aprendizaje 9 y 10	94
Contenidos, recursos y actividades recomendadas.....	94
Semana 5	94
Unidad 5. Unidad Central de Procesamiento	94
5.1. Organización de los registros	97
5.2. Proceso de ejecución de un programa	98
5.3. Gestión de interrupciones.....	101
5.4. Unidad de control	102
Actividades de aprendizaje recomendadas	108
Autoevaluación 5	109
Resultados de aprendizaje 11 y 12	111
Contenidos, recursos y actividades recomendadas.....	111

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Índice	
Semana 6	111
Unidad 6. Módulos de entrada y salida.....	111
6.1. Función del módulo de E/S.....	112
6.2. Técnicas de entrada y salida	115
6.3. Acceso directo a memoria.....	122
6.4. Buses.....	126
6.5. El bus PCI	130
Actividades de aprendizaje recomendadas	134
Autoevaluación 6	135
Actividades finales del bimestre.....	137
Resultados de aprendizaje 13 y 14.....	137
Contenidos, recursos y actividades recomendadas.....	137
Semana 7	137
Actividades de aprendizaje recomendadas	138
Semana 8	138
Segundo bimestre	139
Resultados de aprendizaje 15 y 16	139
Contenidos, recursos y actividades recomendadas.....	139
Semana 9	139
Unidad 7. Principios de los sistemas operativos	139
7.1. Sistema operativo.....	140
7.2. Generaciones de los sistemas operativos.....	142
7.3. Estructura de los sistemas operativos.....	147
7.4. Tendencias de los sistemas operativos.....	153
Actividades de aprendizaje recomendadas	155
Autoevaluación 7	157
Resultados de aprendizaje 17 y 18	160
Contenidos, recursos y actividades recomendadas.....	160

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Índice	
Primer bimestre	
Segundo bimestre	
Solucionario	
Referencias bibliográficas	
Semana 10	160
 Unidad 8. Planificación de procesos	160
8.1. Introducción a Procesos	161
8.2. Planificación de la CPU	168
Actividad de aprendizaje recomendada	177
Autoevaluación 8	178
Semana 11	181
 Unidad 9. Sincronización de Procesos	181
9.1. Introducción.....	181
9.2. Problemas de la sección crítica	182
9.3. Hardware de sincronización	184
9.4. Semáforos.....	185
9.5. Problemas clásicos de sincronización	186
9.6. Monitores	190
Actividades de aprendizaje recomendadas	191
Autoevaluación 9	193
Semana 12	197
 Unidad 10. Interbloqueo de Procesos	197
10.1.Fundamentos de interbloqueo.....	197
10.2.Condiciones para que se cumpla el interbloqueo	199
10.3.Grafos de asignación de recursos.....	200
10.4.Previsión de interbloqueos	203
10.5.Predicción de interbloqueos	204
10.6.Detección y Recuperación de Interbloqueos	204
Actividades de aprendizaje recomendadas	207
Autoevaluación 10	209
Resultado de aprendizaje 19	212
Contenidos, recursos y actividades recomendadas.....	212
Semana 13	212
 Unidad 11. Gestión de memoria.....	212

Índice	
Primer bimestre	
Segundo bimestre	
Solucionario	
Referencias bibliográficas	
11.1. Antecedentes.....	212
11.2. Técnicas de gestión de memoria	217
Actividades de aprendizaje recomendadas	235
Autoevaluación 11	236
Resultado de aprendizaje	
20 y 21	239
Contenidos, recursos y actividades recomendadas.....	239
Semana 14	239
Unidad 12. Sistemas de archivos.....	239
12.1. Introducción al sistema de archivos	239
12.2. Métodos de acceso.....	242
12.3. Estructura del sistema de archivos	242
12.4. Estructura de los directorios.....	243
12.5. Métodos de asignación.....	246
Actividades de aprendizaje recomendadas	250
Autoevaluación 12	251
Semana 15	254
Unidad 13. Gestión de dispositivos de E/S.....	254
13.1. Fundamentos de gestión de dispositivos	254
13.2. Tipos de dispositivos de E/S	255
13.3. Acceso directo a memoria (DMA)	257
13.4. Subsistema de E/S del kernel	257
Actividades de aprendizaje recomendadas	260
Autoevaluación 13	261
Actividades finales del bimestre:.....	264
Semana 16	264
4. Solucionario	265
5. Referencias bibliográficas	283

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



1. Datos de información

1.1. Presentación de la asignatura.



1.2. Competencias genéricas de la UTPL

- Comunicación en inglés

1.3. Competencias específicas de la carrera

- Administrar los servicios de tecnologías de información de la organización utilizando buenas prácticas de la industria asegurando la continuidad operacional del negocio.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

1.4. Problemática que aborda la asignatura en el marco del proyecto

- Comprender la arquitectura de computadores, basándose en los principios de los componentes básicos que los conforman para determinar la más adecuada a las necesidades organizacionales.
- Aplicar los conocimientos de procesos, manejo de memoria y sistema de archivos en contextos prácticos para garantizar la fiabilidad de los sistemas operativos.

1.5. Proyecto integrador de saberes

Analizar los esquemas de operación y funcionamiento de TI en empresas de su entorno.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas



2. Metodología de aprendizaje

Autoaprendizaje:

Descripción de la metodología:

Se busca que el estudiante se convierta en un elemento activo en el proceso de aprendizaje, guiado por el docente, busca, lee y construye su aprendizaje mediante los recursos disponibles: Texto guía, textos de referencia, biblioteca, recursos educativos abiertos, etc.

Para más información sobre la metodología a emplear puede leer el artículo: “*Un modelo de autoaprendizaje con integración de las TIC y los métodos de gestión del conocimiento*” disponible en:

[**Un modelo de autoaprendizaje con integración de las TIC y los métodos de gestión del conocimiento.**](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultados de aprendizaje 1 y 2

- Describir las principales características de la arquitectura de computadores.
- Explicar el funcionamiento de la máquina de Von Neumann comparándolo con los procesadores actuales.

Contenidos, recursos y actividades de aprendizaje



Semana 1



Unidad 1. Introducción a la arquitectura de computadoras

1.1. Introducción

En la actualidad dentro de esta categoría recaen desde microprocesadores hasta supercomputadoras con varios núcleos; sin embargo, las increíbles variedades de sistemas computacionales guardan una relación en su Arquitectura que ha sobrevivido décadas, desde 1945 cuando se describió “la arquitectura de Von Neumann”.

Para comprender el término Arquitectura de computadoras, se puede hacer una analogía con la arquitectura convencional. En la figura 1, se muestra esta analogía, observe que el arquitecto de computadoras debe considerar los materiales disponibles y el objetivo para lograr un buen diseño que lleve a un producto final.

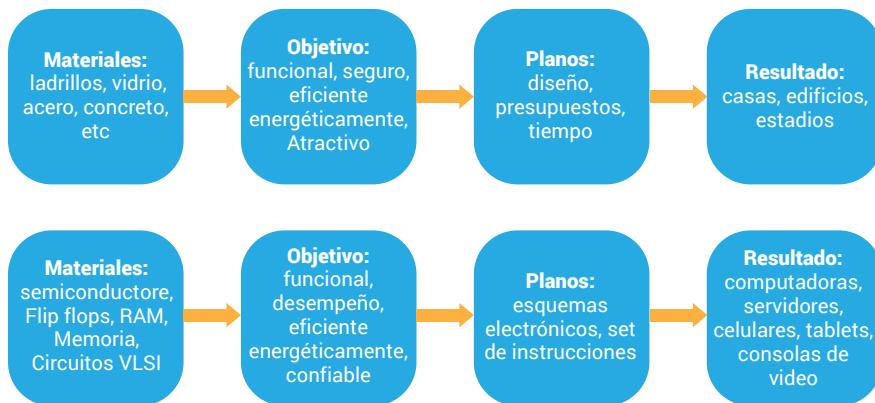


Figura 1. Arquitectura de Computadoras

Fuente: Elaboración propia

En la arquitectura de computadoras el diseñador debe balancear la funcionalidad con otros factores importantes como costo de producción, consumo energético (especialmente dispositivos portables). Los dispositivos diseñados deben ser confiables, es decir mantener su buen funcionamiento durante todo el tiempo de vida del producto.

La arquitectura de computadoras no consiste tan solo en diseñar computadoras para estaciones de trabajo. Actualmente se buscan mejores y eficientes diseños para:

- Tecnología móvil: laptops, celulares, tabletas. Necesitan consumir poca energía, alto desempeño, incluir chips de comunicaciones inalámbricas. (Intel Atom, AMD Turion, procesadores ARM, etcétera).
- Tecnología embebida: microcontroladores que generalmente encontrados en el control de máquinas industriales, vehículos, y redes de sensores, necesitan consumir poca energía, tener bajo costo, aquí también destacan los procesadores basados en ARM, y los Procesadores de Señales Digitales (DSP).

1.2. Organización y arquitectura de Computadoras

Existe cierta variedad en la definición de los términos “organización” y “arquitectura” de computadoras. En la literatura los autores han intentado acercarnos a una definición inequívoca que ayude a diferenciarlos. Stallings (2007) se refiere a “arquitectura de computadoras” como: “Los parámetros que son visibles al programador” es decir, todos aquellos parámetros que tienen un directo impacto en la ejecución lógica de un programa. Ejemplos de arquitectura son: el set de instrucciones, el número de bits usados para representar datos, técnicas de acceso a memoria, métodos de entrada y salida.

Se define, entonces, el término “organización” como: “Los parámetros que son transparentes al programador”, es decir, las unidades operacionales y las interconexiones que obedecen a las especificaciones arquitectónicas. Por ejemplo: señales de control, tecnología de memoria RAM usada, interfaces entre la computadora y periféricos.

1.3. Funcionamiento de una computadora

1.3.1. Computadora de uso general

Una computadora de uso general puede ser descrita por cuatro unidades básicas, se debe aclarar que este esquema es muy general y cada bloque puede abarcar muchos subsistemas y bloques funcionales. En la figura 2, se muestran los cuatro bloques principales de una computadora que consiste en la memoria principal, unidad de control, unidad aritmético-lógica y las interfaces de entrada y salida. Este diseño fue propuesto por John Von Neumann y se implementó en la primera computadora digital llamado “IAS machine”, por sus siglas en inglés: Institute for Advanced Study en Princeton, EUA.

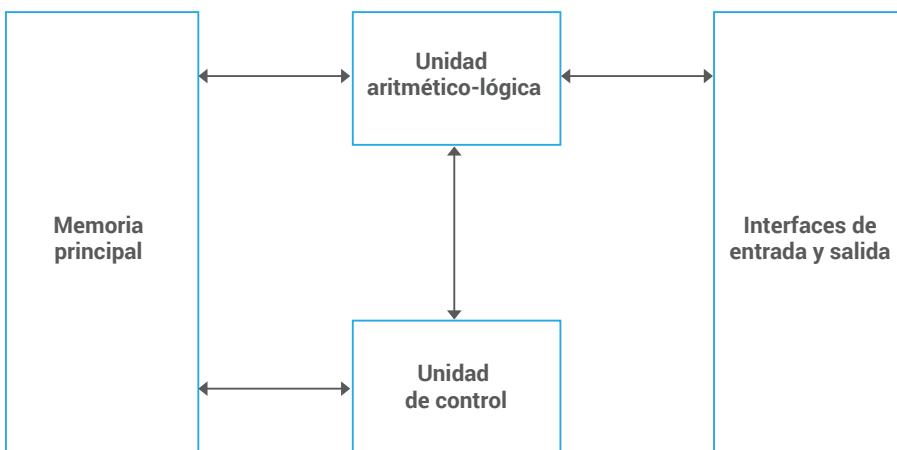


Figura 2. Estructura de una computadora de uso general.

Fuente: Elaboración propia.

La Unidad Central de procesamiento está compuesta por los bloques:

- **Unidad aritmético-lógica (ALU):** este bloque está encargado de recibir datos en sus registros y ejecutar operaciones lógicas y aritméticas.
- **Unidad de control:** contiene un set de instrucciones y circuitos electrónicos encargados de decodificar instrucciones provenientes desde la memoria, y generar señales de control para cada componente de la computadora.
- **Las interfaces de entrada y salida:** datos provenientes desde el exterior (teclado, mouse, CD-ROM, memorias portables, puerto de red, etc.) deben ser almacenadas en la memoria de la computadora mediante las señales de control de la unidad de control. De la misma manera los datos almacenados o resultados generados pueden ser mostrados al usuario o ser enviados a una red de computadoras por medio de los dispositivos de salida (Impresora, monitor, memorias portables, tarjeta de red).
- **Unidad de memoria:** es usada para almacenar datos y programas. La unidad de control y la ALU deben trabajar con la información guardada en esta unidad. Esta memoria en la actualidad se divide en distintos niveles de jerarquía según la velocidad de acceso a ella:
- **Memoria principal:** generalmente está compuesta de dispositivos semiconductores (transistores, FLIP-FLOPS) a ello debe su nombre el término “Memoria semiconductora”: RAM (*Random Access Memory*), ROM (*Read Only Memory*), PROM (*Programable ROM*). EPROM (*Erasable PROM*), EEPROM (*Electrically Erasable PROM*).

- **Memoria secundaria:** la memoria secundaria se caracteriza por no ser volátil y se usa para almacenar gran cantidad de datos; sin embargo, la velocidad de acceso a ella desde la CPU es mucho más lento en contraste con la memoria principal, entre algunos ejemplos: disco duro, CD-ROM, Pen Drive.

1.4. Representación de información en una computadora

Los componentes funcionales de una computadora están conformados por circuitos electrónicos que trabajan con señales eléctricas. Existen dos tipos de señales, analógicas y digitales. Las señales analógicas son de naturaleza continua, es decir tiene una variación continua en el tiempo y las señales digitales son discretas, es decir toman valores discretos en espacios discontinuos en el tiempo. Los dispositivos que trabajan con señales continuas se conocen como Dispositivos Analógicos y los circuitos que manejan señales discretas se conocen como dispositivos digitales. En la actualidad, mayoría de computadores son de naturaleza digital.

Una computadora digital trabaja con dos estados de voltaje, ALTO y BAJO. Según el estándar, estos estados pueden ser 0V y +5V, 0V y +3.3V o 0V y +12V. En el ámbito del diseño de una computadora no se considera estos estados de voltaje, en lugar, trabajaremos con los estados lógicos que representan "0" y "1". La unidad de información más básica es el "*bit*", que puede tomar solo dos valores "1" o "0". Cuatro *bits* conforman un "*nible*", ocho *bits* forman un "*byte*".

Todos los estados de una computadora, incluida la información almacenada, recibida o generada, se expresan en formato binario. Así, los caracteres ingresados a través del teclado se "codifican" en ASCII en formato de 7 bits, como se muestra en la figura 3. En cambio, las instrucciones que se almacenan en la memoria se codifican en el formato que considere el diseñador para el "set de

instrucciones”, un ejemplo de instrucciones y su codificación se puede observar en la figura 4.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELLI]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	I	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figura 3. Código ASCII estándar.

Tomado de: <https://easytechnow.com/learn-technology/what-is-ascii-code/>

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

COD OP	Intrucción	Descripción
00001010	LOAD MQ	Transferir el contenido del registro MQ al Acumulador AC
00001001	LOAD MQ,M(X)	Transferir el contenido de la posición X de memoria a MQ
00100001	STOR M(X)	Trasferir el contenido de AC a la posición de memoria X
00000001	LOADM(X)	AC<- M (X)
00000010	LOAD-M(X)	Transferir el complemento de M(X) a AC
00000100	LOAD M(X)	Transferir M(X) a AC
00000101	ADD M(X)	AC<-AC+M(X)
00000111	ADD M(X)	AC<-AC+ M(X)
00000110	SU B M(X)	AC<-AC-M(X)
00001000	SU B M(X)	AC<-AC M(X)
00001011	MUL M (X)	[AC][MQ]<-ACxM(X)
00001100	DIV M(X)	[AC][MQ]<-AC/M(X)
00010100	LSH	AC<-ACx2
00010101	RSH	AC<-AC/2
00001111	JUMP+M(X,B:19)	Si AC es mayor o igual a 0 saltar a la instrucción de la izquierda de M(X)
00001000	JUMP+M(x,28:39)	Si AC es mayor o igual a 0 saltar a la instrucción de la derecha de M(X)
00001101	JUMP M(x,8:19)	Saltar a la instrucción de la izquierda en M(X)
00001110	JUMP M(x,28:39)	Saltar a la instrucción de la derecha en M(X)
00010010	STOR M(x,8:19)	Reemplaza el campo de dirección de la izquierda de M(X) por los 12 bits de la derecha contenidos en el AC
00010011	STOR M(x,28:39)	Reemplaza el campo de dirección de la derecha de M(X) por los 12 bits de la derecha contenidos en el AC

Figura 4. Conjunto de instrucciones IAS.

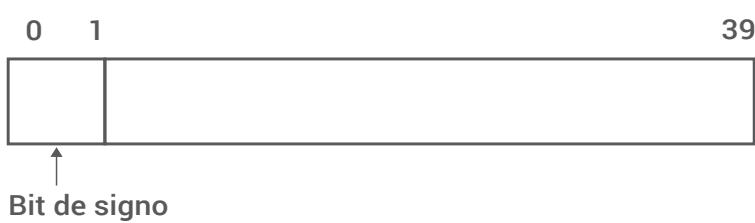
Adaptado de Stallings, W. 2007

1.5. Funcionamiento básico de una computadora

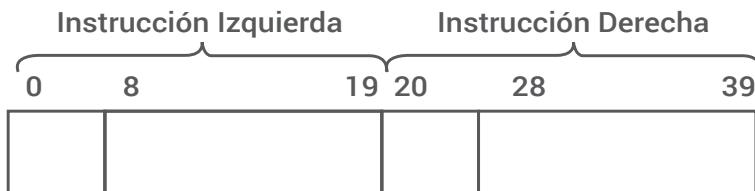
En esta sección se explicará el principio básico del funcionamiento de una computadora de uso general. Utilizaremos una computadora hipotética. Consideraremos una unidad aritmético-lógica y una unidad de control interactuando con la información (datos e instrucciones) de una memoria principal.

Consideremos primero la Unidad Aritmético Lógica, esta podrá realizar las cuatro operaciones aritméticas básicas (suma, resta, división y multiplicación), y cuatro operaciones lógicas (AND, OR, NOT, NAND). Es labor del arquitecto de computadoras determinar, qué cadena de bits corresponde a cada operación. Por ejemplo, consideremos el set de instrucciones de la tabla 1 y su código binario correspondiente.

Cada instrucción debe contener un código de operación (Suma, resta, or, etc.) y a continuación un campo de dirección (posición de memoria donde se encuentra el dato a operar). En la figura 5 se muestra un ejemplo de formato de una instrucción de la máquina hipotética que consideramos en esta unidad.



(a) Palabra Número



(b) Palabra Instrucción

Figura 5. Formato de memoria IAS.

Fuente: Elaboración propia



Actividades de aprendizaje recomendadas

Actividad 1:

Para introducirse al mundo de la arquitectura de computadores, es necesario comenzar conociendo la historia de la computadora.

En el siguiente video se describe las diferentes tecnologías, personas e instituciones que permitieron la evolución de la computadora hasta llegar a nuestros días, además que describe indirectamente la estructura de una computadora a través de la historia.

VIDEO

History Channel. (2013). *La Historia del Computador*. [video]. USA. Recuperado de [La historia del computador](#).

Luego de observar el video, le invito a que responda las siguientes preguntas:

¿Qué evento mundial motivó la creación de computadores como Colossus y ENIAC?

¿Cuál fue la influencia de la invención del transistor en el desarrollo de las computadoras?

¿Quién fue Alan Turing y John Von Neumann?

Actividad 2:

- **Actividad de aprendizaje:**
Proceso de computación.
- **Procedimiento:**
Realice el ejercicio descrito a continuación.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Para comprender mejor los conceptos básicos del funcionamiento de la computadora, es recomendable que realice el siguiente ejercicio (Alberca y Jiménez, 2013).

Encontrar el mayor de un número en un arreglo de 100 posiciones en la memoria de una computadora IAS (Institute for Advanced Study).

La computadora IAS fue la primera computadora digital y en su diseño colaboró John Von Neumann, especialmente en su arquitectura.

	000	6	
D	001	5	
I	002	4	A
R	003	1	R
E	004	2	R
C	005	3	E
C	006	4	G
I	007	10	L
O	:	11	O
N	:	8	
E	099	9	
E	100		
I	101	0	Contiene cada elemento del arreglo (S)
A	102	0	Contiene el elemento mayor (B)
S	103	99	Contiene el número de posiciones a repetir (C)
	104	1	Contiene una constante (D)

Figura 6. Esquema de memoria IAS.

Fuente: Alberca y Jiménez (2013).

Observe que la memoria IAS se compone de direcciones de una longitud de 12 bits de acuerdo con la definición. En este ejemplo se ha enumerado el arreglo secuencialmente desde 000 hasta 099, para que pueda entender que se trabajará con cien elementos.

Generalmente estas direcciones se encuentran en hexadecimales,

pero para facilitar la comprensión del ejemplo se ha manejado con expresiones decimales (en base 10). Además de las direcciones, otro componente es la información misma que en este caso son números también representados en formato decimal.

Bajo el arreglo se observa que los espacios de memoria 101, 102, 103 y 104 son utilizados temporalmente para el proceso de obtener el mayor de los elementos.

En las posiciones siguientes de memoria se encuentran las instrucciones que permiten obtener el mayor número de los elementos. Observe que parte desde la posición 105 (figura 7).

	Codop	Dirección	Codop	Dirección
105	LOAD M(X)	099	STORE M(X)	101
106	LOAD M(X)	102	SUB M(X)	101
107	JUMP+MX(X,0:19)	108	JUMP M(X,0:19)	111
108	LOAD M(X)	103	SUB M(X)	104
109	STOR M(X)	103	STOR M(X,8:19)	105
110	JUMP + M(X,0:19)	105	JUMP M(X,0:19)	113
111	LOAD M(X)	101	STOR M()	102
112	JUMP M(X,0:19)	108	-----	-----
113				
114				
115				
116				

Figura 7. Instrucciones guardadas en memoria.

Fuente: Alberca y Jiménez (2013).

El algoritmo de la figura 7 describe lo siguiente.

1. Inicializar a B con 0, aquí se guardará temporalmente el elemento mayor en cada comparación.
2. Inicializar S con el primer elemento a comparar.
3. Realizar la operación B-S.
4. Si la operación anterior es negativa, entonces ubicar en B el valor de S (indica que S es mayor).

5. Si la operación es positiva (indica que B es mayor o igual), mantener el valor de B.
6. Asignar a S el siguiente elemento a comparar.
7. Volver al paso 3 hasta que se recorran todos los elementos del arreglo. (C<99).

Más adelante regresaremos a estos conceptos con más detalle.

La unidad aritmético-lógica realizará en su *hardware* las operaciones correspondientes; sin embargo, ¿cuál es el papel de la unidad de control?

La unidad de control es la encargada de recibir la instrucción y decodificarla generando la señal correspondiente a ser enviada a la Unidad Aritmético Lógica (ALU), y esta a su vez ejecutar la operación indicada en la instrucción. Una vez decodificada la instrucción, se extraen los datos de la memoria principal, estos a su vez se almacenan temporalmente en pequeñas unidades de memoria llamadas “Registros”, por ejemplo: el MDR (Registro de Datos de Memoria) o el MAR (Registro de Direcciones de Memoria).

Si observamos el número de bits que hemos reservado para la memoria, notamos que con 4 bits es posible indicar 16 posiciones de memoria, $2^4=16$, es decir, desde la posición 0000 a la posición 1111. En una computadora real necesitamos direcciones mucho más largas para direccionar toda la memoria disponible.

En las siguientes unidades describiremos los componentes nombrados en esta unidad a detalle.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Actividad 3.

Actividad de aprendizaje:

Autoevaluación.

Procedimiento:

Estimado estudiante, las siguientes preguntas constituyen una herramienta para evaluar la comprensión de esta unidad. En caso de que alguna de ellas se le dificulte o no pueda contestar, por favor revise nuevamente la unidad.



Autoevaluación 1

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolla la autoevaluación, responda correctamente a las cuestiones planteadas.

1. ¿A qué se refiere la arquitectura de un computador?
 - a. A los atributos de un sistema que son visibles para un programador.
 - b. A los atributos estéticos de un computador.
 - c. Unidades funcionales e interconexiones.
2. Los atributos arquitectónicos de un computador son:
 - a. Unidades funcionales e interconexiones.
 - b. Señales de control e interfaces entre el computador y los periféricos.
 - c. Los conjuntos de instrucciones, mecanismos de E/S.
3. ¿Qué característica de un computador ha demostrado mantenerse a lo largo de muchos años?
 - a. Su arquitectura.
 - b. Su organización.
 - c. Su estructura.
4. ¿Qué nos indica la estructura de un computador?
 - a. El modo en cómo los componentes están relacionados.
 - b. El tipo de memoria y sus distintos niveles de acceso.
 - c. La operación de cada componente.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

5. La estructura de un computador general está conformada por:
 - a. Memoria, E/S, multiplexores, unidad de control.
 - b. ALU, Unidad de control, memoria, E/S
 - c. ALU, memoria, contadores, reloj del sistema.
6. De los siguientes literales elija uno que forme parte de la arquitectura de un computador:
 - a. El bus del sistema.
 - b. El número de bits usados para representar varios tipos de datos.
 - c. Los registros.
7. Entre las funciones básicas que ejecuta un computador podemos encontrar:
 - a. Toma de decisiones.
 - b. Planificación del tiempo.
 - c. Transferencia de datos y control.
8. ¿De qué se encarga la Unidad de Control?
 - a. Controla el funcionamiento del CPU y, por tanto del computador.
 - b. Proporciona almacenamiento interno a la CPU.
 - c. Transfiere datos entre el computador y el entorno externo.
9. ¿Cuál de los siguientes literales abarca algunos de los componentes estructurales de un computador?
 - a. Periféricos, Líneas de Comunicación.
 - b. La CPU, Memoria Principal, E/S, Sistema de interconexión.
 - c. Registros, Unidad de Control, ALU.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

10. ¿Qué componente estructural del computador se encarga de la comunicación entre la CPU y la memoria principal?
- a. Líneas de comunicación.
 - b. E/S.
 - c. Sistema de interconexión.

[Ir al solucionario](#)

Resultados de aprendizaje 3 y 4

- Conoce la función de la Unidad Aritmético Lógica dentro del computador.
- Representa cantidades numéricas e información en formato binario.

Contenidos, recursos y actividades de aprendizaje



Semana 2



Unidad 2. Unidad Aritmético-Lógica

A partir de esta unidad se describirá a detalle los bloques funcionales de una computadora, comenzaremos por la Unidad Aritmético-Lógica (ALU). La ALU es la encargada de realizar las operaciones de una computadora. Estas operaciones están implementadas a nivel de *hardware*. Como lo revisamos en la introducción, se pueden realizar operaciones aritméticas y lógicas.

Una ALU está conformada por compuertas y circuitos lógicos que realizan estas operaciones entre dos registros de bits.

Consideremos por ejemplo una ALU que realice cuatro operaciones aritméticas (suma, resta, multiplicación, división) y cuatro operaciones lógicas (OR, AND, NOT, X-OR). En hardware necesitaremos tres líneas de control para elegir la operación deseada. Mediante una tabla de verdad podemos apreciarlo.

Tabla 1. Tabla de verdad para codificación de operaciones

D_0	D_1	D_2	Operación
0	0	0	Suma
0	0	1	Resta
0	1	0	Multiplicación
0	1	1	División
1	0	0	AND
1	0	1	OR
1	1	0	NOT
1	1	1	X-OR

La unidad debería enviar una señal de tipo $D_0=0, D_1=1, D_2=1$, es decir (001), para que la ALU realice la operación de resta entre las cantidades guardadas en los registros de entrada. La operación que se forma es $C = A-B$.

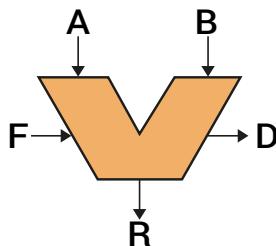


Figura 8. Símbolo de una unidad aritmético-lógica.

Las estradas A y B son los operandos, la entrada F es la línea de control, R es la salida de respuesta y D una señal tipo bandera (Acarreo, desborde aritmético, etc.) Fuente: [enlace web](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Como lo indica la figura 8, una ALU tiene generalmente dos registros de entrada llamados A y B, y un registro de salida R que es la respuesta. Cabe indicar que la salida de una ALU puede contener registros de salida de un solo bit como por ejemplo: acarreo o cero.

Recordemos que las operaciones básicas de la ALU son implementaciones en *hardware*, las operaciones más complejas que una computadora puede realizar se basan en *software* que descomponen operaciones complejas en una serie de operaciones básicas. Por ejemplo, para resolver una Ecuación Diferencial se puede aplicar el Método de Euler y la operación se reduce a una serie de iteraciones utilizando las cuatro operaciones básicas.

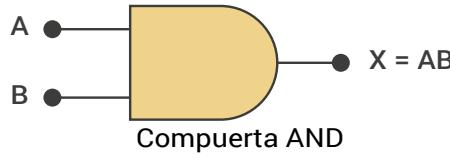
2.1. Descripción del hardware de una ALU

Una ALU está formada por una serie de dispositivos basados en semiconductores, los dispositivos más básicos son las compuertas lógicas, que a su vez están conformadas por dos transistores.

Compuerta AND: la compuerta realiza la operación lógica AND entre dos bits en su entrada, esta operación es similar a la multiplicación, su salida es también un *bit* que responde a la siguiente tabla de verdad.

AND		
A	B	$x = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

(a)



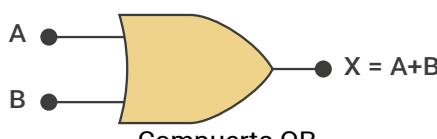
(b)

Figura 9. a) Tabla de verdad de una compuerta AND, b) Símbolo lógico
Fuente: Tocci, et. al. (2017).

Compuerta OR: la compuerta realiza la operación lógica OR, es similar a la operación suma. al. (2017).

OR		
A	B	$x = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

(a)



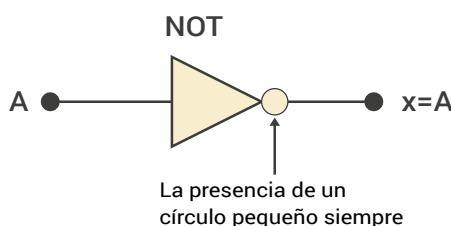
(b)

Figura 10. a) Tabla de verdad de una compuerta OR, b) Símbolo lógico.
Fuente: Tocci, et. al. (2017).

Compuerta NOT: esta compuerta realiza la operación del complemento o negación.

NOT		
A		$x = A$
0		1
1		0

(a)



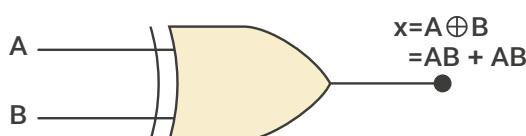
(b)

Figura 11. a) Tabla de verdad de la compuerta inversora, b) Símbolo lógico.
Fuente: Tocci, et. al. (2017).

Compuerta X-OR: La compuerta realiza la operación *OR exclusivo*. La salida es cero, solo cuando las dos entradas son iguales.

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Figura 12. a) Tabla de verdad OR exclusivo, b) Símbolo lógico.

Fuente: Tocci, et. al. (2017).

2.2. Circuitos aritméticos

Para que la ALU pueda realizar operaciones aritméticas es necesario combinar algunas compuertas lógicas. Observe la figura 13, donde se muestra un esquema de las interconexiones de la ALU con la unidad de control y la memoria, dentro de la ALU podemos observar tres bloques: un acumulador, registro B y circuitos lógicos, el bloque “circuitos lógicos”, contiene el conjunto de compuertas lógicas que forman circuitos que operaran aritmética o lógicamente el contenido de *bits* que se encuentra en el acumulador y el registro B. Cabe destacar que en algunas arquitecturas de computador pueden existir más registros dentro de la ALU.

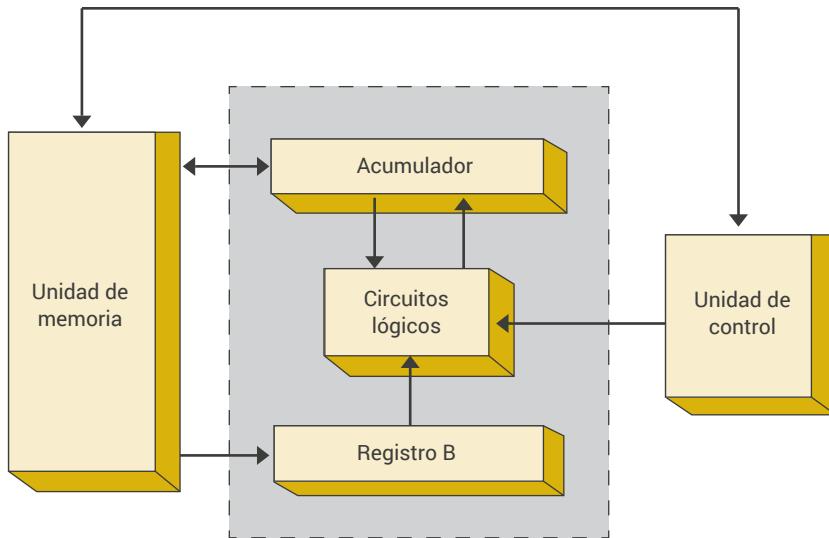


Figura 13. Esquema de una ALU y el uso de los circuitos lógicos.
Fuente: Tocci, et. al. (2017).

2.2.1. Sumador completo

En esta sección describiremos la forma como se puede realizar la operación suma dentro de la ALU. La operación de la suma requiere que se aplique la siguiente lógica cuando se suman dos bits:

Tabla 2. Tabla de verdad sumador dos bits.

A	B	Acarreo (C)	Suma (S)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

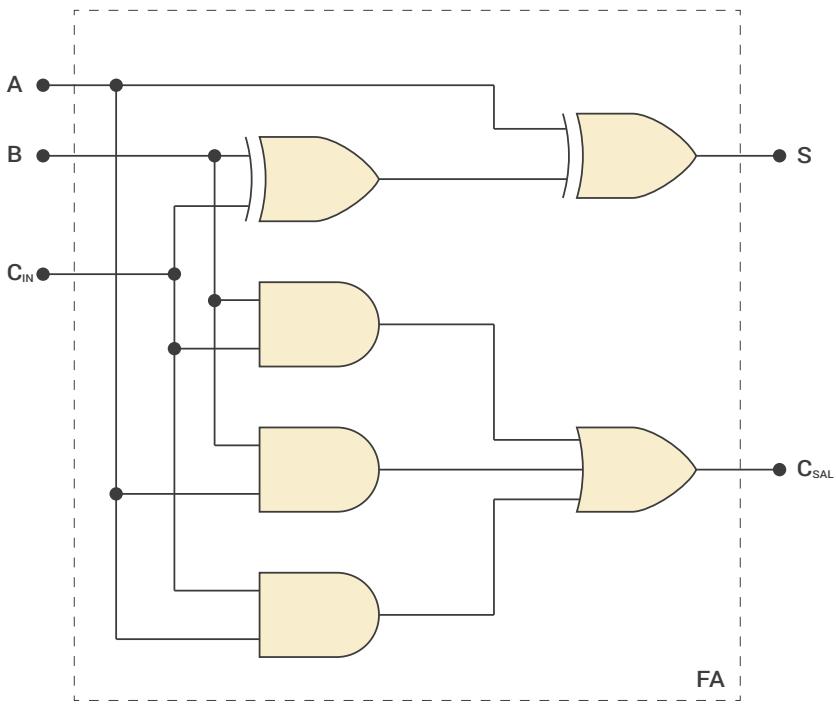


Figura 14. Circuito de sumador completo.

Fuente: Tocci R. et al. (2017)

El circuito anterior solo puede sumar dos *bits* a la vez. Para conseguir sumar dos registros de varios bits, es necesario agrupar varios *sumadores completos*, siendo la salida de acarreo del bit menos significativo la entrada de acarreo del siguiente sumador completo, así como se muestra en la figura 15, aquí se suman dos registros de 5 *bits* cada uno. Observe cómo cada *bit* del registro B se suma con el *bit* correspondiente del A. Desde la derecha se puede apreciar el *bit* B₀ y A₀ como entradas del sumador completo (Full Adder) FA#0, este sumador generará una suma binaria S₀ y un acarreo, el acarreo se corresponderá a la entrada C₁ del segundo sumador completo FA #1.

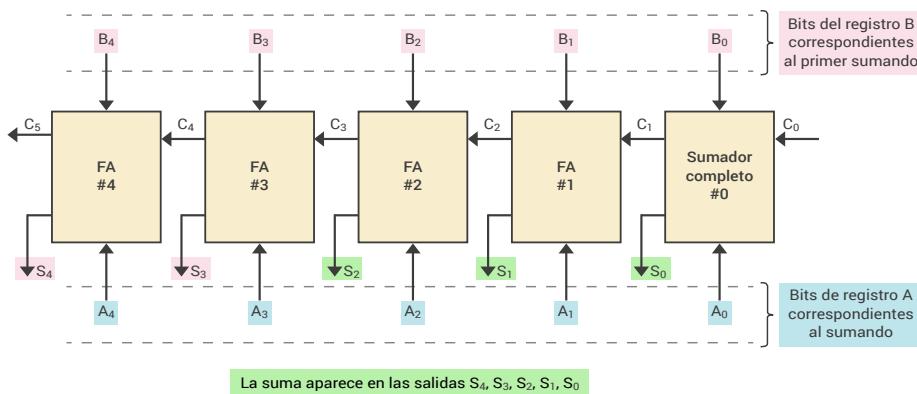


Figura 15. Esquema de un sumador de dos registros de 5 bits.

Fuente: Tocci, et. al. (2017).

2.2.2. Resta Binaria

Otra operación importante que realiza la ALU es la resta. La resta binaria se realiza con ayuda del circuito sumador, simplemente se debe negar el registro B, así:

$$A - B = A + (-B)$$

Para realizar esta negación es necesario obtener el complemento a 2 del segundo registro. El complemento a 2 consiste en los siguientes pasos.

1. Obtener el complemento a 1 de los bits que conforman el registro.
2. Sumar 1 al complemento a 1.

2.2.3. Multiplicación binaria

La unidad aritmético-lógica básicamente puede realizar físicamente solo sumas y restas, el resto de las operaciones como la multiplicación que se describe aquí se realizan con ayuda de algoritmos que permiten utilizar las operaciones aritméticas suma y resta y las operaciones lógicas para ejecutar operaciones más

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

complejas. La multiplicación binaria puede realizarse mediante un proceso de desplazamiento (SHIFT) y sumas. Por ejemplo, la multiplicación de 4 X 2 en binario:

$$\begin{array}{r} 0100 \\ \times \frac{0010}{0000} \\ 0100 \\ 0000 \end{array}$$

El proceso es el mismo de la multiplicación aritmética, donde cada número del multiplicador se multiplica por el multiplicando, cada respuesta se desplaza una posición hacia la izquierda, finalmente se suman las respuestas parciales.

Sin embargo, el proceso descrito anteriormente puede ser demasiado demandante para una computadora. En el ejemplo anterior se necesitaría un circuito que multiplique dígito a dígito (16 multiplicaciones) y 3 desplazamientos y 12 sumas. Esto implicaría muchos ciclos de reloj (una operación por cada ciclo de reloj). Para optimizar este proceso se recurre a un algoritmo basado en microcódigo, este algoritmo permite realizar la multiplicación usando solo sumas y desplazamientos. Este proceso se conoce como *algoritmo de Booth*. Observemos este algoritmo en el flujoograma de la figura 16. Se inicia con una primera fila donde se ubican los registros C (acarreo) y A (registro auxiliar para respuesta), la longitud del registro A debe corresponder con la longitud del multiplicando y el multiplicador M y Q, respectivamente. El algoritmo comienza verificando si el bit menos significativo del registro Q es uno, si es así, se realiza la operación suma entre los registros A y M. La respuesta de la suma anterior se registra en A, en caso de existir un acarreo se registra este bit en C. Luego se realiza una operación de desplazamiento (SHIFT) de toda la fila, sin tomar en cuenta al registro

M. Puede observar un ejemplo en la actividad recomendada que se encuentra a continuación.

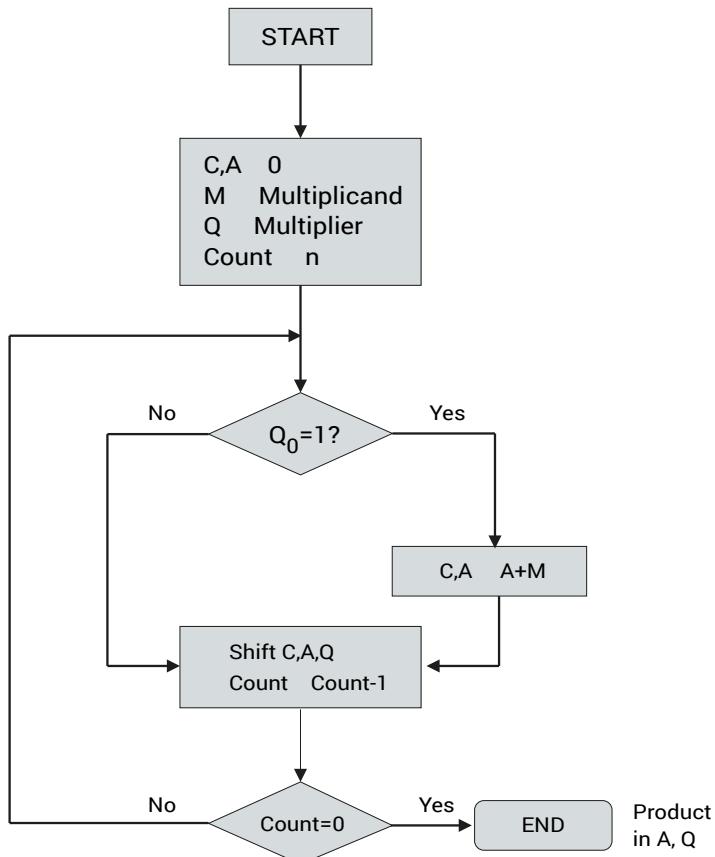


Figura 16. Algoritmo de Booth.

Fuente: Stallings (2007).



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Suma y resta de números binarios.

- **Procedimiento**

Realice los ejercicios propuestos a continuación.

1. Obtener el complemento a 2 de los siguientes números binarios: 0011, 0010₂, 0101₂.
2. Obtener el complemento a 1 de 0011₂ equivalente a 3₁₀ en decimal.
 - a. 0011-> 1100
 - b. Sumamos un 1 bit al número binario en complemento a 1.
$$\begin{array}{r} 1100 \\ + 0001 \\ \hline 1101 \end{array}$$
 - c. 1100

El (-3) binario será 1101₂

Repita el ejercicio para los números 0010₂ y 0101₂

3. Realizar la operación resta utilizando el complemento a 2 de los siguientes números: 9_{10} y 7_{10} , cuyo equivalente binario es 1001_2 y 0111_2 .

El primer paso es convertir el operando que se va a restar a complemento a 2, en este caso el 7_{10} :

$$\text{a. } 0111_2 \rightarrow \text{Ca2} \rightarrow 1001_2$$

Realizamos la operación suma del primer operando 9_{10} con el operando en complemento a 2 7_{10} .

b.

$$\begin{array}{r} 1001 \\ + 1001 \\ \hline 10010 \end{array}$$

Ignoramos el 5 bit a la izquierda que es el acarreo de la operación, en este caso es un desborde aritmético y en la solución solo se consideran los 4 bits menos significativos (derecha): 0010_2 que en decimal es 2_{10} obteniendo la respuesta correcta.

Así queda demostrado que no se necesita un circuito extra para la resta, se utiliza el mismo circuito de la suma, solo es necesario agregar un circuito que realice el complemento. La unidad de control enviará las señales necesarias para activar o desactivar el circuito complemento a 2 dependiendo de la operación a realizar: Suma o resta.

Actividad 2:

- **Actividad de aprendizaje**
Multiplicación mediante algoritmo de Booth.
- **Procedimiento:**
Realice los ejercicios propuestos a continuación.

Multiplique: $11 \times 13 \rightarrow 1011_2 \times 1101_2$

C	A	Q	M	Valores iniciales: registros C y A en 0.
0	0000	1101	1011	
0	1011	1101	1011	Suma
0	0101	1110	1011	Desplazamiento } Primer ciclo
0	0010	1111	1011	Desplazamiento } Segundo ciclo
0	1101	1111	1011	Suma
0	0110	1111	1011	Desplazamiento } Tercer ciclo
1	0001	1111	1011	Suma
0	1000	1111	1011	Desplazamiento } Cuarto ciclo

Figura 17. Ejecución algoritmo de Booth.

Fuente: Stallings (2007).

Multiplique: 10×8

Multiplique: 10×-8

Para realizar el segundo literal considere realizar la operación de signo al final del proceso de multiplicación. Existe una versión del algoritmo de Booth que permite multiplicar directamente números con signo. Puede consultarla en el texto *Arquitectura y organización de computadores* Stallings (2007, p. 318).

Estimado profesional en formación: hemos concluido con el estudio de la unidad 2 donde describimos el funcionamiento de la ALU. A continuación, le invitamos a realizar la autoevaluación 2. Si tiene

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

alguna duda con alguna pregunta, revise la retroalimentación al final de este texto-guía o consulte los recursos en el entorno virtual de aprendizaje.

Actividad 3:

Actividad de aprendizaje

Autoevaluación

Procedimiento:

Realice la autoevaluación a continuación.

Estimado estudiante, las siguientes preguntas constituyen una herramienta para evaluar la comprensión de esta unidad. En caso de que alguna de ellas se le dificulte o no pudo contestar por favor revise nuevamente la unidad.



Autoevaluación 2

Arquitectura de Computadoras y Sistemas Operativos.

Dentro de las casillas correspondientes escriba V o F y opción múltiple para cada uno de los siguientes items.

1. () La unidad aritmético lógica es en esencia un bloque de instrucciones de la memoria principal
2. () La unidad aritmético lógica realiza operaciones lógicas y aritméticas básicas
3. () Los circuitos de la ALU están conformados por compuertas lógicas, las que a su vez forman estructuras que permiten realizar operaciones aritméticas como la suma
4. () Un sumador completo permite operar solo dos bits a la vez
5. () La operación de la multiplicación se realiza en la ALU mediante un algoritmo micro programado llamado Algoritmo de Booth
6. () Según la arquitectura de Von Neumann una ALU tiene conexión directa con la memoria principal
7. () La unidad de control es la encargada de decodificar la instrucción y emitir las señales adecuadas a la ALU para que ejecute la operación solicitada.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Responda según corresponda.

8. En el caso de que la solución de una operación realizada por la ALU sobrepase el número bits de longitud soportado por la ALU se activa la salida de bandera indicando un...
 - a. Error.
 - b. Desborde aritmético.
 - c. Cero.
9. Las operaciones lógicas AND, OR, NOT son realizadas por unidades llamadas...
 - a. Compuertas lógicas.
 - b. Flip-Flops.
 - c. Circuitos analógicos.
10. Las operaciones aritméticas básicas se realizan en la ALU mediante...
 - a. Instrucciones software y hardware.
 - b. Circuitos lógicos.
 - c. Sólo software.

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultados de aprendizaje 5 y 6

- Ordena en una pirámide de jerarquía a las distintas tecnologías de memoria en función de su rapidez.
- Balancea el diseño de una caché en función de su rapidez, tecnología y tamaño.

Contenidos, recursos y actividades recomendadas



Semana 3



Unidad 3. Sistema de memoria

En la siguiente unidad abordaremos uno de los bloques principales de una computadora, el sistema de memoria.

3.1. Concepto de memoria

Según el modelo general de Von Neumann, la computadora utiliza un sistema de memoria para guardar tanto datos como instrucciones. Una primera clasificación puede comprender dos tipos de memoria: interna y externa. La memoria interna se utiliza para las tareas de la CPU, y la externa, por lo general, se utiliza para guardar cantidades grandes de información.

El sistema de memoria actual ha evolucionado a una jerarquía de acuerdo con distintas necesidades de velocidad, tamaño y precio. Esta jerarquía es:

- Registros.
- Memoria caché.
- Memoria principal.
- Memoria secundaria.
- Medios extraíbles.

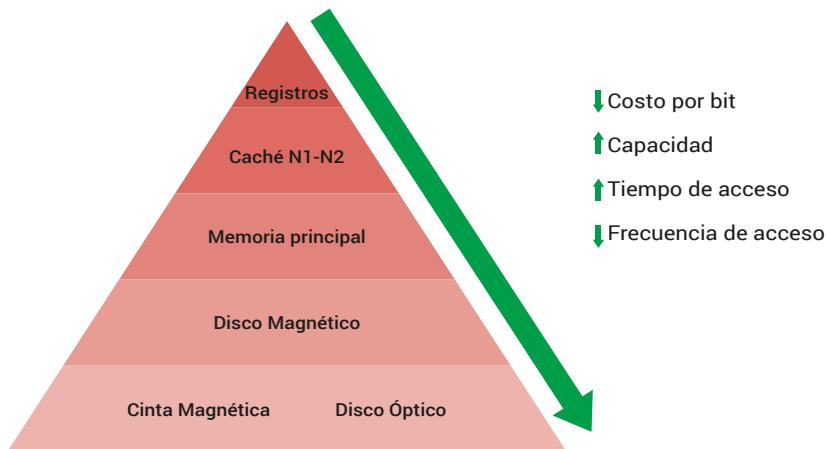


Figura 18. Jerarquía de memoria en una computadora.

Fuente: Alberca y Jiménez (2013).

La figura 18 describe importantes características sobre la jerarquía de memoria, la ubicación de cada tecnología de memoria en la pirámide define su costo, capacidad, tiempo de acceso y frecuencia de acceso.

Así, podemos identificar que los registros, por estar en el nivel más alto de la pirámide tienen el menor tiempo de acceso, la mayor frecuencia de acceso, pero la menor capacidad y mayor costo por bit.

Al contrario, con la menor jerarquía, en la base de la pirámide tenemos al disco magnético (disco duro), que tiene el menor costo por bit, la mayor capacidad, pero el mayor tiempo de acceso y la menor frecuencia de uso.

Registros:

Son pequeñas memorias a nivel de unos cuantos *bits* de extensión, residen en la CPU, por ejemplo, los registros de la ALU, los registros de direcciones (MAR), el *buffer* de memoria MBR, el contador de programa (PC) que lleva la cuenta de la siguiente instrucción a ejecutar. Los registros tienen la mayor velocidad de acceso y el menor tamaño de toda la jerarquía.

Memoria caché:

Es una memoria ubicada entre la CPU y la memoria principal. Son memorias basadas en semiconductores, es una versión reducida de la memoria principal, pero mucho más rápida; sin embargo, no es posible igualar su tamaño al de la memoria principal debido al elevado costo de su tecnología. La memoria caché es usada para almacenar la información más reciente usada o con más frecuencia, la cual es traída desde la memoria principal. Es decir, las instrucciones más frecuentemente usadas o las últimas usadas se almacenan en la caché para facilitar el acceso desde la CPU. Actualmente se pueden encontrar CPU con caché interna.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Memoria principal:

La memoria principal se fabrica también con base en semiconductores; sin embargo, es más lenta que la memoria caché. La memoria principal almacena la información, datos o instrucciones con los que trabajará la CPU.

Memoria secundaria:

La memoria secundaria comprende los dispositivos que almacenan información en bruto, por ejemplo, discos duros, pen drive, CD-ROM, etc. El CPU no puede trabajar con estos datos directamente, para esto, se debe enviar solo las instrucciones o datos necesarios para una tarea específica a la memoria principal.

Actividad Recomendada:

Es necesario que usted identifique toda la organización de memoria y jerarquía en el estado de la tecnología actual. Considere en qué nivel de la pirámide se encuentra el almacenamiento en "la nube". Discuta su respuesta en el entorno virtual de aprendizaje.

3.1.1. Operación de la memoria caché

El procesador no conoce específicamente de la existencia de la caché, el procesador emite órdenes tipo: READ o WRITE; sin embargo, estas instrucciones contienen solo direcciones de la memoria principal. El circuito que controla el acceso a memoria determina si la dirección solicitada por la CPU se encuentra en la caché, de no ser así, se carga desde la memoria principal.

El proceso de intercambiar datos e instrucciones desde la memoria principal a la caché se conoce como *funciones de correspondencia*.

Cuando la caché está llena y se necesita ejecutar un número de instrucción que no se encuentra en la caché, se debe remplazar

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

alguna instrucción previamente guardada en la caché para ingresar la nueva instrucción o grupo de instrucciones, para este proceso se utiliza los llamados *Algoritmos de Sustitución*.

Cuando la CPU necesita escribir una línea de memoria se puede proceder de algunas maneras. Por ejemplo, la técnica *Escritura Inmediata*, la operación de escritura o remplazo de una línea de memoria se la hace al mismo tiempo, tanto en la caché como en la memoria principal. (Recordemos que la caché contiene una copia de los bloques de memoria principal). Esto asegura que tanto las direcciones de memoria en la caché como en la memoria principal siempre estén actualizadas; sin embargo, esto puede crear tráfico innecesario en el *bus de memoria* y un cuello de botella. Este problema se puede minimizar con la técnica de posescritura, aquí se actualiza solo la caché, y se activa un *bit* conocido como *Dirty bit* o simplemente *bit de actualización*, cuando el bloque de memoria de caché debe ser sustituido, este se copia de vuelta en memoria principal, solo si, el *bit* de actualización está en estado activo.

Consideré el caso en que la CPU dirccione o necesite una palabra que no está en la caché. El bloque que contiene la palabra debe pasarse desde la memoria principal a la caché; sin embargo, la palabra que la CPU necesita podría estar disponible al mismo tiempo que lo está para la caché, a esto se le llama *carga inmediata*. El mismo proceso puede considerarse la operación escritura, si la CPU necesita en este caso escribir información en una dirección de memoria que no se encuentra en la caché, podría hacerlo directamente en la memoria principal.

3.2. Funciones de correspondencia

La caché es una memoria muy rápida, pero también mucho más pequeña que la memoria principal, las funciones de correspondencia se utilizan para hacer corresponder de manera óptima los bloques de memoria principal con las líneas de la memoria caché.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Para comprender mejor los siguientes ejemplos le sugerimos hacerlos en una hoja de notas.

Consideré el siguiente caso:

Una CPU cuenta con una caché de 4 KB cada palabra tiene la longitud de un byte, la misma está organizada en bloques de 32 palabras, consecuentemente, contiene 128 bloques ($32 \times 128 = 4096$). Los bits necesarios para direccionar cada bloque serán 7 ($2^7=128$), y los bits necesarios para direccionar cada palabra serán 5 ($2^5=32$), así, se necesitan 12 bits en total para direccionar cada línea de la caché.

Consideremos ahora una memoria principal con 16 MB, de igual manera está organizada en bloques de 32 palabras, conteniendo 524 K bloques (524288) necesitando 19 bits para direccionar todos los bloques, y 5 bits para direccionar las palabras. En total 24 bits para direccionar la memoria principal.

Luego de hacer estas consideraciones podemos discernir que en cualquier momento solo 128 de 524288 bloques de la memoria principal pueden alojarse en la memoria caché. Debido a esto se debe considerar algunas técnicas para corresponder un bloque de la memoria principal en un bloque de la caché.

Las técnicas principales se detallan a continuación.

3.2.1. Correspondencia directa

Los bloques de memoria principal pueden llevarse a una sola línea específica de la memoria caché. Es la técnica más simple, pero carece de flexibilidad.

En esta técnica el bloque k de la memoria principal se graba en el bloque k módulo m de la caché, donde m es el número de bloques de la caché, en el caso anterior, $m = 128$ bloques. Se debe lograr que un bloque específico de la memoria principal se pueda transferir a un bloque específico de caché.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Esto se logra dividiendo la dirección de memoria principal en tres campos: etiqueta + bloque + palabra. Para el ejemplo específico que se detalla arriba, el primer campo palabra contiene 5 bits que identifica una palabra dentro de un bloque. Los restantes 19 bits se dividen en dos bloques: el primero de 7 bits, se usa para seleccionar un bloque dentro de los 128 de la caché. El segundo bloque de 12 bits se usa como una etiqueta para identificar el bloque específico que se copió en la caché.

Cuando un bloque de la memoria principal ya está copiado en caché, los bits más significativos (*Most Significant bit* MSB) se guardan en el campo etiqueta. Cuando la CPU genera un pedido de memoria, el bloque de 7 bits determina el bloque específico de la caché. El campo Etiqueta de ese bloque se compara con los 12 bits superiores de la dirección solicitada por la CPU, si coinciden, la palabra deseada especificada por los 5 bits inferiores se encuentra efectivamente en ese bloque y línea de caché, esto se llama *Acierto de Caché*. En el caso de que el campo Etiqueta del bloque de caché y los 12 bits superiores de la dirección solicitada por la CPU no coincidan, significa que la información no ha sido copiada aún a la caché y se debe buscar directamente en la memoria principal. Luego se debe copiar el bloque solicitado a la memoria caché cambiando el campo etiqueta por el apropiado. En la figura 19, podemos observar un ejemplo de correspondencia directa, que muestra gráficamente lo explicado anteriormente, donde el bloque de la derecha representa la memoria caché con sus campos de etiquetas y datos y líneas, los bloques de la izquierda representan la memoria RAM o principal dividida en bloques, se observa que cada bloque se corresponde a una línea de caché.

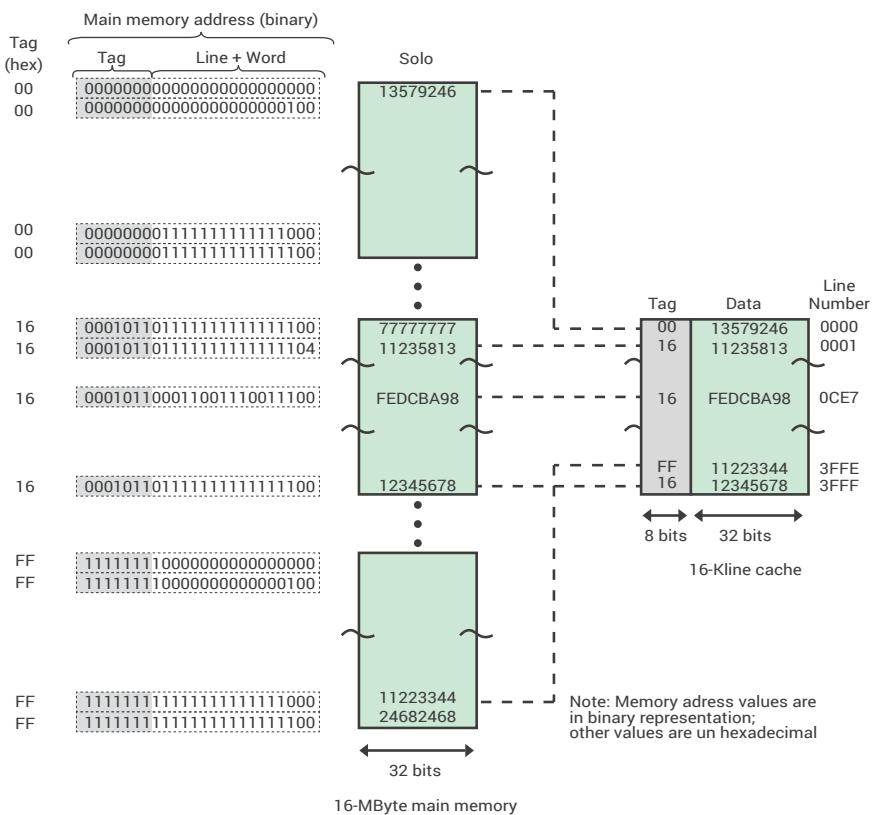


Figura 19. Correspondencia directa.

Fuente: Stallings (2007).

3.2.2. Correspondencia asociativa

Aquí cualquier bloque de memoria principal puede copiarse en cualquier línea de caché, aportando flexibilidad.

En este caso, las direcciones de la memoria principal se dividen en solo dos campos: etiqueta + Palabra, los *bits* inferiores identifican el lugar de la palabra dentro de cada bloque, y los *bits* superiores identifican el bloque. En este caso los 24 *bits* de la memoria principal estarían divididos en 5 *bits* que identifican la posición de la palabra y 19 *bits* que sirven como Etiqueta. Cuando hay un pedido de memoria

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

desde la CPU, se hace un proceso similar al anterior, pero solo se compara los *bits* de la etiqueta de la caché con los 19 *bits* superiores de la dirección, si coinciden se produce un acierto de caché, caso contrario, se debe buscar la dirección en memoria principal y remplazar los bloques en la caché. En la figura 20, podemos apreciar un ejemplo gráfico de esta correspondencia, aquí el bloque de la memoria principal ya no está dividido en bloques, el bloque de la caché ahora debe tener un campo más grande para la etiqueta.

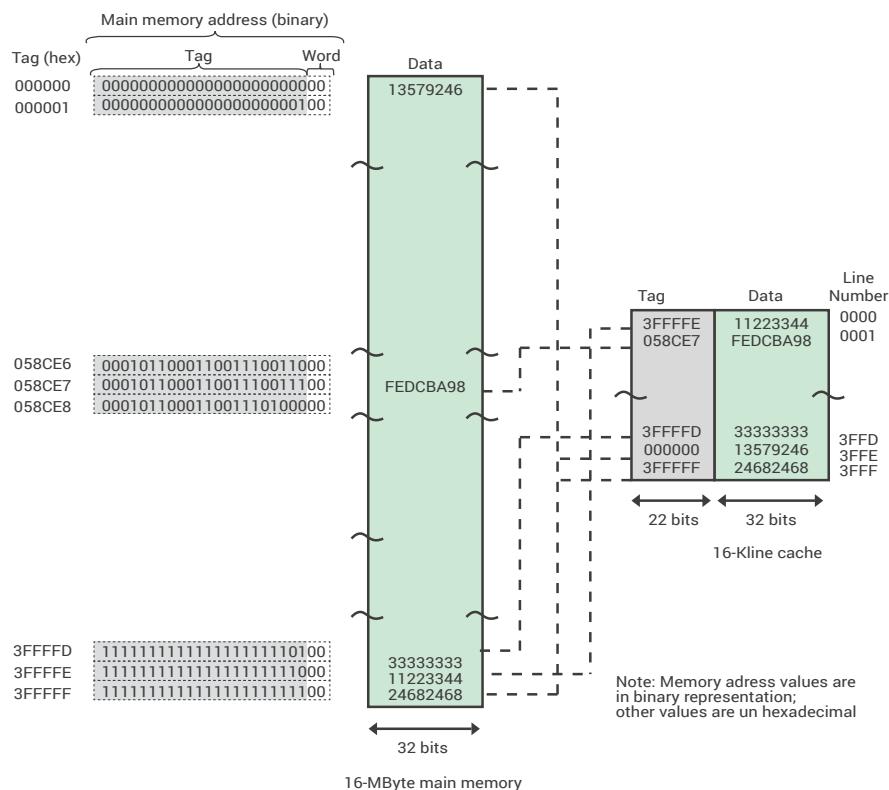


Figura 20. Correspondencia asociativa
Fuente: Stallings (2007)

3.2.3. Correspondencia asociativa por conjuntos

En esta técnica se combina las ventajas de las dos técnicas anteriores. La caché se divide en conjuntos de varias líneas, los bloques de la memoria principal pueden copiarse en cualquier bloque de un conjunto específico.

Los bloques de la memoria caché se agrupan en *conjuntos*, la flexibilidad de la correspondencia asociativa se reduce a la flexibilidad dentro de un conjunto de bloques de la caché. Esto también reduce la carga computacional al no buscar bloques sino a un conjunto.

Ahora el campo etiqueta del ejemplo anterior se divide en dos campos: etiqueta + conjunto. Si consideramos que cada conjunto en la caché contiene por ejemplo 4 bloques, tendríamos 32 conjuntos. Las direcciones de la memoria principal estarían divididas en tres campos: los 5 *bits* inferiores se usan igual que las técnicas anteriores para identificar una palabra dentro de un bloque, los siguientes *bits* se deben usar para identificar el conjunto, (en este caso 32 conjuntos necesitan 5 *bits*), los siguientes *bits* (14) se usarán como etiqueta.

Los 5 *bits* del campo conjunto, se usan para localizar el conjunto que podría contener el bloque deseado. Similar a la técnica de correspondencia directa donde se buscaba por bloque, pero en este caso se busca por conjunto. Luego el campo etiqueta de la dirección debe ser comparado con el campo etiqueta de los cuatro bloques del conjunto. Si coinciden, entonces se produce un acierto de caché, caso contrario, se debe traer el bloque que contiene la dirección desde la memoria principal a la caché.

Se puede observar aquí que, si se aumenta el número de bloques por conjunto, el número de *bits* necesarios para el campo Conjunto se reducen, en el caso extremo de disponer de 128 bloques por

conjunto, estaríamos regresando a la correspondencia asociativa. En el caso de reducir el número de bloques por conjunto el número de bits del campo conjunto aumenta y en el caso extremo de un bloque por conjunto, estaríamos regresando al caso de correspondencia directa.

En la figura 21, podemos observar un ejemplo de lo descrito anteriormente. Los campos etiqueta, conjunto y palabra, están representados en inglés como: *tag*, *set* y *word* respectivamente.

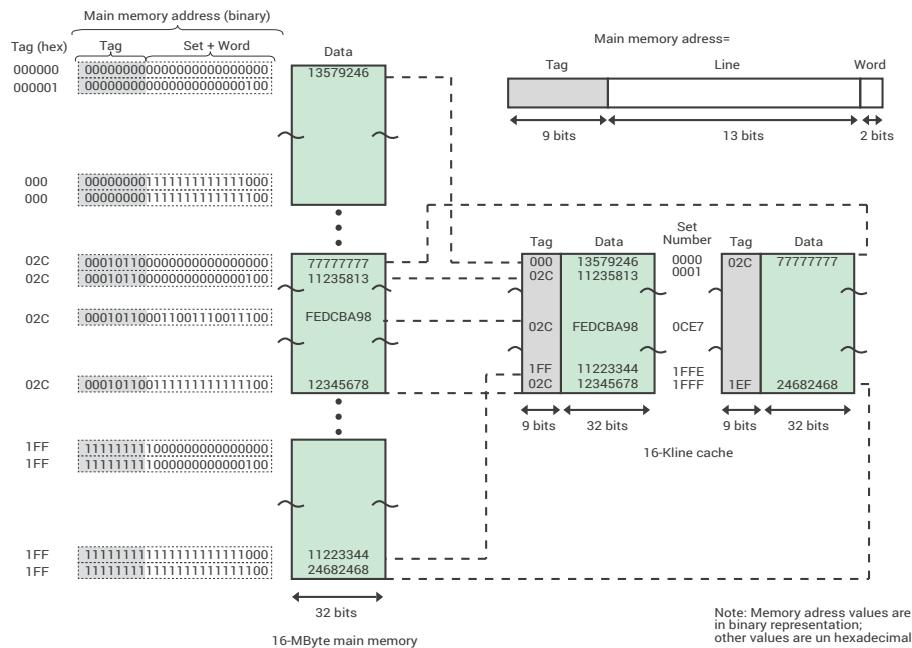


Figura 21. Correspondencia asociativa por conjuntos

Fuente: Fuente: Stallings (2007).

Hasta ahora estudiamos los principales métodos para hacer corresponder el contenido de la memoria principal con las líneas de una caché. Sabemos que la caché es mucho más pequeña en capacidad, en consecuencia, se debe optimizar el espacio utilizado por las instrucciones copiadas desde la memoria principal. Si existen instrucciones ya utilizadas se debe analizar si pueden

ser remplazadas por nuevas. A continuación, estudiaremos los algoritmos de sustitución.

3.3. Algoritmos de sustitución

En el caso de que un nuevo bloque de memoria principal deba copiarse en caché y todas las líneas estén ocupadas, se debe sobrescribir algún bloque o línea antigua. Sin embargo, se debe recurrir a un algoritmo para evitar remplazar bloques que podrían ser usados inmediatamente por la CPU. A continuación, se detalla los más usados.

Utilizado menos recientemente – *Least Recently Used (LRU)*: se remplaza el bloque que no ha sido referenciado durante algún tiempo. Es más probable que un bloque que acaba de ser referenciado sea referenciado en un futuro muy próximo. Se lleva el control de los bloques referenciados mediante un contador el contador se incrementa en uno si el bloque no es referenciado por la CPU. El bloque a remplazar será el que tenga el contador más elevado.

Primero en entrar, primero en salir – *First In First Out (FIFO)*: se remplaza al bloque más antiguo en la caché, en cada solicitud de la CPU, el contador de todos los bloques se incrementa en 1, si un nuevo bloque se copia en la caché este bloque se reinicia a 0.

Escritura aleatoria: se remplaza cualquier bloque de la caché en forma arbitraria, se ha demostrado que es un método muy efectivo y logra niveles de eficiencia muy similares a los algoritmos basados en utilización.

Hemos concluido con la descripción de la memoria caché. Le invitamos a continuar con la siguiente jerarquía de memoria: la memoria principal.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

3.4. Memoria principal

La memoria principal de una computadora es el bloque encargado de almacenar tanto instrucciones como datos, así como lo propuso Von Neumann. La CPU interactúa solo con la memoria principal. La memoria principal de una computadora es de tipo semiconductor. Esto quiere decir que sus unidades básicas son semiconductores (transistores). Principalmente se puede dividir en dos grupos:

RAM: memoria de acceso aleatorio (*Random Access Memory*).

ROM: memoria de solo lectura (*Read Only Memory*).

La diferencia radica en que la RAM es de tipo volátil, es decir, sin energía se pierden los datos guardados en ella. La ROM es de tipo permanente, es decir, que los datos guardados en ella se mantienen aún luego de dejar a la memoria sin energía.

El máximo tamaño que una memoria puede almacenar está determinado por el esquema de direccionamiento. Por ejemplo, una computadora que genera direcciones de 32 bits es capaz de direccionar hasta 2^{32} que es igual a 4 GB de RAM.

En la mayoría de los computadores una dirección individual es asignada a cada byte de información. En estas computadoras una palabra de memoria suele contener uno o más bytes de memoria direccionables individualmente.

Por ejemplo, en una computadora con direcciones de 32 bits. Cada palabra de memoria principal contiene 4 bytes. La forma de direccionar cada byte es asignarle los dos bits menos significativos de una dirección de 32 bits, los 30 bits restantes formarán las direcciones de cada palabra.

Para utilizar la memoria principal la CPU recurre a dos registros conocidos como: registro de Dirección de Memoria (*Memory*

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

*Address Register - MAR) y Registro de Buffer de Memoria (*Memory Buffer Register - MBR*). Las direcciones se transmiten por el *bus* de direcciones el cual es del ancho de las direcciones, en este ejemplo que utiliza direcciones de 32 bits, el ancho del *bus* de direcciones debe ser de también 32 bits. Del mismo modo los datos del MBR se transmiten por el *bus* de datos, el cual debe ser al menos de un byte, 8 bits.*

3.4.1. DRAM y SRAM

Dos formas tradicionales de construcción de memorias RAM son las RAM dinámicas (DRAM) y la RAM estática (SRAM), ambas son de acceso aleatorio y volátiles, pero se diferencian por sus características constructivas.

- **RAM dinámica:** se construye con celdas que guardan los datos (bits) como carga en un capacitor. Es decir, si el capacitor está cargado se interpreta como 1 y si está descargado se interpreta como 0. El problema con esta tecnología es que los capacitores tienden a descargarse con el tiempo, es así como estas celdas requieren de un circuito que refresque la carga continuamente. Una celda típica DRAM se muestra en la figura 22.

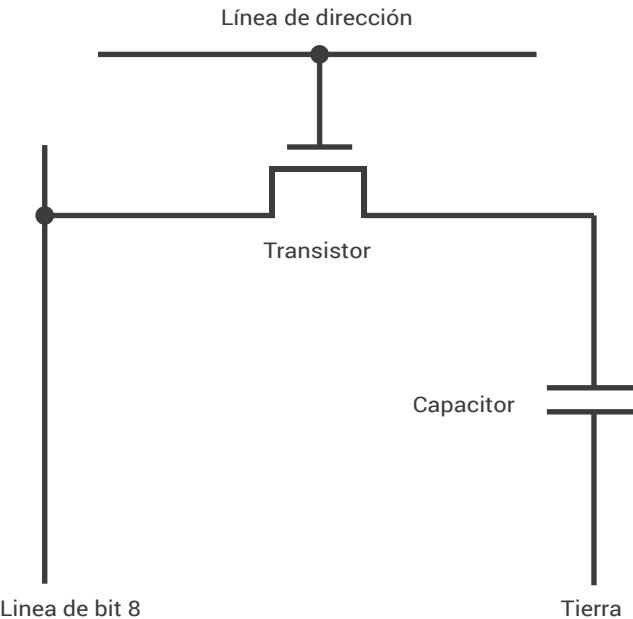


Figura 22. Estructura de RAM dinámica

Fuente: Stallings, W. (2007)

Para la operación de escritura se aplica un voltaje a la línea de bit B. Una señal se aplica también a la línea de dirección, lo que enciende el transistor, permitiendo que el capacitor se cargue.

Para la operación de lectura, se aplica una señal en la línea de lectura activando el transistor, esto hace que la carga del capacitor fluya hacia la línea de bit B.

- **RAM estática:** en una RAM estática se utiliza el dispositivo digital llamado *Flip-Flop* [referenciar flip-flop], este dispositivo está compuesto de dos compuertas NAND o NOR que a su vez se componen de dos transistores cada una. Una típica construcción se puede observar en la figura 23.

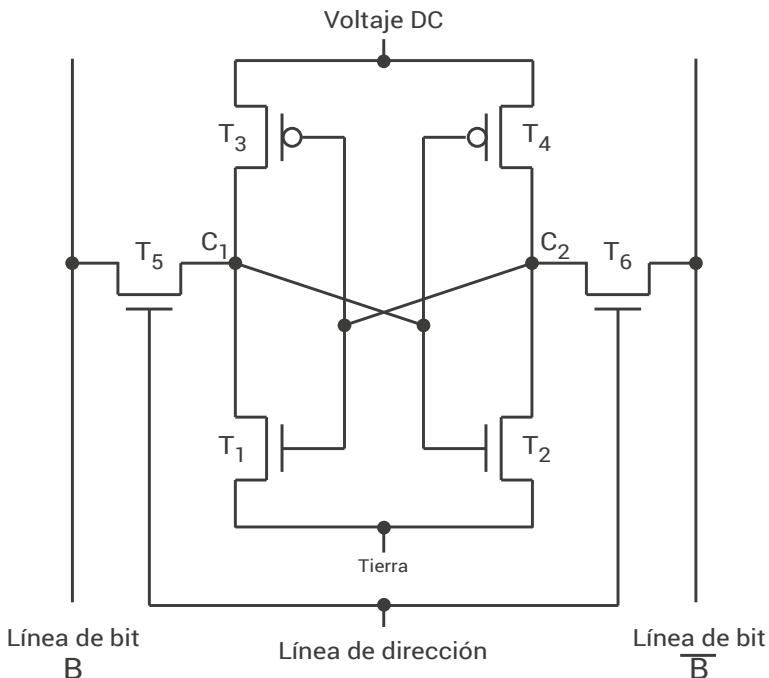


Figura 23. Estructura de la RAM estática

Fuente: Stallings, W. (2007)

Los cuatro transistores (T₁, T₂, T₃ y T₄) están formando las dos compuertas, los que a su vez conforman el *FLIP-FLOP* que es capaz de mantener un *bit* en ALTO en la salida A1 (1) y su complemento en la salida A2 (0). Cuando se almacena un 0, la salida A1 estará en 0 y A2 en 1. Ambos estados se mantendrán así a menos que se corte la alimentación de energía DC.

Para escribir en esta celda es necesario aplicar una señal de voltaje en la línea de *bit* B, lo cual controla los transistores T₅ y T₆, forzando que se cargue un *bit* en el sistema.

Para la operación de lectura se activa la línea de direcciones, lo cual envía el *bit* contenido en A1 a la línea de *bit* B.

3.4.2. SRAM vs DRAM

Ambas tecnologías de memoria son volátiles. Al cortar el suministro de voltaje DC a la memoria, todos los datos de cada celda se borran.

Una memoria DRAM es más simple y pequeña que su contraparte, esto puede traducirse en memorias con mayor densidad, es decir, caben más celdas de memoria en cada circuito. Además, la DRAM es menos costosa que la SRAM.

La DRAM requiere un circuito que refresque continuamente los capacitores, lo que implica un aumento en la energía consumida por la memoria.

Las SRAM son más rápidas de las DRAM. Generalmente las memorias caché son construidas con tecnología SRAM.

3.4.3. Organización interna de los chips de memoria

Físicamente la memoria principal se organiza en chips con matrices de celdas de memoria (estáticas o dinámicas). Para incrementar la velocidad de acceso a los datos de la memoria esta se divide en varios chips, cada chip aporta con un bit, de esta forma se puede extraer una palabra completa en un solo ciclo de lectura.

Cada fila de chips representa el ancho de una palabra, por ejemplo, una memoria con un ancho de palabra de 8 bits necesitará 8 chips separados. Todas las líneas están conectadas a una línea común llamada *línea de palabra*; en la figura 24, podemos observar que se tienen 16 filas, para poder seleccionarlas es necesario contar con una entrada de direcciones de 4 bits (2^4). Cada columna está conectada a un circuito Entrada-Datos/Detección (Sense/Write) que tiene dos líneas llamadas líneas de *bit* usadas para las operaciones de escritura o lectura.

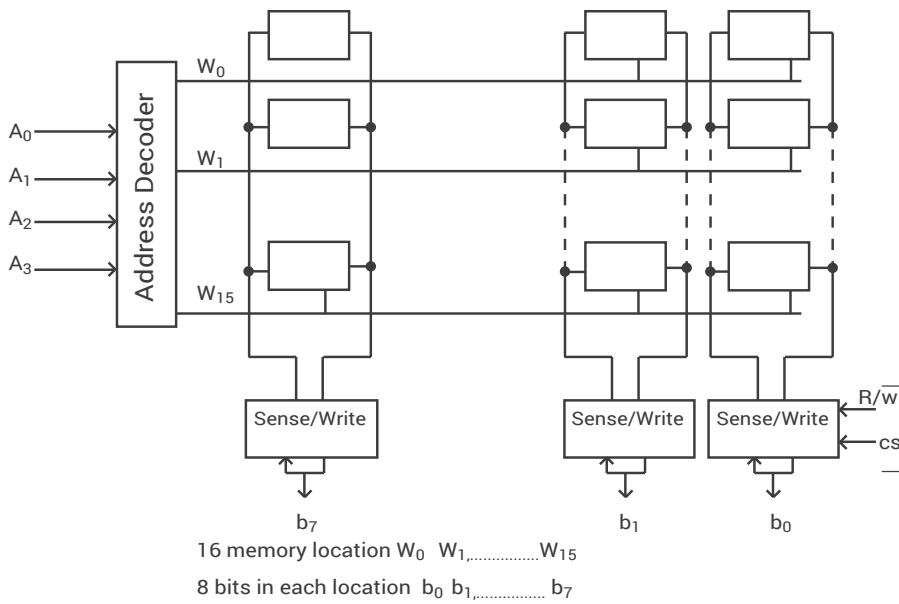


Figura 24. Estructura simplificada de una memoria RAM organizada en 16 filas de palabras y 8 bits de longitud.

Fuente: Deka (2006).

Adicional a las líneas de dirección y datos, se requieren dos líneas de control R/W para diferenciar las operaciones de lectura y escritura y CS (*chip select*) de selección de chip en el caso que se deba seleccionar un determinado chip.

Existen otras memorias en las cuales varias palabras pueden organizarse en una fila, como se puede observar en la figura 25.

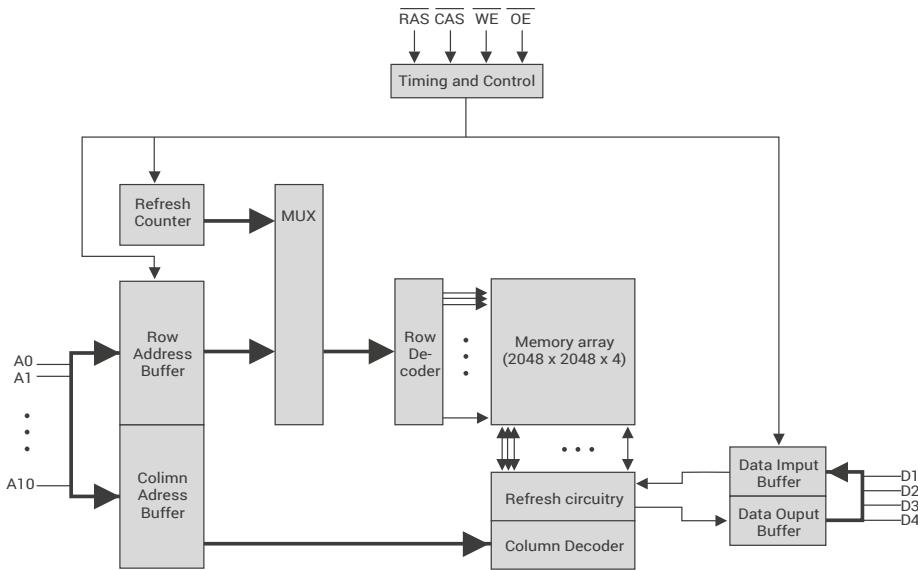


Figura 25. DRAM típica de 16Mb (4M x4)

Fuente: Stallings (2007).

En este caso se forman matrices de memoria, en la figura 25, tenemos una memoria de 16 Mb con un ancho de palabra de 4 bits, formando una matriz de 2048 x 2048 x 4, es decir, 4 matrices cuadradas de 2048 x 2048. Para poder elegir una palabra será necesario ingresar la dirección de la fila y columna en el *buffer* de dirección. Para esto es necesario suministrar direcciones de 11 *bits* de longitud ($2^{11} = 2048$). Ahora describamos otro ejemplo similar: se tiene una memoria de 4 K con cada palabra de 8 *bits* formando una matriz de 64 x 64 x 8, es decir, 8 matrices cuadradas de 64 x 64 elementos. Para elegir cualquier palabra será necesario suministrar direcciones de 12 *bits*, debido a la configuración en forma de matriz se divide la dirección de 12 *bits* en dos grupos de 6 *bits* que aportan con la dirección de las filas y las columnas. De esta manera podemos acceder a una palabra en particular dentro la memoria. En este caso la entrada y salida de datos estará conectada a un registro de 8 bits. Le invitamos a que analice lo siguiente ¿Cómo cambiaría la figura 25 considerando el ejemplo este segundo ejemplo?

Ahora analice la figura 26, en ella se muestra la organización de una RAM de 256 KB, cada palabra tiene 8 bits (1 byte). Para decodificar las filas y columnas que son matrices de 512 bits x 512 bits es necesario que el registro MAR (Memory Address Register) tenga dos secciones de 9 bits los primeros nueve bits para direccionar las columnas y los otros 9 bits para las filas, recordar que $2^9 = 512$. Así de cada bloque de 512 x512 se obtiene un solo bit. Todos los 8 bloques producirán un bit que en el MBR (Memory Buffer Register) se reflejará como una palabra completa de 8 bits.

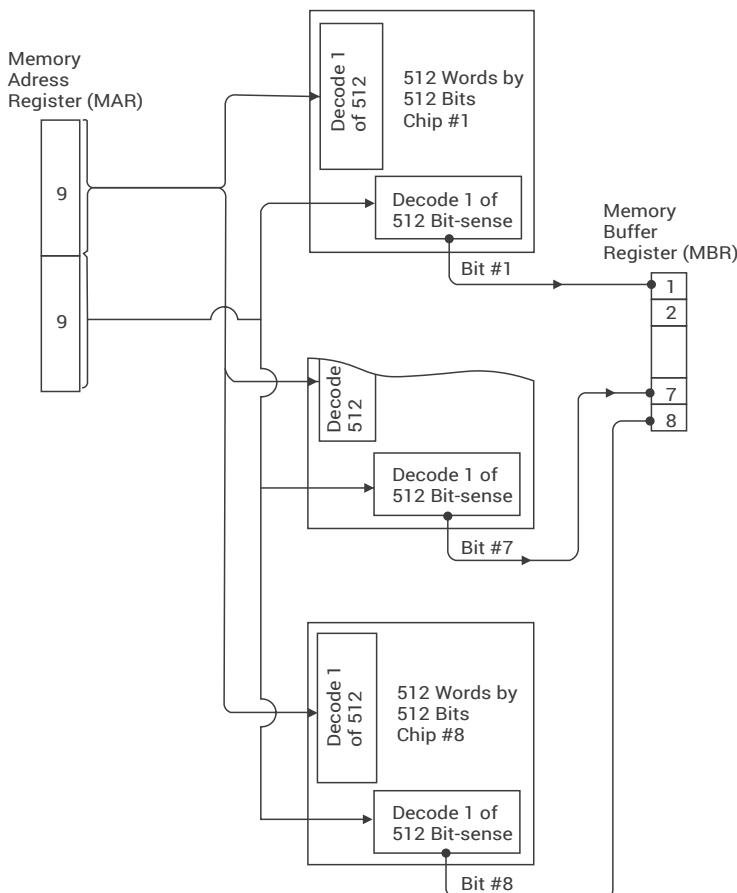


Figura 26. Organización de un bloque de 256 KB.

Fuente: Stallings (2007).

Para diseñar un sistema de memoria completo es necesario agrupar algunos chips de memoria. Por ejemplo, si necesitamos diseñar un sistema de 16 K x 16. Podemos agrupar 4 chips de 4 K x 8, como el presentado en el ejercicio anterior. Para el requerimiento de 16 K se necesitan direcciones de 14 bits, para cada chip de 4 K se necesitan direcciones de 12 bits. Es decir, para seleccionar una palabra de cada chip de 4 K necesitamos solo 12 bits, esto nos deja 2 bits libres de la dirección de 16 K, estos dos bits libres se utilizan para seleccionar uno de entre cuatro chips que tiene el sistema a través de la línea de control CS (*Chip Select*). La figura 27, nos muestra este sistema gráficamente.

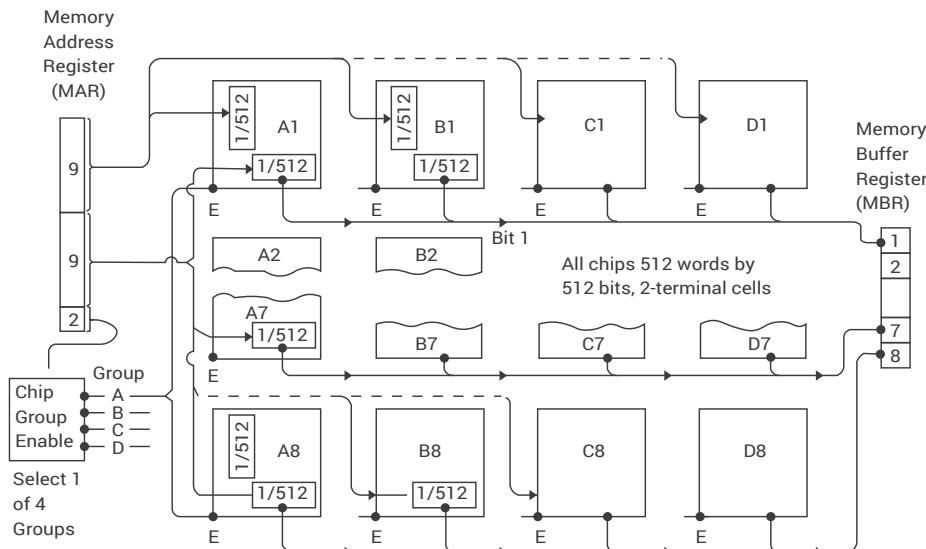


Figura 27. Organización de un módulo de RAM completo de 1 MB.
Fuente: Stallings (2007).

Hemos concluido con el estudio de la unidad 3 sobre los sistemas de memoria, en la unidad se ha discutido solo la memoria principal y la caché.

Para mejorar su aprendizaje es recomendable que revise los conceptos presentados en esta unidad leyendo el capítulo 4 del

texto *Arquitectura de computadoras* (Stallings, 2007), también puede recurrir a un resumen del capítulo, elaborado por la Universidad Politécnica de Madrid (UPM) en el siguiente enlace: [Resumen memoria caché de la UPM](#)

Las tecnologías de memoria de menor jerarquía como la memoria secundaria (disco duro) no se abordan en este texto; sin embargo, le invitamos a profundizar este conocimiento leyendo el capítulo 6 del texto *Arquitectura y organización de computadores* (séptima edición).



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Análisis de recurso audiovisual sobre RAM y Registros.

- **Procedimiento**

El siguiente video es de carácter instruccional, obsérvelo con atención y no dude en detenerse en algún momento para comprender los gráficos y esquemas que se observan, el fin de esta actividad es que comprenda la jerarquía de memoria, registros y memoria RAM, y, lo más importante: Cómo se puede almacenar un bit con compuertas lógicas desde el punto de vista del hardware.

VIDEO: CrashCourse. (2017). *Registers and RAM*. [video]. Recuperado de [Registers and RAM: Crash Course Computer Sciense #6](#)

Luego de observar el video invito a que responda las siguientes preguntas:

¿Cuál es la diferencia entre registros y memoria RAM?

¿Cómo se construye una unidad de memoria (1 bit)?

Actividad 2:

- **Actividad de aprendizaje**

Ejercicio de investigación: Almacenamiento en la nube.

- **Procedimiento**

Considere la pirámide de jerarquía de memoria estudiada en la unidad 3, no consta en ella la tecnología de almacenamiento que usted seguro alguna vez ha utilizado: El almacenamiento en la nube. Realice una nueva pirámide de memoria incluyendo al almacenamiento remoto o “en la nube”. Compártala con sus compañeros usando la wiki que se habilitará para este propósito en el entorno virtual de aprendizaje.

Actividad 3:

- **Actividad de aprendizaje**

Ejercicio de función de correspondencia:

- **Procedimiento**

Revise la sección 2.3 del texto base, grafique el ejercicio que se describe en dicha sección, realice dos tablas, una para simular la memoria principal y la segunda para la memoria caché, adjunte su propuesta en la wiki que se habilitará para este propósito en el entorno virtual de aprendizaje.

Actividad 4:

- **Actividad de aprendizaje**

Autoevaluación

- **Procedimiento**

Realice la autoevaluación 3 a continuación

Le invitamos a realizar la autoevaluación 3 para que ponga a prueba lo aprendido. De igual forma, puede exponer sus dudas al docente.



Autoevaluación 3

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. ¿Cómo funciona el sistema memoria caché/principal?
 - a. La caché extrae una copia parcial de la memoria principal para ofrecer mayor velocidad de acceso al CPU.
 - b. La memoria principal utiliza la caché como almacenamiento extra.
 - c. La memoria caché transfiere una copia de sus direcciones a la memoria principal para que esta se comunique con el CPU.
2. ¿Cuál es el objetivo de incluir una memoria caché en un sistema?
 - a. Ofrecer una compuerta de acceso directo a los dispositivos de E/S.
 - b. Lograr una velocidad de acceso mayor que la memoria principal.
 - c. Interactuar con el bus PCI.
3. ¿Si la dirección que busca el CPU no se encuentra en la caché, qué acción se lleva a cabo?
 - a. Se ejecuta una instrucción que se encuentre en una dirección de memoria de la caché.
 - b. Se busca el bloque que contiene la dirección en la memoria principal y este se transfiere a la caché.
 - c. Se desconecta la caché y la memoria principal pasa a conectarse directamente con el procesador.

4. ¿Cuál es la relación entre tiempo de acceso y coste de una memoria?
 - a. A menor tiempo de acceso menor costo.
 - b. A mayor tiempo de acceso mayor costo.
 - c. A mayor tiempo de acceso menor costo.
5. Elija el literal que indique la relación correcta entre tamaño y densidad de memoria con respecto al costo.
 - a. A menor tamaño y mayor densidad, mayor costo.
 - b. A menor tamaño y mayor densidad, menor costo.
 - c. A mayor tamaño y menor densidad, mayor costo.
6. ¿Para qué se necesita contar con una «jerarquía de memoria» en un sistema?
 - a. Para asignar un tipo de registro diferente a cada nivel de memoria.
 - b. Para aprovechar el bajo coste por bit de los medios con mayor capacidad y la velocidad de los medios más costosos, pero más rápidos.
 - c. Para almacenar grandes archivos como fotografías en los medios con menor tiempo de acceso.
7. Si se hace corresponder cada bloque de la memoria principal a solo una línea posible de caché, se afirma que el tipo de función de correspondencia es:
 - a. Correspondencia asociativa.
 - b. Correspondencia directa.
 - c. Correspondencia por conjuntos.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

8. ¿Cuál es el algoritmo de sustitución más efectivo para una memoria caché?
- a. LFU, Least Frequently Used.
 - b. FIFO, First in, First out.
 - c. LRU, Least Recently Used.

9. Indique al menos dos aplicaciones de las ROM.
-
-
-

10. ¿Qué diferencias existen entre las memorias EPROM, EEPROM y Flash?
-
-
-

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultados de aprendizaje 7 y 8

- Conoce la organización de un procesador clásico y compararlo con un procesador moderno.
- Comprende el papel de los registros dentro del procesador desde el punto de vista de cantidad, ancho de palabra y especialización.

Contenidos, recursos y actividades recomendadas



Semana 4



Unidad 4. Set de instrucciones y direccionamiento

4.1. Modos de direccionamiento

Los modos de direccionamiento permiten dar flexibilidad al campo de instrucciones cuando necesitan referirse a un operando.

4.1.1. Direccionamiento Inmediato

La forma más simple de direccionar es el direccionamiento inmediato, en el cual el operando se encuentra presente en la misma instrucción.



Figura 28. Direccionamiento inmediato.

Una vez que se capta la instrucción no se necesita recurrir de nuevo a memoria para extraer el operando ya que está implícito en la misma instrucción. Una ventaja de este tipo de direccionamiento es que se requieren menos ciclos de reloj para realizar las operaciones. Una desventaja es que la longitud del operando está restringida por la longitud de la instrucción.

4.1.2. Direccionamiento directo

Una forma simple de sobrelevar la desventaja del direccionamiento anterior, es que el contenido de la instrucción en lugar de contener al operando referencia una dirección de memoria donde se encuentre el operando. Esto permite que el operando pueda ser de mayor longitud al poder utilizar todo el ancho de palabra de la línea de memoria.

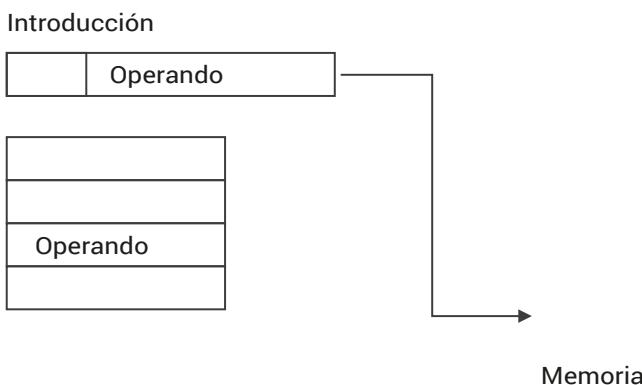


Figura 29. Direccionamiento directo.

Fuente: Stallings (2007).

4.1.3. Direccionamiento indirecto

En algunos casos el espacio de la instrucción destinado a direccionar un contenido de memoria es menor a la longitud de palabra, lo que limita el número de direcciones que se puede referenciar en memoria. Una solución es que el campo de dirección refiera una dirección de memoria que a su vez contenga la dirección completa del operando.

Introducción

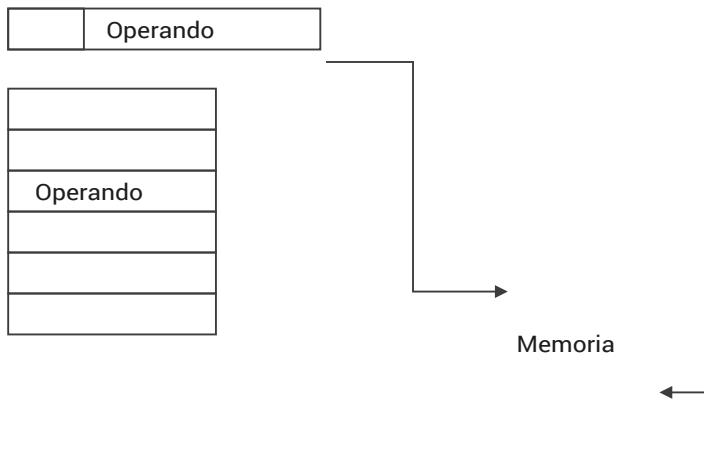


Figura 30. Direccionamiento indirecto.

Fuente: Stallings (2007).

4.1.4. Direccionamiento de Registros

Una instrucción puede contener la referencia a un operando que se encuentra en un registro, esto provoca que no se refiera a memoria, optimizando en algunos casos el ciclo de instrucción.

Introducción

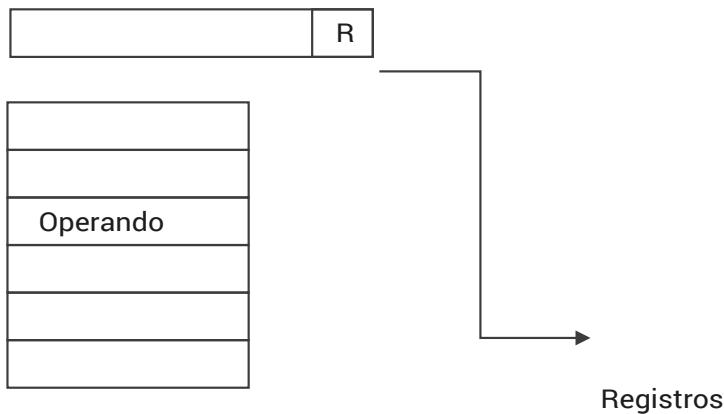


Figura 31. Direccionamiento de registros

Fuente: Stallings (2007).

4.1.5. Direccionamiento Indirecto con Registro

En este caso se tiene una estructura similar al direccionamiento indirecto, con excepción de que el campo dirección se refiere a un registro en lugar de una posición de memoria. El campo del registro no contiene el operando sino la dirección del operando en memoria.

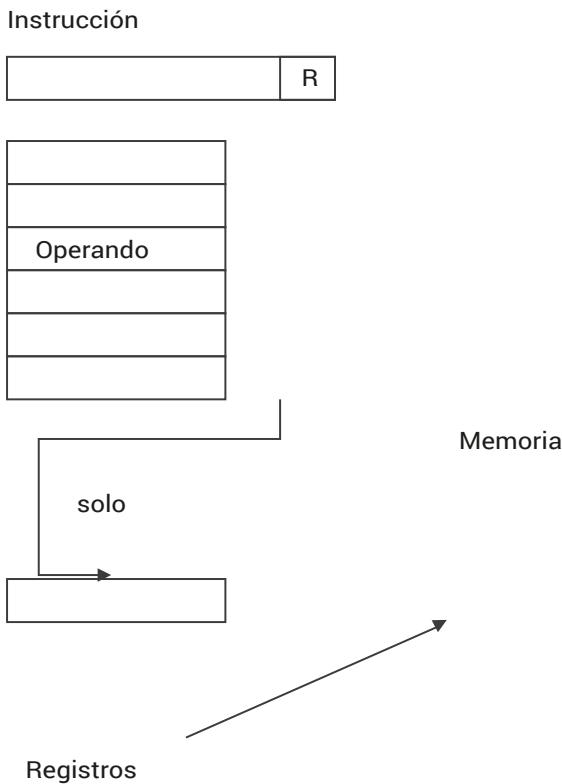


Figura 32. Direccionamiento indirecto con registros.

Fuente: Stallings (2007).

4.1.6. Direccionamiento con desplazamiento

El modo de direccionamiento con desplazamiento combina el direccionamiento directo con el direccionamiento indirecto con registro. La dirección efectiva del operando en memoria se logra sumando el contenido de dirección de la instrucción con el contenido de la referencia del registro, como se observa en la figura 33.

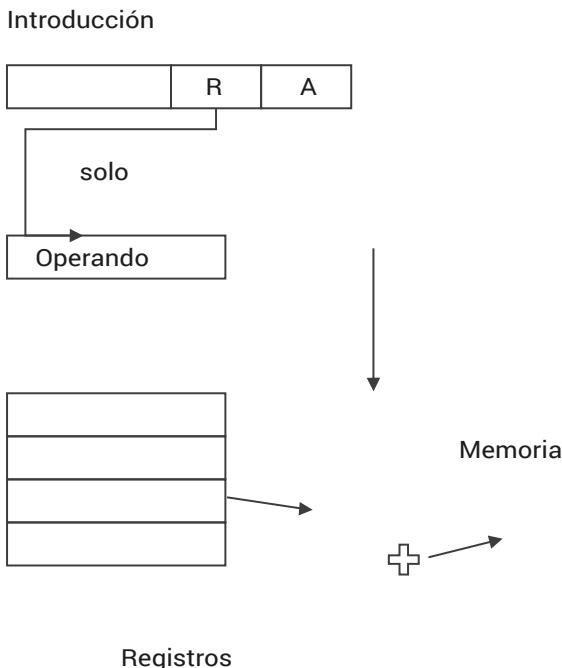


Figura 33. Direccionamiento con desplazamiento.

Fuente: Stallings (2007).

4.2. Instrucciones máquina

Las operaciones de un procesador están determinadas por las instrucciones que ejecuta, estas instrucciones se conocen como *instrucciones máquina*. El conjunto de todas las instrucciones máquina que el procesador es capaz de ejecutar se conoce como *set de instrucciones*.

Cada instrucción debe contener la información requerida para la ejecución en el procesador, los elementos que componen a una instrucción son:

- **Código de operación (CODOP):** indica la operación a realizar (add, store etc.) La operación se especifica mediante un código binario.

- **Referencia a operandos:** este campo hace referencia a los operandos que serán las entradas para la operación, pueden implicar uno o más.
- **Referencia al operando de destino:** este campo hace referencia a la dirección donde se guardará el resultado de la operación en el caso de que exista

4.2.1. Referencia a la siguiente instrucción

Este campo indica al procesador la dirección donde se encuentra la siguiente instrucción a captar. Solo en el caso de que la siguiente instrucción no siga a la que se acaba de ejecutar.

Los pasos que sigue la ejecución de una instrucción son los siguientes, el ciclo parte desde la parte inferior izquierda *Cálculo de dirección de la instrucción* (figura 34).

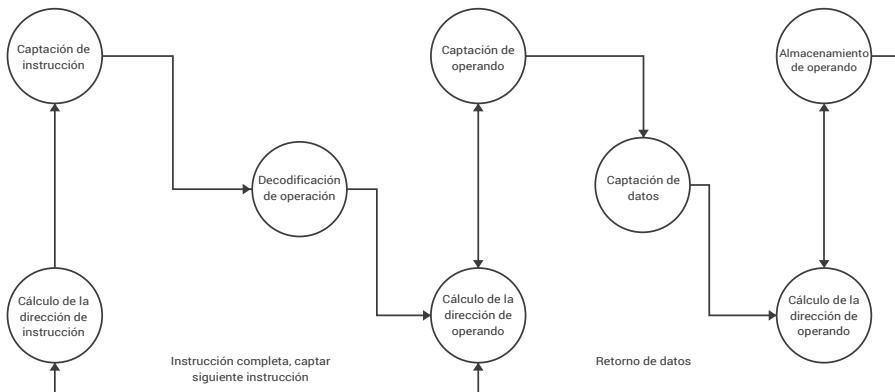


Figura 34. Pasos que intervienen en la ejecución de la instrucción.
Fuente: Stallings (2007).

4.2.2. Representación de una instrucción

Cada instrucción se representa por medio de una cadena de bits. La instrucción se divide en varios campos que indican los elementos de

una instrucción como lo vimos arriba. Por ejemplo, una instrucción de 32 bits, como se muestra en la figura, podría dividirse en tres campos:

Operando 1 8 bits	Operando 2 12 bits	Operando 3 12 bits
----------------------	-----------------------	-----------------------

Figura 35. Representación de una instrucción.

Fuente: Stallings (2007).

Con una longitud de 8 *bits* para el CODOP (Código de Operación) tendríamos $2^8 = 256$ posibles códigos de operación. Con 12 *bits* para cada operando podemos referenciar hasta $2^{12} = 4096$ diferentes operandos.

Los CODOP son leídos por procesador en binario; sin embargo, para es complicado para los programadores trabajar en este formato, es así que comúnmente se utiliza una representación simbólica de estos códigos, se los conoce como nemotécnicos.

Por ejemplo:

ADD Suma

SUB Restar

MULT Multiplicar

DIV Dividir

LOAD Cargar datos desde la memoria al CPU

STORE Guardar datos en memoria desde la CPU

Los operandos también se representan simbólicamente. Por ejemplo, la instrucción.

DIV R, X

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Lo que significa dividir el dato contenido en la posición X de memoria para el contenido del registro R y poner el resultado en el mismo registro R, ($R \leftarrow R/X$)

Así, es posible escribir un programa en forma simbólica, cada símbolo CODOP tiene una sola representación de bits, cada símbolo de operando tiene su ubicación en memoria.

4.2.3. Tipos de instrucciones

Las instrucciones se pueden clasificar en cuatro grupos, a saber:

- Procesamiento de datos: instrucciones aritméticas y lógicas, provee a la computadora la capacidad de procesar datos numéricos y lógica booleana.
- Almacenamiento de datos: se refiere a las instrucciones encerradas en el movimiento de datos entre memoria y registros del procesador.
- Transferencia de datos: instrucciones que permiten transferir datos o programas desde el exterior a la memoria o viceversa.
- Control: instrucciones dedicadas a comprobar el valor o resultado de una operación o estado de un conjunto de instrucciones.

4.2.4. Número de direcciones

Dependiendo de la longitud de una instrucción el diseñador podría considerar variar el número de direcciones que conforman una instrucción. La mayoría de operaciones aritméticas o lógicas necesitan uno o dos operandos. Así, la instrucción necesita máximo dos direcciones a los operandos. El resultado de una operación se debe guardar, esto sugiere una tercera dirección que indica el destino del resultado. Luego de ejecutar la operación se necesita decirle al

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

procesador la dirección de la siguiente operación a ejecutar, una cuarta dirección es necesaria.

El ejemplo anterior sugiere que necesitaríamos instrucciones de cuatro direcciones; sin embargo, esto extendería demasiado la longitud de *bits* necesaria para cada instrucción. En las computadoras actuales la mayoría maneja instrucciones de dos o tres direcciones, la dirección de la siguiente instrucción suele estar implícita en el contador de programa.

CODOP

Instrucción de cero direcciones

CODOP Dirección

Instrucción de cero direcciones

CODOP Dirección 1 Dirección 2

Instrucción de dos direcciones.

CODOP Dirección 1 Dirección 2 Dirección 3

Instrucción de tres direcciones.

Consideré la siguiente expresión matemática a evaluar:

$$Y = (A + B) / (C * D * E)$$

Con instrucciones de una sola dirección se necesita que la segunda dirección esté implícita, generalmente esta segunda dirección es el Acumulador (AC). El acumulador puede guardar uno de los operandos o ser el destino del resultado de la operación. La expresión de arriba se resolvería de la siguiente forma con instrucciones máquina.

Instrucción	Descripción
LOAD C	$AC \leftarrow C$
MULT D	$AC \leftarrow AC * D$
MULT E	$AC \leftarrow AC * E$
STORE Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
ADD B	$AC \leftarrow AC + B$
DIV Y	$AC \leftarrow AC / Y$
STORE Y	$Y \leftarrow AC$

Figura 36. Algoritmo en formato de instrucción máquina, una dirección.
Fuente: elaboración propia.

En el caso de las instrucciones de dos direcciones podemos prescindir del acumulador ya que la segunda dirección es explícita, en esta misma dirección se guardará el resultado. Se procede de la siguiente manera tabla.

Instrucción	Descripción
MOVE Y, A	$Y \leftarrow A$
ADD Y, B	$Y \leftarrow Y + B$
MOVE X, C	$X \leftarrow X$
MULT X, D	$X \leftarrow X * D$
MULT X, E	$X \leftarrow X * E$
DIV Y, X	$Y \leftarrow Y / X$

Figura 37. Algoritmo con instrucciones de dos direcciones.
Fuente: elaboración propia.

Ahora, cuando se trata de instrucciones de tres (3) direcciones cada instrucción tiene dos operandos y una tercera dirección para guardar el resultado.

Instrucción	Descripción
ADD Y, A, B	$Y \leftarrow A + B$
MULT X, C, D	$X \leftarrow C * D$
MULT X, X, E	$X \leftarrow X * E$
DIV Y, Y, T	$Y \leftarrow Y / X$

Figura 38. Algoritmo con instrucciones de tres direcciones.
Fuente: elaboración propia.

Cuando se trata de cero direcciones todas las instrucciones están implícitas, se usa una pila. Por ejemplo, para realizar la operación $Y = A + B$ se procede así.

Instrucción
push A
push B
add
pop Y

Figura 39. Instrucciones de cero direcciones.

Fuente: elaboración propia.

Luego de este análisis, se puede obtener algunas conclusiones importantes con respecto al número de direcciones óptimo para las instrucciones de un procesador.

Mayor número de direcciones: instrucciones más complejas, se necesitan más registros, pero, los programas serán más cortos, debido a la reducción de las instrucciones necesarias.

Menos direcciones: instrucciones menos complejas, más instrucciones por programa, en ciertos casos esto puede implicar que el ciclo de instrucción sea también más simple y, por ende, más rápido.

El diseño del repertorio de instrucciones es muy importante debido a que de esto depende toda la arquitectura de la computadora, ancho de buses, nivel de complejidad de la ALU, hardware de la unidad de control, etc. Los aspectos más importantes para tomar en cuenta son:

- El número de operaciones: cuántas y cuáles operaciones es capaz de realizar el procesador.
- Los tipos de datos: con qué datos es capaz de realizar operaciones el procesador.

- El formato de instrucción: número de direcciones, longitud de palabra, etcétera.
- El número de registros que pueden ser usados por las instrucciones.
- Y el tipo de direccionamiento.

Con respecto al **número de operaciones** que puede realizar, debemos considerar que las mismas se deben realizar en *hardware*, en circuitos dentro del procesador, es así que mientras más operaciones y más complejidad en las mismas la ALU debe ser más compleja y el procesador se vuelve más complejo. En general los tipos de operaciones que se utilizan son:

- **Transferencia de datos.** Estas instrucciones especifican las posiciones de los operandos en la fuente (por ejemplo: memoria) y destino de los datos, además de la longitud de los datos. Pueden haber instrucciones que especifican la fuente (memoria o registro) por ejemplo la IBM 370 o instrucciones que especifican un solo tipo de operación y la fuente o destino de los datos depende del formato de direcciones, por ejemplo la VAX.
- **Aritméticas.** Estas instrucciones especifican las operaciones básicas, suma, resta, multiplicación y división. Principalmente estas instrucciones pueden trabajar con datos de tipo entero con signo y en algunos casos directamente con punto flotante. También incluyen instrucciones de tipo incremento ($a++$), decremento ($a-$) o negación ($-a$).
- **Lógicas.** Las instrucciones lógicas realizan operaciones booleanas como AND, OR, NOT y X-OR. En general trabajan bit a bit. Además, pueden incluir algunas instrucciones para desplazamiento de bits hacia la izquierda o derecha.

- **Conversión.** Estas instrucciones permiten convertir de un formato a otro, por ejemplo, de Decimal a Binario, o de binario a BCD, etcétera.
- **I/O.** son instrucciones muy específicas, debido a que las computadoras modernas suelen utilizar un procesador de E/S que se encarga de estas operaciones. El procesador suele incluir solo algunas instrucciones en forma de órdenes.
- **Control del sistema.** Son instrucciones reservadas principalmente por el sistema operativo o cuando el programador pueda tener acceso privilegiado, debido a que estas instrucciones modifican parámetros de funcionamiento de la máquina por ejemplo protecciones de memoria, o alteración de un registro de control.
- **Transferencia de control.** Estas instrucciones permiten interrumpir el curso normal de un grupo de instrucciones, es decir interrumpir la secuencia natural. Por ejemplo, saltar al final del programa si el resultado de una operación es cero. O por ejemplo ejecutar las mismas instrucciones cientos de veces en un ciclo repetitivo (*Loops*).

Los tipos de operandos con que el procesador trabaja pueden agruparse en cuatro grupos principales:

- Direcciones: indican la dirección de una posición de memoria. Generalmente son cadenas de *bits* en formato binario sin signo y de tipo entero.
- Números: las computadoras modernas pueden trabajar con un extenso número de formatos; sin embargo, se pueden agrupar en dos tipos: enteros y en punto flotante.

Los números enteros se representan en formato binario con signo o en complemento a dos. Para expresar cantidades muy pequeñas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

o muy grandes se utiliza el formato de punto flotante. También se puede encontrar el formato decimal empaquetado, que no es más que la utilización del código BCD para expresar cada dígito decimal.

Los caracteres se refieren a la forma de expresar en la computadora información tipo texto o cadenas de caracteres. Como se conoce, la computadora trabaja con bits. El texto o caracteres se codifican por medio de ASCII (*American Standard Code for Information Interchange*). Tiene una longitud de 7 bits.

Los datos de tipo lógico es información en forma de verdadero o falso, generalmente 1 es verdadero y 0 es falso.

En la figura 40, podemos apreciar los distintos formatos de números y longitudes que soporta la arquitectura de un Pentium x86.

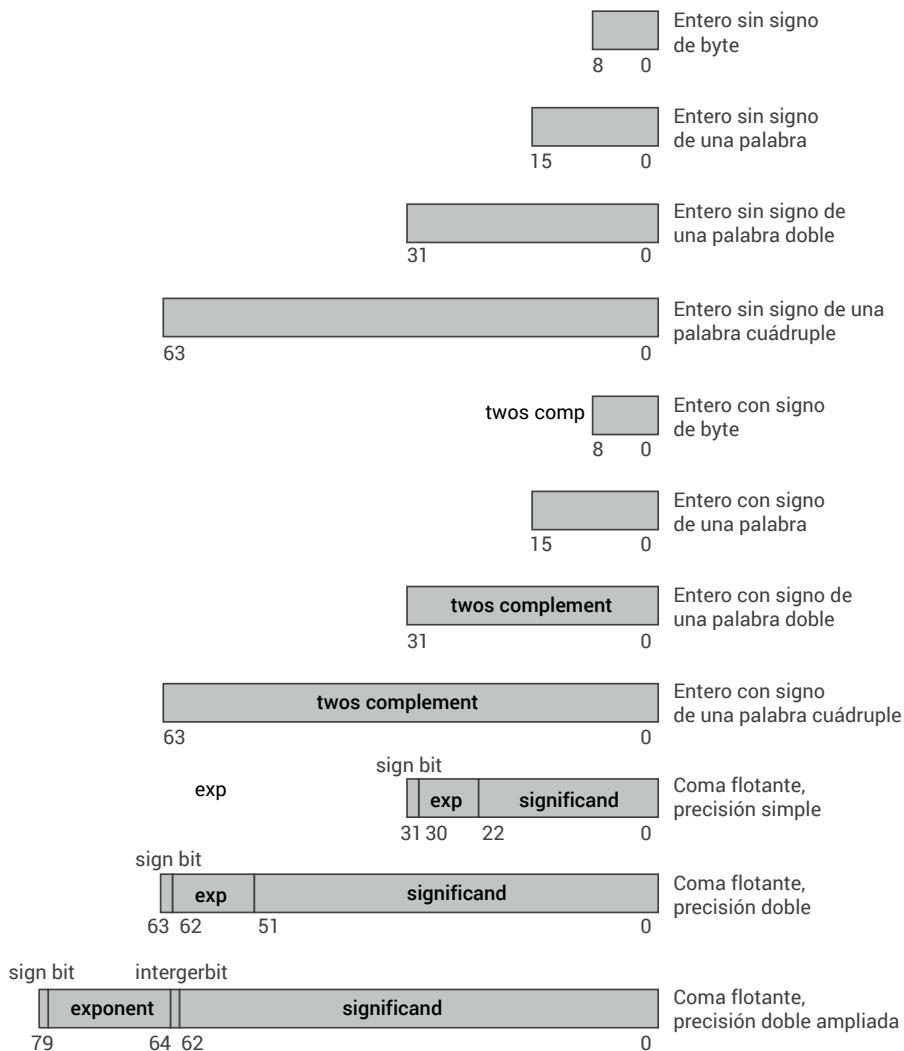


Figura 40. Distintos formatos que soporta la arquitectura Pentium x86.
Fuente: Stallings (2007).

4.2.5. Formato de instrucciones

En algunos procesadores todas las instrucciones tienen la misma longitud, en otros pueden tener distintas longitudes dependiendo del tipo de instrucción. Tener un formato de instrucciones de

un solo tamaño facilita el trabajo de la unidad de control ya que se puede decodificar más rápido, pero en algunos casos se desperdicia espacio. Para evitar esto se suele variar el tamaño de las instrucciones. Por ejemplo:

1 Palabra de longitud			
Instrucción 1		Instrucción 2	
Instrucción 1	Instrucción 2	Instrucción 3	Instrucción 4
Instrucción			

Figura 41. Distintos formatos de instrucción.

Fuente: elaboración propia.

Generalmente existe una correlación entre la longitud de la instrucción y la longitud de palabra de transferencia memoria, pueden ser de la misma longitud o al menos ser una de los dos múltiplos de la otra.

El diseñador debe considerar también que mientras más CODOP se tenga mayor número de *bits* serán necesarios para el campo CODOP en la instrucción, restando el espacio disponible para direcciones. Además de esto, se debe balancear los recursos con los métodos de direccionamiento y el número de operandos y direcciones en la instrucción. Una buena alternativa es usar lo máximo posible los registros para el direccionamiento. El uso de los registros como direcciones implícitas (por ejemplo: el acumulador) ayudan a reducir el ancho de las instrucciones además de ser de mayor velocidad de acceso. Para finalizar estas consideraciones, se debe agregar que el número de registros disponibles es otro asunto de balance de recursos, tradicionalmente se ha tenido de 8 a 16 registros de uso múltiple; sin embargo, la tendencia actual es tener bancos de registros especializados.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Ejercicio

- **Procedimiento**

Realice el ejercicio propuesto

Según las instrucciones contenidas en el bloque de memoria del recuadro. ¿Cuál es el número que se guardará en la dirección 407 al finalizar la secuencia de instrucciones?: Los CODOPs son los siguientes:

0001: Cargar AC desde memoria.

0002: Sumar a AC un dato de memoria.

0003: Almacenar AC en memoria.

0004: Dividir AC por el dato desde memoria.

0005: Multiplicar AC por el contenido en memoria.

Memoria	
Dirección	Contenido
200	1 406
201	2 407
202	3 407
203	1 407
204	5 408

Memoria	
Dirección	Contenido
205	3 407
...	...
...	...
406	0004
407	0005
408	0002

Figura 42. Conjunto de instrucciones y datos en un bloque de memoria.
Fuente: elaboración propia.

Si seguimos en secuencia las instrucciones de las posiciones 200, 201, 202... descubrimos que el algoritmo es como sigue; Cargar a AC el contenido de la posición de memoria 406, luego sumar a AC el contenido de 407, guardar el contenido de AC en la posición 407, de nuevo, cargar el contenido de 407 en AC y finalmente multiplicarlo por la posición 408.

Actividad 2:

- **Actividad de aprendizaje**

Ejercicio

- **Procedimiento**

Realice el ejercicio propuesto

Realice el siguiente ejercicio para fortalecer su conocimiento sobre formato de instrucciones.

Restar dos vectores de números contenidos en las direcciones de memoria 000-009 y 010-019, vector A y Vector B respectivamente. La suma del contenido de cada posición se debe guardar en las posiciones de memoria 020-029.

La operación que se realizará es $A(i) - B(i) = C(i)$ siendo i el *bit* menos significativo LSB de la posición de memoria.

Los vectores se verían de la siguiente manera, las instrucciones tienen una longitud de 32 bits. 8 *bits* para el CODOP y 24 *bits* para el campo dirección

Vector A		Vector B		Vector C	
000	10	010	1	020	
001	9	011	2	021	
002	7	012	5	022	
003	5	013	3	023	
004	8	014	3	024	
005	6	015	4	025	
006	12	016	5	026	
007	11	017	3	027	
008	8	018	2	028	
009	8	019	5	029	
:		:		:	

Figura 43. Conjunto de datos en un bloque memoria.

Fuente: Elaboración propia

El conjunto de instrucciones de una dirección que calcularían lo pedido:

200	LOAD(009)
201	SUB(019)
203	STOR(029)
204	LOAD (D1)
205	SUB(D4)
206	STOR(D1)
207	STOR (200,8:31)

208	LOAD (D2)
209	SUB(D4)
210	STOR(D2)
211	STOR M (201,8:31)
212	LOAD (D3)
214	SUB (D4)
215	STOR (D3)
216	STOR (203,8:31)
217	JUMP + (200)

Figura 44. Conjunto de instrucciones secuenciadas en un bloque de memoria.

Fuente: Elaboración propia

Actividad 3:

- **Actividad de aprendizaje**

Autoevaluación.

- **Procedimiento**

Realice la autoevaluación de la unidad 4. Si tiene dudas, consulte de nuevo el texto guía, o el texto: *Stallings, W. (2007). Organización y arquitectura de computadoras, Madrid (España), PEARSON Prentice Hall.* De igual forma, puede exponer sus dudas al docente.



Autoevaluación 4

Arquitectura de Computadoras y Sistemas Operativos

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. ¿Cómo está conformada una instrucción máquina?
2. ¿En qué lugares pueden estar los operandos de origen?:
 - a. Memoria principal.
 - b. Registros del procesador.
 - c. E/S.
 - d. Todas las opciones.
3. ¿Cómo se clasifican a los tipos de instrucciones que realizan operaciones aritméticas o lógicas?
 - a. Procesamiento de datos.
 - b. Almacenamiento de datos.
 - c. Transferencia de datos.
 - d. Control.
4. ¿Cuántas direcciones contienen las instrucciones que utilizan el acumulador como registro temporal?
 - a. Una dirección.
 - b. Dos direcciones.
 - c. Tres direcciones.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. ¿Identifique una ventaja de utilizar instrucciones de pocas o una dirección?
 - a. Programas más complejos.
 - b. Menos ciclos de reloj para completar la tarea.
 - c. Requieren un procesador más simple.
6. Realice el siguiente ejercicio:

Escriba las instrucciones necesarias para realizar la siguiente operación con tres direcciones, según el repertorio de instrucciones de la tabla 4.

$$Y = (A * B + (C - D)) / (E - F)$$

Tabla 3. Repertorio de instrucciones.

1 Dirección	2 Direcciones	3 Direcciones
ADD M	ADD (X <- X+Y)	ADD (X <- Y+Z)
SUB M	SUB (X <- X-Y)	SUB (X <- Y-Z)
MUL M	MUL (X <- X*Y)	MUL (X <- Y*Z)
DIV M	DIV (X <- X/Y)	DIV (X <- Y/Z)
LOAD M	MOVE (Z <- Y)	MOVE (X <- Y)
STOR M		

7. Repita el ejercicio anterior pero esta vez con 2 direcciones, según el repertorio de la tabla 6.
8. Repita el ejercicio anterior pero esta vez con 1 dirección, según el repertorio de la tabla 6.
9. Compare las respuestas de los ejercicios 6, 7 y 8 y responda: ¿Qué programa necesita instrucciones más primarias?
10. ¿El programa resultante del ejercicio 6 (tres direcciones) necesitará mayor o menor tiempo de ejecución?

Ir al solucionario

Resultados de aprendizaje 9 y 10

- Conoce todas las etapas del ciclo de instrucción.
- Diferenciar entre ciclo de captación, ejecución e interrupción.

Contenidos, recursos y actividades recomendadas



Semana 5



Unidad 5. Unidad Central de Procesamiento

En la presente unidad abordaremos la descripción de la Unidad Central de Procesamiento (CPU), que no es más que el conjunto de registros, unidad aritmético-lógica, unidad de control y *buses* de datos.

La unidad central de procesamiento realiza las siguientes tareas.

- Captación de la instrucción: la CPU trae una instrucción desde memoria.
- Interpretación de la instrucción: la instrucción es decodificada para conocer qué operación se debe realizar.

- Captación de datos: luego de decodificar la operación puede ser necesario traer datos o los operandos desde la memoria.
- Procesamiento de datos: la operación conjuntamente con sus operandos en registros se ejecuta.
- Escritura de datos: el resultado de la ejecución de la operación puede grabarse en memoria o enviarse a un dispositivo de E/S.

Estas tareas son realizadas con ayuda de algunos componentes internos de la CPU:

- Los registros: guardan temporalmente la operación de la instrucción, operandos y de control por ejemplo el contador de programa (CP).
- La Unidad Aritmético-Lógica (ALU): es la encargada de realizar el proceso de cómputo y procesamiento de datos, esta unidad fue tratada en el capítulo 3.
- La Unidad de Control (*Control Unit - CU*): Controla el movimiento de datos e instrucciones hacia o desde la CPU, además controla el funcionamiento de la ALU.
- El CPU está conectado con el resto de los componentes por medio de un sistema de buses:
- Bus de datos: se usa para transmitir datos entre la memoria y la CPU.
- Bus de direcciones: se usa para acceder a una locación de memoria particular.
- Bus de control: se utiliza para transmitir señales de control a los diferentes componentes del sistema, generalmente estas señales son generadas por la CPU.

La figura 45, muestra un esquema de la interconexión de la CPU y el bus del sistema (control, datos y direcciones).

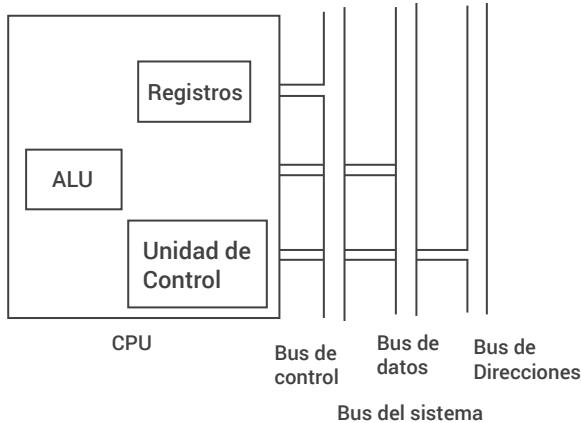


Figura 45. Buses y bus del sistema.

Fuente: Stallings (2007).

Dentro del chip de la CPU los componentes se interconectan a través de un *bus* interno, como lo podemos observar en la figura 46. La unidad de control se interconecta con los componentes a través de un *bus de control*.

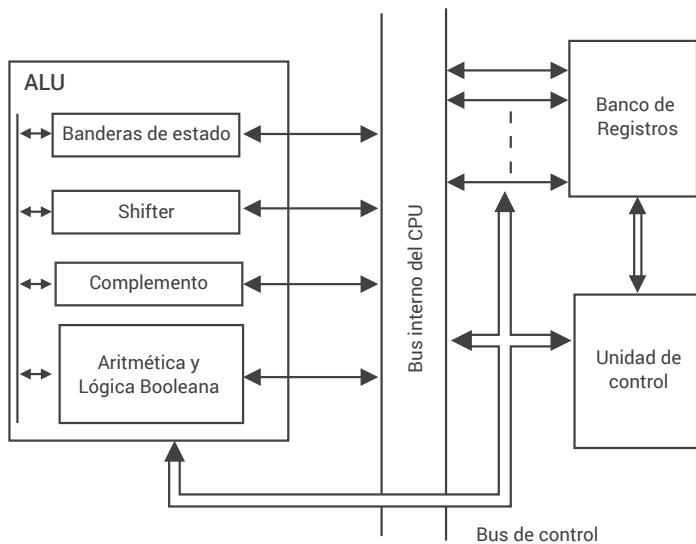


Figura 46. Estructura interna de la CPU.

Fuente: Stallings (2007).

5.1. Organización de los registros

La computadora utiliza un sistema de jerarquía de memoria. En la más alta jerarquía la memoria es más rápida, pequeña y más costosa, en esta jerarquía se encuentran los registros del procesador. Pueden organizarse en dos grupos:

- **Registros visibles al usuario:** estos registros pueden ser referenciados en las instrucciones escritas por programadores.
- **Registros de estado y control:** son usados por la unidad de control para controlar la operación de la CPU. También pueden ser utilizados por los sistemas operativos en modo privilegiado.

Registros visibles al usuario. Se subcategorizan en:

- Registros de propósito general: estos registros son muy flexibles y depende del programador usarlos para varias funciones, por ejemplo, para almacenar direcciones, instrucciones, etc.
- Registros de datos: solo pueden ser usados para guardar datos, no se pueden usar para almacenar operandos ni direcciones.
- Registros de direcciones: se los utiliza para guardar direcciones extraídas de las instrucciones.
- Códigos de condición: también llamados banderas. Son registros muy pequeños generalmente de un bit, que suelen acompañar al resultado de las operaciones, por ejemplo: acarreo, desborde aritmético (*overflow*), resultado cero, positivo o negativo, etcétera.

Registros de control

- Contador de programa (PC): este registro contiene la dirección de la siguiente instrucción a ser captada por la CPU, se incrementa en uno luego de cada ejecución de instrucción. Sin embargo, puede ser modificada por alguna instrucción de tipo salto condicional.
- Registro de instrucción (IR): guarda la instrucción recientemente captada desde memoria.
- Registro de dirección de memoria (MAR): contiene la dirección de una posición de memoria de donde se debe extraer información o donde se debe guardar el resultado de alguna operación. Este registro está conectado directamente al *bus* de direcciones.
- Registro temporal de memoria (MBR): contiene el dato a ser escrito en memoria, este registro está conectado directamente al *bus* de datos.

Además de estos registros pueden existir dentro de la CPU registros que no son visibles al usuario, como por ejemplo registros tipo *buffer* o los registros de la ALU que contienen los operandos temporalmente antes de ser computados.

5.2. Proceso de ejecución de un programa

En una computadora las instrucciones de un programa se almacenan en la memoria en secuencia. Para ejecutar el programa la CPU capta instrucción por instrucción y ejecuta la acción requerida por cada una de ellas. La captación se realiza en secuencia, excepto cuando una instrucción indique un salto. El CPU conoce cuál es la dirección de memoria donde se encuentra la instrucción a ejecutar consultando el registro PC (contador de programa). Luego de ejecutar la instrucción

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

el PC se incrementa en 1 o se carga con la dirección de la siguiente instrucción a ejecutar.

El proceso de captación de una instrucción consiste en captar los contenidos de la posición de memoria indicada por el PC e incrementar el PC en 1. Los contenidos de aquella posición se interpretan como una instrucción a ser ejecutada, entonces se guarda la instrucción en el registro de instrucción (IR).

Luego, el proceso de ejecución consiste en realizar la operación que exige la instrucción grabada en IR mediante la decodificación del CODOP contenido y la búsqueda de los operandos.

Estos dos procesos se conocen como ciclo de instrucción. En algunos casos las instrucciones pueden ser más largas que una palabra entonces el proceso de captación se puede repetir varias veces hasta extraer toda la instrucción contenida en varias posiciones de memoria.

La figura 46, muestra un esquema general del llamado “ciclo de instrucción”.

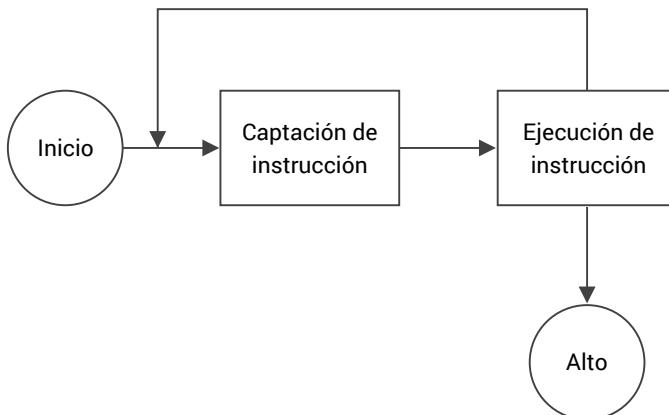


Figura 47. Ciclo de instrucción.

Fuente: Elaboración propia

El flujo de datos en el ciclo de instrucción se puede describir de la siguiente manera:

Supongamos una computadora con registros MAR, MBR, IR y PC. Cuando se lleva a cabo el ciclo de captación el PC contiene la dirección de la instrucción a ejecutar, el contenido de PC se carga en MAR, que la transmite al *bus de direcciones*. La unidad de control lee la dirección en el *bus de direcciones* e inmediatamente ordena una lectura de memoria en esa dirección. Memoria pone el contenido de la dirección en el *bus de datos*. El registro MBR se carga con los datos que se encuentran en el *bus de datos*. El MBR envía estos datos al registro de instrucción IR para su ejecución. Observe el flujo de datos gráficamente en la figura 48.

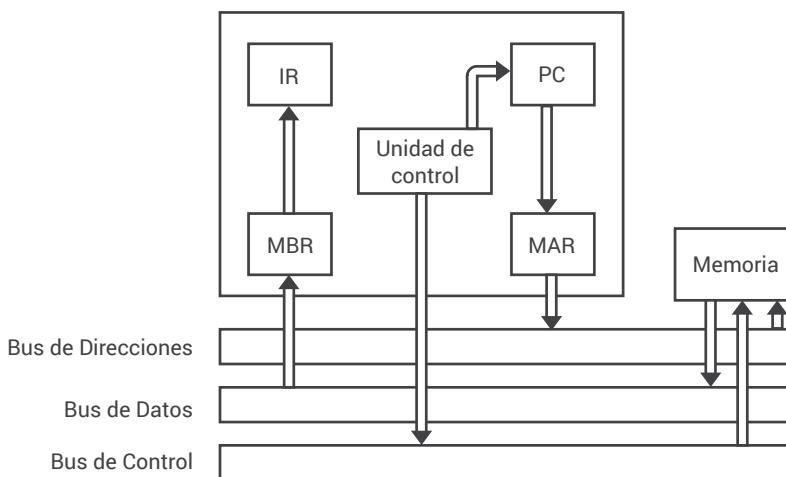


Figura 48. Flujo de datos.

Fuente: Elaboración propia

Las acciones que el procesador puede ejecutar se pueden agrupar en cuatro categorías, a saber:

- **Procesador-memoria.** Los datos pueden transferirse desde el procesador a la memoria o desde la memoria al procesador.
- **Procesador- E/S.** Los datos pueden transferirse hacia o desde un dispositivo periférico mediante un módulo de E/S.

- **Procesamiento de datos.** El procesador realiza operaciones aritméticas o lógicas con los datos.
- **Control.** Una instrucción puede especificar que la secuencia de instrucción ha sido alterada.

El ciclo de ejecución de una instrucción puede requerir más de una referencia a memoria. Además, en lugar de requerir datos desde memoria la instrucción puede indicar alguna operación de E/S. A continuación, se presenta un diagrama más completo del ciclo de instrucción (figura 49).

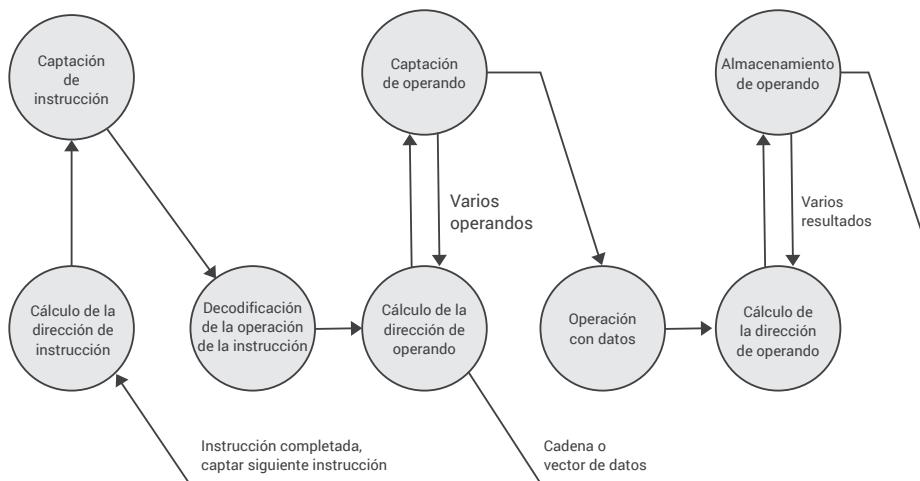


Figura 49. Diagrama de estados del ciclo de instrucción.

Fuente: Stallings (2007).

5.3. Gestión de interrupciones

Todas las computadoras de uso general necesitan ejecutar varias tareas a la vez, es así que es necesario que el procesador pueda ser interrumpido para atender otra tarea con mayor prioridad, por ejemplo, un dispositivo de E/S que se conecte o requiera datos. Las clases más comunes de interrupciones son las siguientes.

- Programa: estas interrupciones se pueden originar durante la ejecución de alguna instrucción, por ejemplo, un desborde aritmético, división inválida por cero, o un error en la dirección de memoria.
- Tiempo: se genera por un temporizador dentro del procesador, esto le permite al sistema operativo realizar algunas funciones basadas en tiempo.
- E/S: la interrupción es generada por un controlador de E/S. Para indicar conexión, finalización de una tarea o para indicar alguna condición de error.
- Falla de *hardware*: un error en el *hardware* producido por una falla de energía o error de paridad puede generar una interrupción en el procesador.

5.4. Unidad de control

Durante la ejecución de un programa se realiza un conjunto de pasos que componen el ciclo de captación-ejecución. Cada uno de estos pasos se realiza mediante pequeños procesos llamados microoperaciones. Estas operaciones son gestionadas por la unidad de control a través de señales de control. La figura 50, muestra la descomposición de las microoperaciones a partir de un programa.

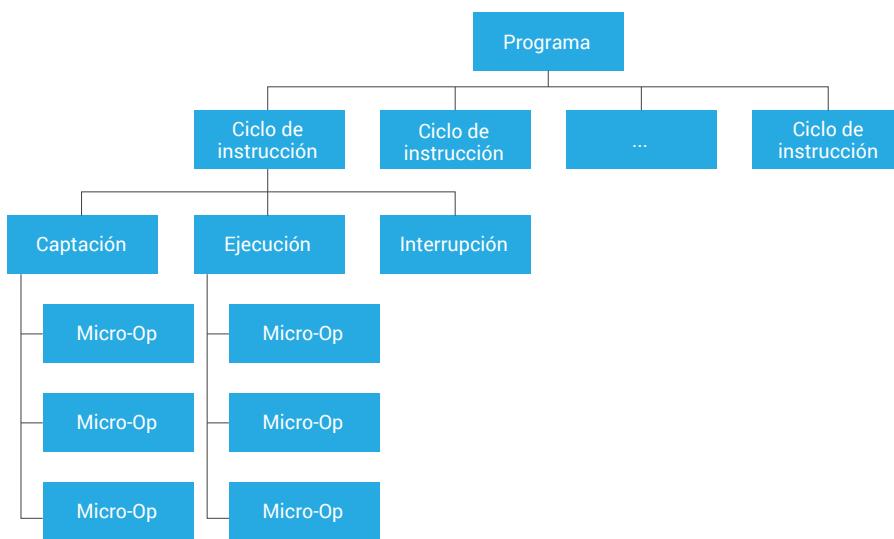


Figura 50. Descomposición de microoperaciones.

Fuente: Elaboración propia

Las microoperaciones generalmente están relacionadas con el movimiento de *bits* entre memoria, *buses* y registros.

En el ciclo de captación intervienen cuatro registros principales, recordemos su nombre y función en contexto de esta sección.

- Registro de dirección de memoria (MAR): está conectado al *bus* de direcciones y especifica la dirección a leer o escribir.
- Registro temporal de memoria (MBR): está conectado al *bus* de datos, y contienen el dato a escribir en memoria o el último dato leído de memoria.
- Contador de programa: contiene la dirección de la siguiente instrucción a captar.
- Registro de instrucción: almacena la última instrucción captada.

Los registros interactúan de la siguiente manera durante el proceso de **captación**.

- La dirección de instrucción a captar se encuentra en el contador de programa PC.
- Esta dirección se carga en el MAR y se ubica el *bus* de direcciones.
- La unidad de control genera un comando READ (lectura).
- El dato leído de memoria aparece en el *bus* de datos. Al estar en el *bus* de datos el dato se copia en el MBR.
- El PC incrementa en 1.
- El dato (instrucción) que se encontraba en MBR se mueve al IR. Dejando el MBR libre para un nuevo proceso de captación.

Estas microoperaciones se realizan en cada ciclo de reloj.

Tabla 4. *Macrooperaciones, ciclo de captación*

Tiempo	Micro-operación
T1	MAR \leftarrow PC
T2	MBR \leftarrow (memory) PC \leftarrow PC+1
T3	IR \leftarrow MBR

Las microoperaciones, aunque sencillas deben seguir una secuencia estricta, es muy importante que no se realice al mismo tiempo un proceso de lectura y escritura sobre el mismo registro. Por ejemplo, MBR \leftarrow memoria y IR \leftarrow MBR no pueden estar en el mismo ciclo de reloj.

En el proceso de incremento del PC+1, se utiliza la ALU, lo cual puede requerir microoperaciones adicionales.

El ciclo de interrupción necesita las siguientes microoperaciones.

- El contenido del PC se carga en MBR.

- El MAR se carga con la dirección guardada, el PC se carga con la dirección de la nueva rutina o programa.
- El contenido de MBR se pone en el *bus* de datos, este dato se guarda en memoria.

Tabla 5. *Microoperaciones ciclo de interrupción.*

Tiempo	Micro-Op
T1	MBR←PC
T2	MAR←Dirección guardada PC ← Dirección nueva rutina.
T3	Memoria ← MBR

Ahora, el ciclo de **ejecución** puede variar mucho dependiendo de la naturaleza de la operación a realizar. Por ejemplo, un ciclo de ejecución ADD puede descomponerse en las siguientes microoperaciones.

La instrucción ADD R, X suma los contenidos de la ubicación de memoria X al registro R, el resultado se guarda en el mismo registro R.

Tabla 6. *Microoperaciones ciclo de ejecución.*

Tiempo	Micro-Op
T1	MAR ← IR
T2	MBR←Memoria
T3	R ← R+MBR

La unidad de control debe ser diseñada de tal forma que responda al set de instrucciones del procesador. Estas señales deben ser enviadas a la ALU, registros, *buses* internos y externos, es decir la unidad de control está interconectada a todos los elementos del procesador.

Entre los tipos de microoperaciones que la unidad de control debe ser capaz de realizar están:

- Transferencia de datos entre registros.
- Transferencia de datos desde los registros al exterior.
- Transferencia de datos desde el exterior a los registros.
- Realizar operaciones aritméticas y operaciones lógicas.

Las principales funciones de la unidad de control son:

- Secuenciación: coordina las microoperaciones.
- Ejecución: ejecuta el código de operación de la instrucción.

Esto se realiza liberando señales de control a cada elemento:

- Reloj: la unidad de control ejecuta una microoperación por cada ciclo de reloj.
- Registro de instrucción: libera el código de operación (CODOP) a la unidad de control determinando que microinstrucciones se realizarán.
- Banderas (Flags): indican el estado de la CPU, almacenan los resultados de operaciones previas.

Interrupciones y ACK (control de recepción de datos): estas señales se generan desde el *bus* de control.

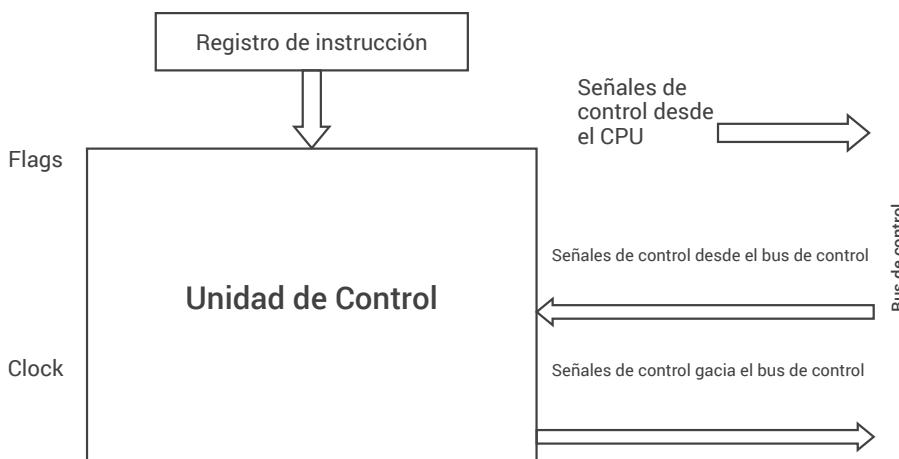


Figura 51. Señales de salida y entrada de la unidad de control

Fuente: Stallings, W. (2007)

En la figura 51, se muestran las señales de salida y entrada con las que trabaja la unidad de control, las señales de control internas al CPU mueven datos y activan funciones específicas como operaciones en la ALU. Las señales que van hacia el exterior se conectan con la memoria o los módulos de entrada y salida.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Ejercicio sobre unidad de control.

- **Procedimiento**

Revise la sección 5.4 sobre la unidad de control. Realice el ejemplo del proceso de instrucción ADD, ahora, proponga usted un conjunto de microinstrucciones que la unidad de control debe realizar para ejecutar una instrucción resta (SUB).

Actividad 2:

- **Actividad de aprendizaje**

Autoevaluación.

- **Procedimiento**

Realice la autoevaluación 5. Si tiene dudas, consulte de nuevo el texto guía, o el texto: • Stallings, W. (2007). *Organización y arquitectura de computadoras*, Madrid (España), PEARSON Prentice Hall. De igual forma, puede exponer sus dudas al docente.



Autoevaluación 5

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. ¿Qué tarea realiza la Unidad de Control?
 - a. Almacenamiento de datos.
 - b. Temporizador de señales.
 - c. Secuenciamiento de micro operaciones.
2. ¿En qué tipo de registros del procesador está agrupado el registro contador de programa (PC)?
 - a. Registros que ingresan en el ALU.
 - b. Registros de control y estado.
 - c. Registros visibles por el usuario.
3. Realice el siguiente ejercicio.

Determine el conjunto de micro-operaciones que realiza la unidad de control cuando tiene una instrucción:

MUL R, X (Multiplicar el contenido de la posición X al registro R1)

4. Repita el ejercicio 3 para la instrucción:

STOR X, R1 (guarda el contenido del registro R1 en la posición de memoria X)

5. ¿Cuál es el proceso que sigue un ciclo de instrucción?
- Ciclo de ejecución, ciclo de E/S, ciclo de codificación.
 - Ciclo de captación, ciclo de ejecución, ciclo de interrupción.
 - Ciclo de captación, ciclo de interrupción y ciclo de ejecución.
6. En el ciclo de captación se ejecuta una de las siguientes acciones:
- Se realiza la operación en la ALU.
 - Se comprueba la existencia de una interrupción.
 - Se decodifica la operación contenida en la instrucción.
7. Escriba las micro-operaciones que se realizan en el proceso de captación.

8. Enumere las clases de interrupciones más comunes que puede atender el procesador.

9. () El registro contador de programa continúa incrementándose en 1 incluso cuando se atiende una interrupción.
10. () Luego de ejecutar una instrucción de un programa el CPU siempre chequea si existe una interrupción.

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultados de aprendizaje 11 y 12

- Reconoce e interpretar la estructura y funcionamiento de las interconexiones del computador.
- Identifica cuál es la mejor técnica de entrada y salida para optimizar el rendimiento e incrementar las prestaciones de un computador

Contenidos, recursos y actividades recomendadas



Semana 6



Unidad 6. Módulos de entrada y salida

Debido a la extensa variedad de periféricos que se pueden conectar a una computadora se requiere de un controlador de entrada y salida separado de la CPU. Esto se debe a que dichos dispositivos

funcionan a diferentes velocidades, trabajan con datos de distintos formatos y cuentan con niveles de voltaje y corriente diferentes a los de la CPU. Todos son más lentos que la CPU y la RAM, conectarlos directamente implicaría que la CPU tenga que trabajar más lento y en una variedad de formatos de datos y voltaje insostenible para el diseñador.

Los módulos de entrada y salida (E/S), hacen de interfaz con la CPU y con la memoria RAM o con otros periféricos.

Un modelo genérico de E/S se puede observar en la figura 52.

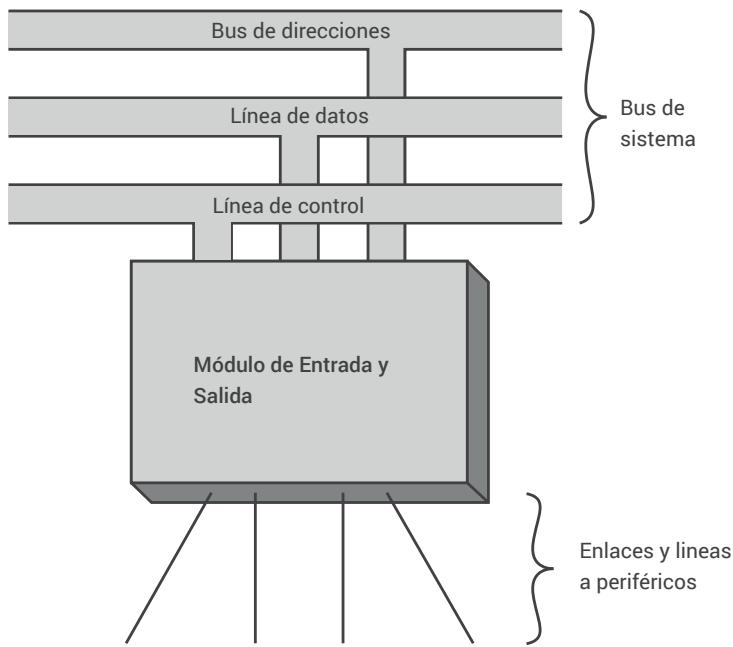


Figura 52. Modelo genérico de módulo de E/S.

Fuente: Stallings (2007).

6.1. Función del módulo de E/S

Las funciones del módulo de E/S son:

Control y temporización: coordina el flujo del tráfico de datos entre la CPU o memoria hacia el exterior.

Comunicación con la CPU: el módulo de E/S acepta comandos desde el procesador, estos comandos son enviados en forma de señales a través del *bus* de datos. También se intercambia información de estado del dispositivo externo, si está apagado, encendido, o listo para enviar o recibir datos. Al igual que con la memoria los módulos de E/S tienen una dirección única para que la CPU envíe o reciba datos.

Almacenamiento temporal: una tarea esencial del módulo de E/S es el almacenamiento temporal de datos (*buffering*). Esto se requiere debido a la diferencia de velocidades entre la CPU, memoria y los dispositivos externos. Los datos no se pueden transmitir a la par, es necesario almacenarlos temporalmente para luego transmitirlos a la velocidad apropiada ya sea a alta velocidad hacia la CPU o a menor velocidad hacia el periférico.

Detección de errores: otra tarea del módulo de E/S es la de detectar errores y reportarlos al procesador. Un tipo de errores comprende los problemas de *hardware* (mecánicos o eléctricos) y otro tipo los errores causados por errores en la transmisión de *bits* o cuellos de botella (*software*).

En general, todo dispositivo externo (teclado, ratón, impresora, etc.) contiene una lógica de control, un *buffer* y un transductor (mecanismos de imagen, partes mecánicas, etc.) En la figura 53, podemos apreciar el esquema general de estos dispositivos.

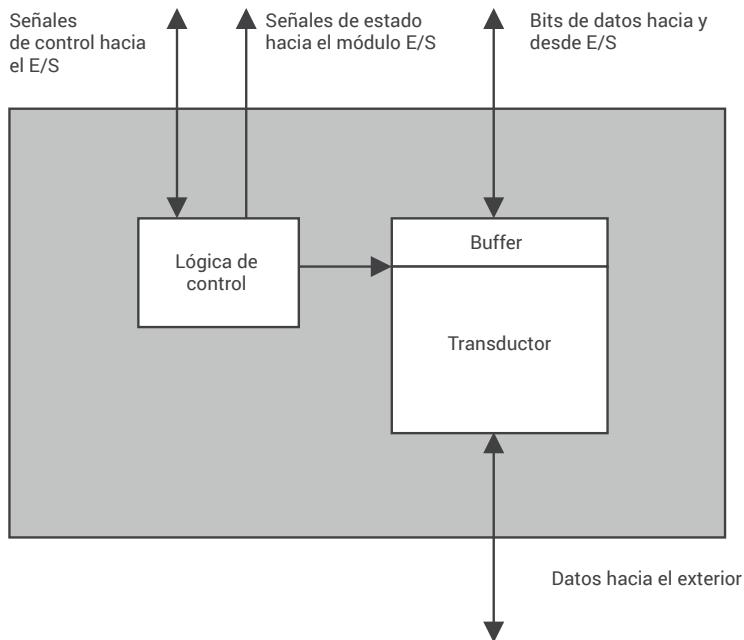


Figura 53. Diagrama de bloques de un dispositivo externo

Fuente: Stallings (2007).

Dentro de la computadora se encuentran los módulos de E/S, la figura 54 muestra los bloques principales que conforman un módulo estándar.

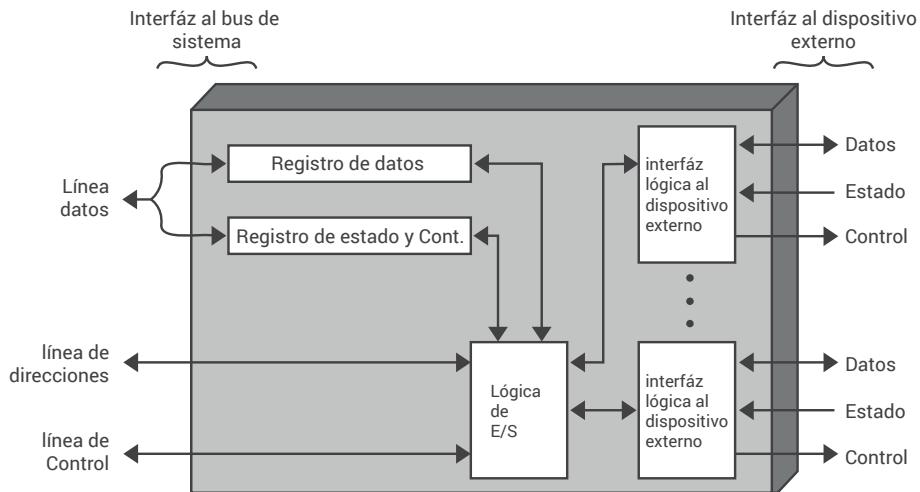


Figura 54. Módulo de E/S.

Fuente: Stallings (2007).

6.2. Técnicas de entrada y salida

Existen tres técnicas básicas que pueden ser implementadas en un sistema de E/S, se observan en la figura 55.

- E/S programada.
- E/S con interrupciones.
- Acceso directo a memoria (DMA).

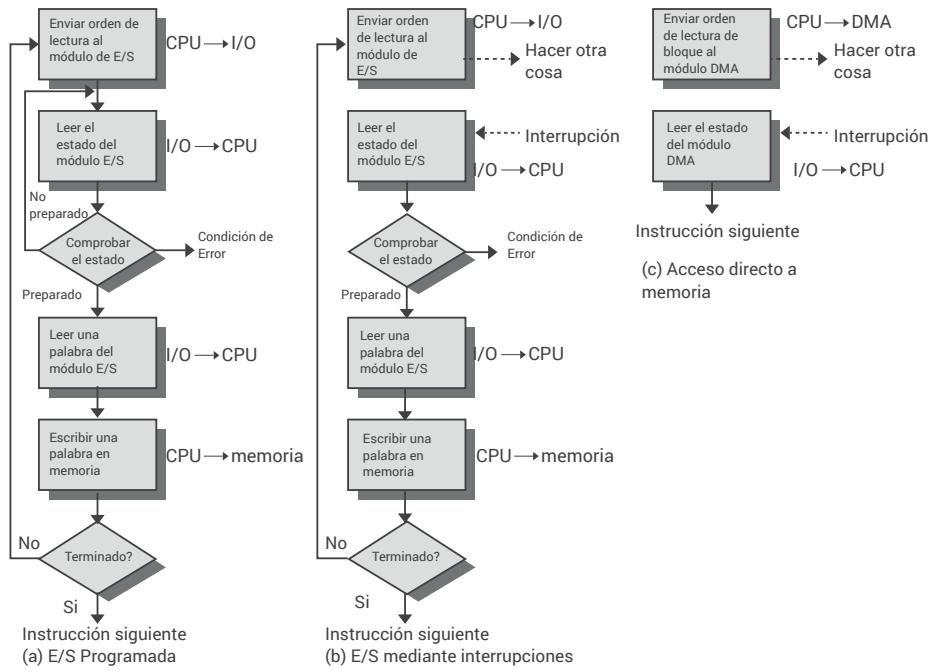


Figura 55. Tres técnicas de entrada de un bloque de datos.

Fuente: Stallings (2007).

6.2.1. Entrada y salida programada

En esta técnica los datos que se transfieren entre la CPU y E/S se realizan con la ayuda de una rutina de software. Cuando el procesador está ejecutando un programa y encuentra una instrucción de E/S, este la ejecuta generando las señales apropiadas para el módulo de E/S.

El módulo de E/S realizará las operaciones solicitadas con el periférico; sin embargo, este no emite alertas ni estados al procesador. Es tarea del procesador escanear periódicamente el estado del módulo de E/S hasta que este complete las tareas solicitadas.

En esta técnica el procesador una vez que emite un comando al módulo de E/S debe esperar hasta que la operación de E/S se complete para poder seguir trabajando. Esto disminuye el desempeño del procesador ya que el tiempo de la CPU es desperdiciado.

6.2.2. Entrada y salida mediante interrupciones

El problema con la E/S programada es que el procesador debe esperar un largo tiempo por el módulo de E/S. Además, debe emitir constantemente señales de consulta al módulo para checar su estado y comprobar si ha finalizado las tareas. Este tiempo de espera puede ser aprovechado para completar otras acciones. La solución a este inconveniente es implementar un sistema de interrupciones. Así, el procesador emite un comando al módulo de E/S y mientras este trabaja el procesador puede continuar realizando otras operaciones. El módulo de E/S interrumpe al procesador una vez que el periférico está listo para intercambiar datos o se ha completado la tarea, una vez que el procesador atiende el requerimiento este puede seguir trabajando en otras tareas hasta la próxima interrupción del módulo de E/S.

Desde el punto de vista del módulo de E/S el proceso se realiza de la siguiente forma.

- a. El procesador emite un comando READ.
- b. El módulo de E/S procede a leer los datos de un periférico.
- c. Una vez que los datos se encuentran en los registros del módulo, el módulo de E/S envía una interrupción al procesador por medio del bus de control.
- d. El módulo espera hasta que el procesador lo atienda.
- e. Cuando el procesador atiende la interrupción, el módulo pone los datos en bus de datos y está listo para otra operación de E/S.

Desde el punto de vista del procesador el proceso mediante interrupciones se realiza de la siguiente forma.

El procesador emite un comando READ.

El procesador continua realizando otras acciones (otros programas).

Al final de cada ciclo de instrucción el procesador comprueba si existen interrupciones.

Cuando encuentra una interrupción desde un módulo de E/S, el procesador almacena lo que estaba haciendo (guarda el contenido del PC y de los registros).

En este caso el procesador procede a leer los datos que envía el módulo de E/S y los guarda en memoria.

Luego reestablece el programa que estaba ejecutando antes de la interrupción (trae de memoria el contenido del PC y de los registros nuevamente).

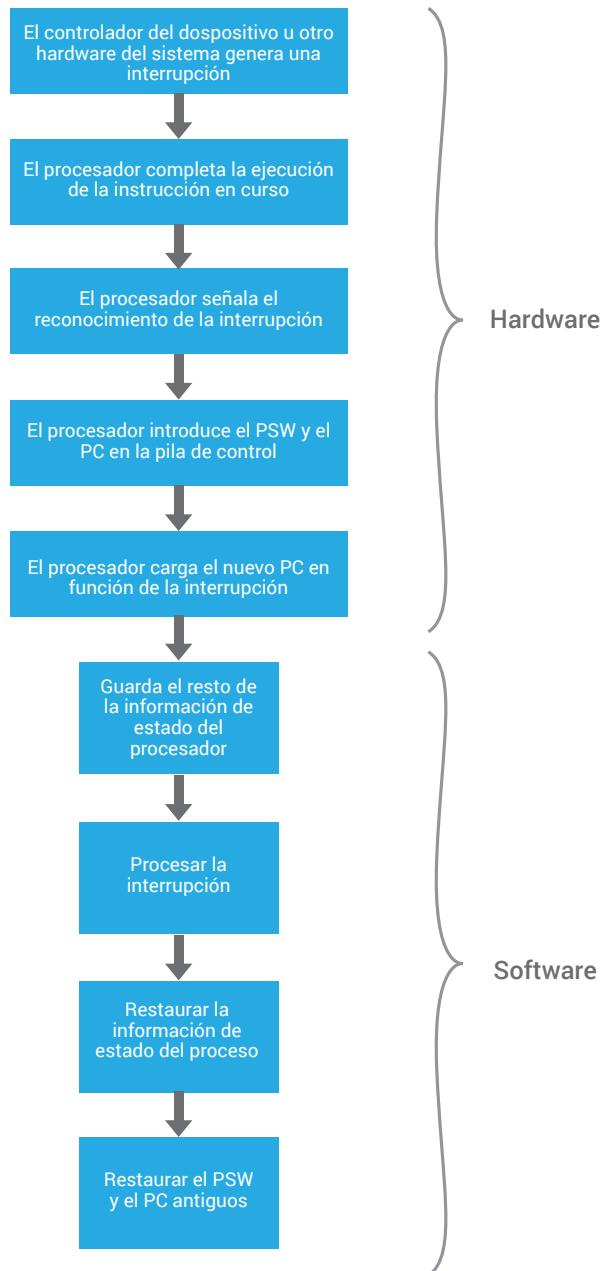


Figura 56. Procedimiento de interrupción simple.

Fuente: Stallings (2007).

En el momento en el que varios dispositivos están conectados al mismo tiempo puede surgir dos problemas: 1) ¿Cómo puede identificar el procesador de dónde provienen una interrupción? 2) Si ocurren múltiples interrupciones al mismo tiempo, ¿cómo decide el procesador a cuál atender primero?

Primero, para la identificación de dispositivos se tienen cuatro categorías

- **Múltiples líneas de interrupción**

Simplemente se provee múltiples líneas de interrupción en el procesador y los módulos de E/S. Es un tanto impráctico ya que se debe dedicar pines físicos del procesador a las líneas de interrupción. Además, se puede llegar al caso en que múltiples módulos de E/S estén conectados a cada línea de interrupción, así que se debe aplicar una de las siguientes técnicas.

- **Consulta mediante software**

Este método necesita que el procesador consulte cada uno de los módulos de E/S para conocer cuál emitió la interrupción. Este método consume mucho tiempo ya que por cada interrupción el procesador debe consultar todos los módulos de E/S. Se puede implementar de dos maneras: el procesador puede tener una línea de comando específica para esto (Test I/O), cuando se realiza la consulta pone la dirección específica de cada módulo E/S en el bus de direcciones. El módulo de E/S responderá cuando reconozca su propia dirección el bus. Alternativamente se puede implementar en cada módulo de E/S un registro de estado. El procesador deberá leer cada uno de los registros de estado da cada E/S para reconocer cual emitió la interrupción.

- **Encadenamiento (consulta hardware, vectorizado)**

Esta técnica permite conectar varios módulos de E/S a una sola línea de interrupción; sin embargo, todos están conectados en forma de cadena a una línea adicional llamada *Interrupt Acknowledge (INTA)*. Cuando el procesador detecta una interrupción, envía una señal INTA. La señal se propaga por la cadena, cuando llega al módulo que emitió la interrupción, este responde enviando una palabra en el *bus* de datos, esta palabra puede ser un vector indicando su dirección o identificación en la cadena. Esta técnica ahorra mucho tiempo al procesador al evitar consultar a todos los módulos uno a uno, puede observar esto en la figura 57.

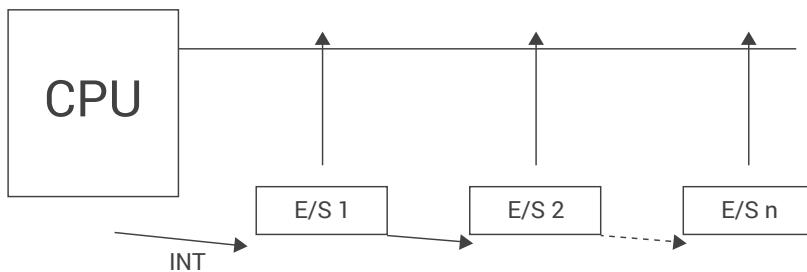


Figura 57. Bus encadenado.

Fuente: Elaboración propia.

- **Bus arbitrado (sectorizado)**

En el *bus arbitrado*, el módulo de E/S debe ganar el control del *bus* antes de emitir una interrupción. Así solo un módulo puede enviar interrupciones al CPU. El procesador responde con un ACK en la línea y el módulo de E/S pone su dirección o vector en la línea de datos.

En el caso de tener múltiples interrupciones a la vez existen algunos métodos según las técnicas arriba descritas.

- **Con múltiples líneas de interrupción**

Al procesador se le asigna una jerarquía para cada línea de interrupción, esto se hace durante el diseño. En caso de múltiples interrupciones el procesador atenderá a la línea con mayor prioridad.

- **Con consulta software**

En este caso el orden de consulta de cada módulo de E/S determina su prioridad.

- **Con configuración en cadena**

La prioridad es determinada por la posición de los módulos de E/S con respecto al procesador. El módulo con mayor prioridad es el que está más cercano al procesador, debido a que será el primero en recibir la señal de ACK desde el procesador.

- **Bus arbitrado**

En este caso se necesita un método de arbitraje antes de ceder el *bus* a un determinado módulo. Este método de arbitraje puede ser centralizado o distribuido. En el esquema centralizado un solo hardware llamado controlador de *bus* es el encargado de manejar los tiempos de acceso de cada módulo al *bus*. En un esquema distribuido no existe controlador central, en su lugar cada módulo de E/S contiene una lógica de acceso y los módulos actúan juntos para compartir el *bus*.

6.3. Acceso directo a memoria

Las dos técnicas de E/S que revisamos anteriormente (E/S programada y E/S mediante interrupciones) requieren una participación del procesador para transferir datos desde memoria a periféricos o viceversa. Cada *bit* debe atravesar el procesador,

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

formando un camino de datos desde los módulos de E/S y la memoria. Imagine los casos donde se debe guardar gran cantidad de información desde un periférico a memoria, la transferencia de estos datos estará limitada a los recursos del procesador y a su disponibilidad. El procesador estará muy ocupado gestionando esta transferencia de datos y no podrá ejecutar muchas instrucciones de la computadora o estas se ralentizarán.

Esta transferencia de datos podría ser gestionada por otro dispositivo, y permitir que el procesador se libere, este último solo gestionará el inicio y final de la transmisión de datos. Esta técnica se conoce como Acceso Directo a Memoria (*Direct Memory Access - DMA*).

El controlador DMA es un circuito de control asociado a los módulos de E/S. El DMA permite la transferencia de datos entre dispositivos externos y memoria principal sin intervención continua de procesador, el esquema de este módulo se muestra en la figura 58.

El controlador DMA toma el control del sistema desde el procesador y la transferencia de datos tiene lugar sobre el *bus* del sistema.

Para esto el DMA usa este *bus* solo cuando el procesador no lo está usando, o debe forzar al procesador a pausar su uso temporalmente. Esto se conoce como robo de ciclo, debido a que el DMA en efecto roba un ciclo de *bus* al procesador.

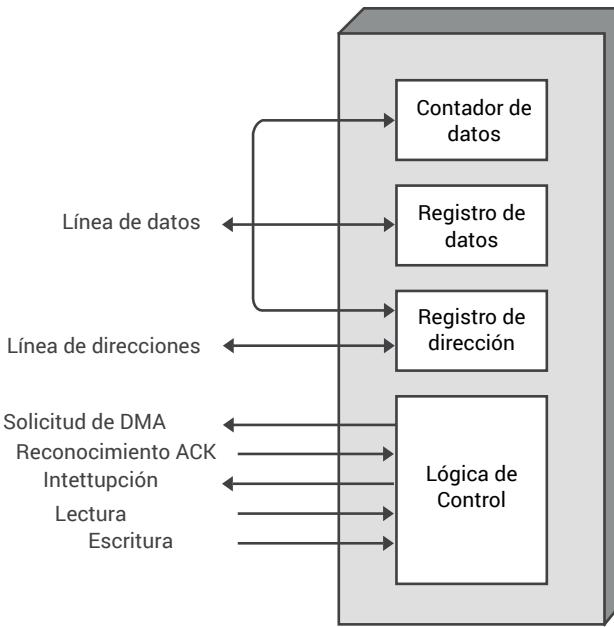


Figura 58. Diagrama de bloques de un módulo DMA.

Fuente: Stallings (2007).

Cuando el procesador desea leer o escribir un bloque de datos, envía un comando al módulo DMA con lo siguiente.

- Si la operación es de lectura o escritura: *read/write*,
- La dirección del módulo de E/S.
- La locación de inicio del bloque de memoria a leer o escribir, estas direcciones se envían a través de las líneas de datos y se almacenan en los registros de direcciones.
- La cantidad de datos que se deben transmitir.

El CPU mientras tanto puede continuar con otro trabajo, el DMA trabaja en la transmisión de datos, una vez que se ha terminado la transmisión el DMA envía una interrupción al procesador.

El mecanismo DMA puede ser configurado de diferentes maneras. Las más comunes son:

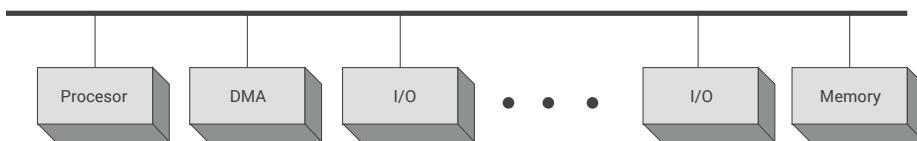


Figura 59. Bus único.

Fuente: Stallings (2007).

Bus único, DMA independiente. Cada transferencia utiliza el *bus* dos veces, de E/S a DMA y de DMA a memoria, figura 59.

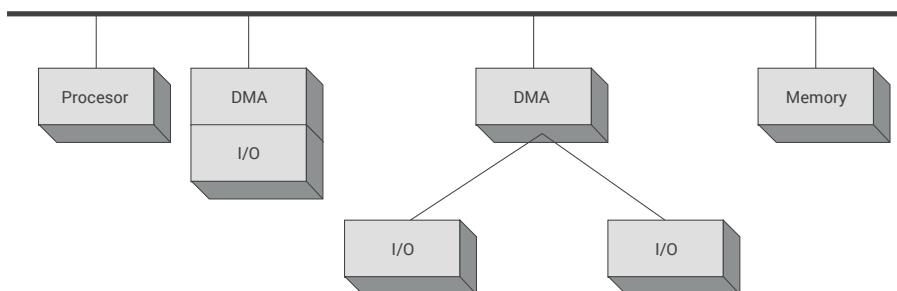


Figura 60. Integración de DMA en bus único.

Fuente: Stallings (2007).

Bus único, DMA-E/S integrados. El DMA puede soportar más de un dispositivo. Cada transferencia usa el *bus* una sola vez: de DMA a memoria (figura 60).

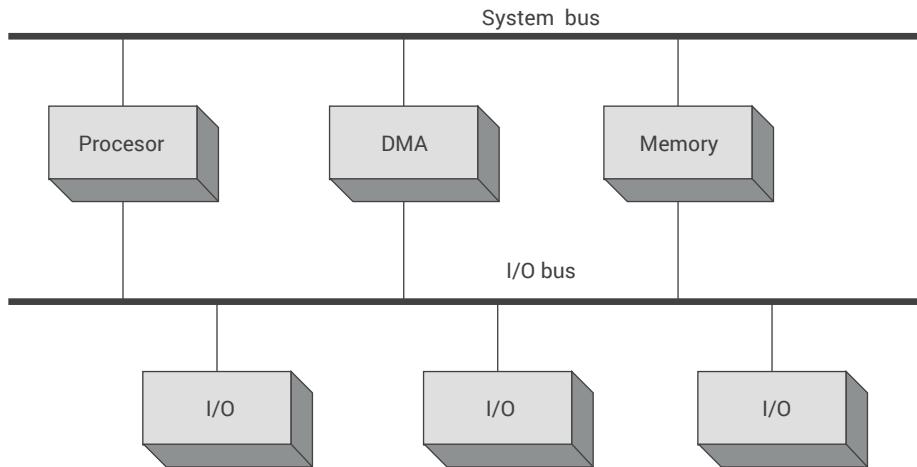


Figura 61. Bus de E/S y bus de sistema a través de DMA.

Fuente: Stallings (2007).

Bus de E/S. El *bus de E/S* soporta todos los módulos conectados al DMA. El *bus del sistema* es usado una vez en cada transmisión (figura 61).

6.4. Buses

El procesador, la memoria principal y los módulos de E/S pueden estar interconectados mediante líneas de comunicación de datos llamadas *buses*.

La principal función de los *buses* es ser un camino entre las estructuras funcionales de la computadora permitiéndoles transmitir datos entre ellas. Los *buses* incluyen líneas de control para soportar las interrupciones y métodos de arbitraje.

Los *buses* se pueden agrupar en tres categorías:

- *Bus de datos.*
- *Bus de direcciones.*
- *Bus de control.*

Los *buses* deben transmitir datos, operaciones de lectura READ o escritura WRITE R/W, además de comunicar la longitud de datos a transmitir, deben transmitir las señales de control para la sincronización de transmisión de datos entre procesador y memoria o E/S.

Los tipos de sincronización en los *buses* se pueden agrupar en dos.

Bus síncrono: en un *bus* síncrono todos los dispositivos se sincronizan con una señal de reloj común. Solo pueden transmitir durante un periodo de reloj. Los datos pueden ser movidos en un pulso de reloj (figura 62), un dispositivo maestro (procesador o DMA) pone una dirección y comandos en el *bus* de direcciones y control durante todo un periodo de reloj, al T1 la memoria o módulo de E/S responde poniendo datos en el *bus* de datos. El pulso en bajo del reloj (T2-T1) debe ser al menos de la misma longitud que el tiempo de propagación de los datos desde memoria o E/S al *bus* de datos (figura 62).

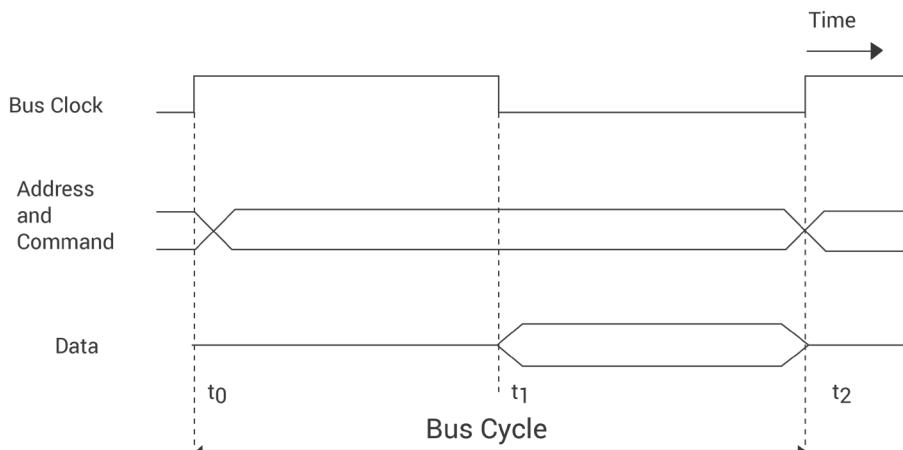


Figura 62. Temporización de transferencia de entrada en un *bus* síncrono.
Fuente: Stallings (2007).

Es posible mejorar la transferencia de datos síncrona al permitir que los datos puedan acomodarse durante varios ciclos de reloj, así, se puede aumentar la frecuencia del reloj del sistema e incrementar la velocidad, como se muestra en la figura 63.

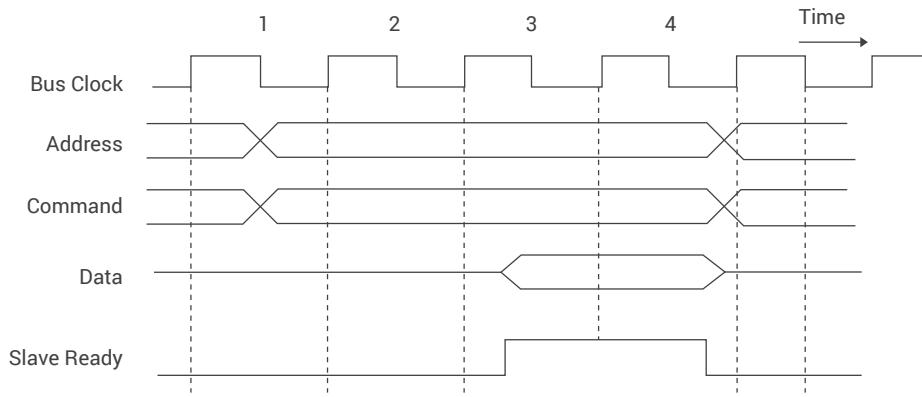


Figura 63. Transferencia de entrada usando múltiples ciclos de reloj.

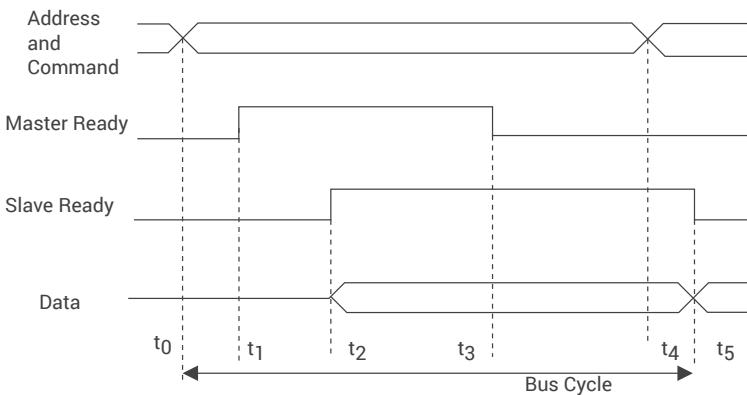
Fuente: Stallings (2007).

Bus asíncrono

En el modo de transferencia asíncrona, se usa una señal llamada *Handshake Signal* entre un dispositivo maestro y uno esclavo. En este tipo de buses no se utiliza una señal de reloj, esta señal es remplazada por dos señales de control: maestro-ready y esclavo-ready.

El dispositivo maestro pone una dirección y comandos en el *bus*. Entonces indica a todos los dispositivos esta acción activando una señal *master-ready*.

Esto provoca que todos los dispositivos conectados al *bus* decodifiquen la dirección, si un dispositivo reconoce su dirección en el *bus*, este activa la señal *esclavo-ready* y empieza la operación de escribir o leer datos del *bus* (figura 64).



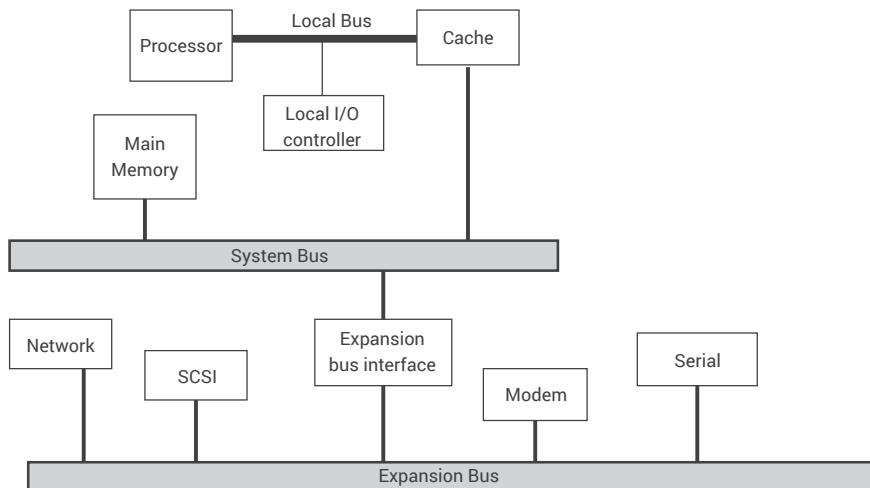
Handshake control of data transfer during an input operation

Figura 64. Control tipo “Handshake” de transferencia de datos durante una operación de entrada.

Fuente: Stallings (2007).

Arquitecturas de bus

- Arquitectura de *bus* tradicional, como se muestra en la figura 65, el *bus* de expansión se utiliza para conectar los dispositivos más lentos como la tarjeta de red, la comunicación serial, impresoras etcétera.

*Figura 65.* Bus tradicional.

Fuente: Stallings (2007).

- Arquitectura de altas prestaciones, esta agrega un *bus* llamado de “altas prestaciones” que posteriormente sería remplazado por el *bus*: *Peripheral Component Interconnect (PCI)*; el *bus* de altas prestaciones permite que dispositivos con alta velocidad de transferencia de datos se conecten directamente a la caché (figura 66).

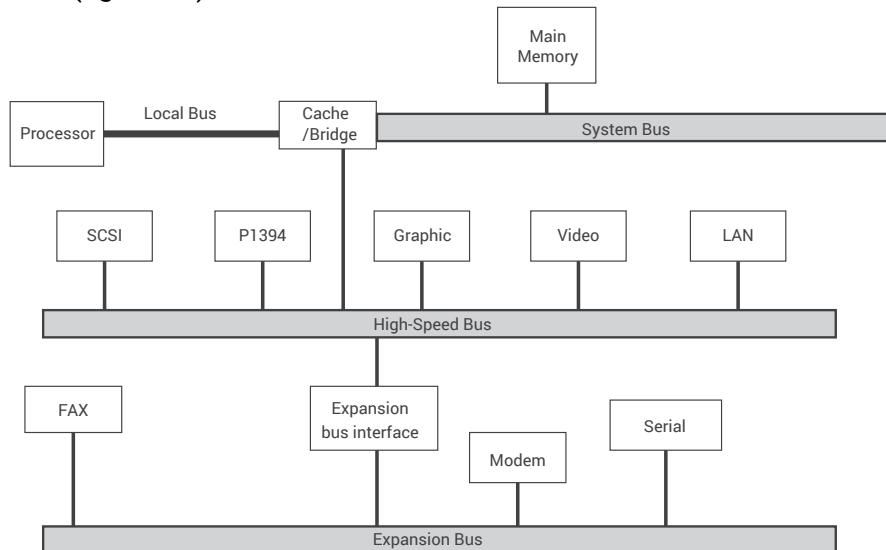


Figura 66. *Bus* de altas prestaciones.

Fuente: Stallings (2007).

6.5. El *bus* PCI

Uno de los *buses* más populares fue el *bus* ISA, desarrollado por IBM en 1982, tenía un ancho de 16 bits y operaba a 4.77 MHz, este *bus* era capaz de transmitir información a la velocidad de 9 MBps. Muchas generaciones de computadoras usaron este *bus* hasta hace muy poco debido a su compatibilidad con equipos antiguos, a la popularidad de IBM y a que muchos fabricantes usaban este *bus* como un estándar. Además, en esa época no había muchos periféricos que utilizaran esa velocidad, así que no se necesitaba que el *bus* acepte gran cantidad de dispositivos conectados a él.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Debido a la aparición de más periféricos que necesitaban transmitir gran cantidad de datos, aparecieron otros *buses* como el EISA (*Extended Industry Standard Architecture*) con mayor ancho de palabra (32 bits), y a una frecuencia de 8 MHz. También se empezó a utilizar el Bus-VL (Vesa Local Bus), este *bus* fue un estándar de VESA (*Video Electronics Standards Association*) tenía también 32 bits de ancho pero podía funcional a la misma velocidad del procesador. La desventaja de estos estándares de *buses* era que podían funcionar siempre y cuando se conecten a ellos uno o dos periféricos, especialmente tarjetas de video.

Durante la década de los 90 Intel desarrolló el *bus PCI* (*Peripherical Component Interconnect*). El nuevo *bus PCI* recogía las ventajas de los *buses ISA* y *VL-Bus*. Con la ventaja de que este *bus* tiene un método de acceso a memoria directo (DMA), lo que permite eliminar la interferencia con el procesador cuando se transmite gran cantidad de datos.

El *bus PCI* soporta anchos de palabra de hasta 64 bits, multiplexadas sobre líneas de 32 bits. El *bus* soporta también sistemas de multiprocesadores o monoprocesador, el esquema se observa en las siguientes figuras. En el *bus PCI* las direcciones y datos están multiplexadas sobre las mismas líneas, el esquema de arbitraje es de tipo centralizado, y la temporización que utiliza es síncrona.

En las figuras 67 y 68 se pueden observar los dos tipos de organización del *bus PCI* en una computadora, en el primer caso el *bus PCI* se conecta directamente al controlador de memoria, y el *bus* de expansión se convierte en esclavo del *bus PCI*. En el segundo caso, usado para servidores o sistemas con varios procesadores, el *bus PCI* se conecta al *bus* del sistema.

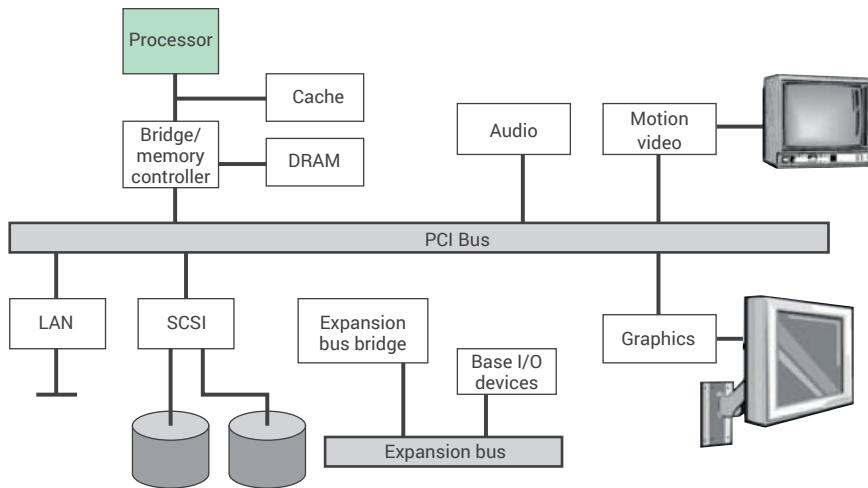


Figura 67. Organización del *bus PCI* típico.

Fuente: Stallings (2007).

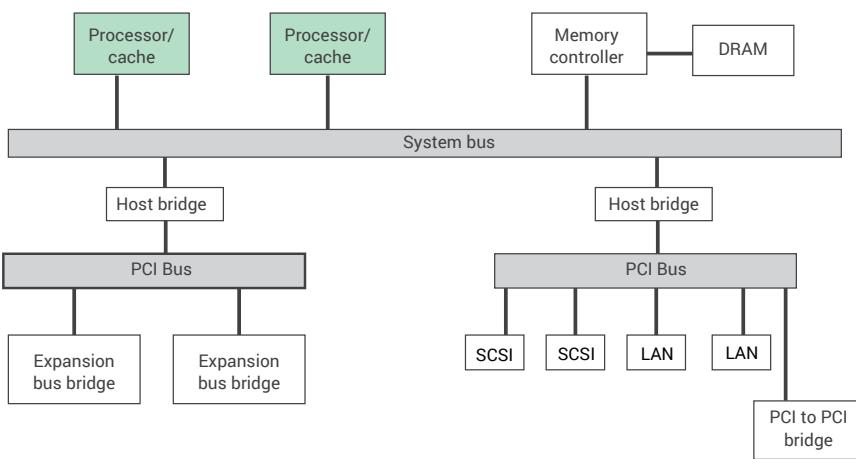


Figura 68. Esquema de organización del *bus PCI* para un servidor.

Fuente: Stallings (2007).

El *bus PCI* es el más aceptado por los sistemas computacionales actuales, el cual es mantenido y actualizado por la asociación PCI SIG (*Special Interest Group*).

¿Dónde está el bus PCI?

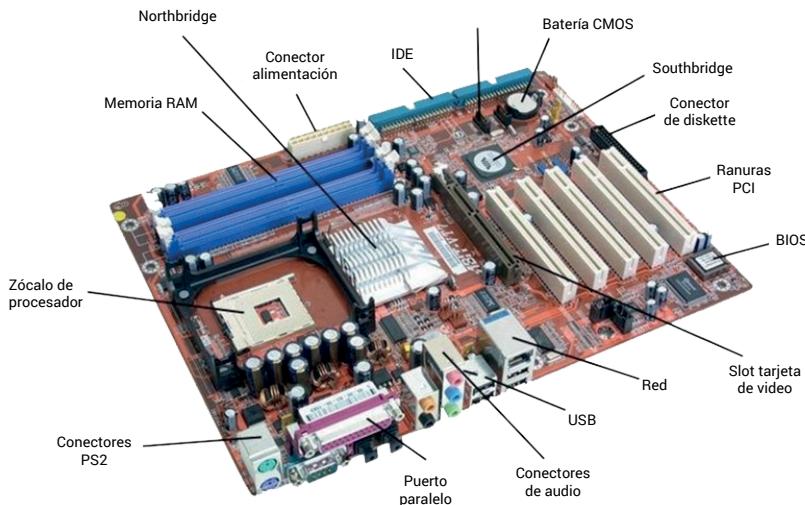
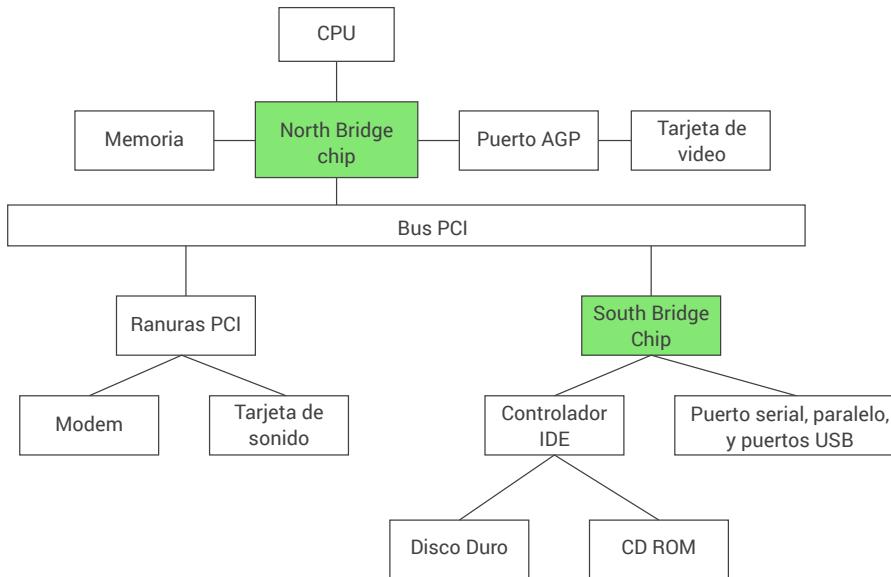


Figura 69. Tarjeta madre de un PC típico.

Fuente: [Partes de una placa base](#).

Esta estructura la veríamos en forma de esquema de la siguiente manera (figura 69).





Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Ejercicio DMA.

- **Procedimiento**

Revise la sección 6.3 del texto base, extraiga las diferencias que existen entre el DMA y las técnicas de acceso programada y mediante interrupciones. Con ayuda de un esquema o flujo gráfico indique las diferencias entre las tres técnicas.

Actividad 2:

- **Actividad de aprendizaje**

Autoevaluación.

- **Procedimiento**

Realice la autoevaluación de la unidad 6. Donde podrá comprobar sus conocimientos sobre las técnicas de acceso de E/S y la estructura de interconexión del computador.

Si tiene dudas, consulte de nuevo el texto guía, o el texto: • Stallings, W. (2007). *Organización y arquitectura de computadoras*, Madrid (España), PEARSON Prentice Hall. De igual forma, puede exponer sus dudas al docente.



Autoevaluación 6

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. La función principal del módulo de E/S es:
 - a. Actuar de interfaz entre el computador y dispositivos externos.
 - b. Actuar como interfaz entre memoria y procesador.
 - c. Reemplazar al bus del sistema.
2. El mecanismo de DMA puede configurarse de diversas formas una de ellas es la siguiente:
 - a. Bus único, DMA independiente.
 - b. Bus de altas prestaciones.
 - c. Canales de E/S.
3. Un módulo DMA está constituido por.
 - a. Sólo hardware.
 - b. Un conjunto de hardware y software.
 - c. Sólo software.
4. El Bus de datos se utiliza para interconectar las siguientes estructuras.
 - a. CPU, Unidad de Control, ALU.
 - b. CPU, Memoria, E/S.
 - c. CPU, Memoria, Unidad de Control.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. La anchura del Bus determina las prestaciones del sistema, si un procesador es capaz de ejecutar instrucciones de 16 bits de longitud, ¿cuál sería la anchura mínima del bus para no reducir el rendimiento de todo el sistema?
 - a. Debe tener mínimo 8 líneas.
 - b. Debe tener mínimo 32 líneas.
 - c. Debe tener mínimo 16 líneas.
6. Elija el literal correcto que enumere los tipos de buses.
 - a. Dedicado y multiplexado.
 - b. Centralizado y Distribuido.
 - c. Lectura y Escritura.
7. Los métodos de arbitraje en un bus son:
 - a. Dedicado y multiplexado.
 - b. Centralizado y Distribuido.
 - c. Lectura y Escritura.
8. Existen dos tipos de temporización, los cuales son:
 - a. Dedicado y multiplexado.
 - b. Centralizado y Distribuido.
 - c. Síncrono y asíncrono.

Responda verdadero o falso según corresponda.

9. () En el bus del sistema generalmente se conecta periféricos como: módem, Red, FireWire.
10. () En un equipo de sobremesa típico, el bus PCI puede reemplazar al bus del sistema.

[Ir al solucionario](#)



Actividades finales del bimestre

Resultados de aprendizaje 13 y 14

- Describe las principales características de la arquitectura de computadores.
- Analiza, evalúa y recomienda el uso de arquitecturas de computadoras para un objetivo determinado.

Contenidos, recursos y actividades recomendadas



Semana 7

En esta semana recordaremos lo estudiado en las seis unidades del texto-guía como preparación para la evaluación presencial de la semana 8.

Estudie las unidades 1-6 del texto-base, extraiga las principales características estructurales y funcionales de una computadora.

LECTURA

Castillo T. y Cueva S. (2018). *Arquitectura de computadoras y sistemas operativos*. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En los últimos años se ha visto un resurgimiento de la popularidad de la arquitectura ARM, lea el siguiente artículo y extraiga las diferencias de esta arquitectura frente a las x86 de Intel.

LECTURA

Pastor, J. (2017). *RISC-V frente a ARM y x86: el amanecer de los procesadores personalizados es Open Source*. Recuperado de <https://www.xataka.com/componentes/risc-v-frente-a-arm-y-x86-el-amanecer-de-los-procesadores-personalizados-es-open-source>

Luego de estudiar el primer bimestre y de analizar el artículo anterior, le invitamos a realizar las siguientes actividades de aprendizaje y evaluación.



Actividades de aprendizaje recomendadas

- **Actividad de aprendizaje**
Autoevaluación.
- **Procedimiento**
Estudie de nuevo las preguntas y la retroalimentación de las 6 primeras autoevaluaciones del texto guía como herramienta de estudio para la evaluación presencial

Esta actividad le ayudará como preparación a la evaluación presencial que rendirá.



Semana 8

Evaluación Presencial escrita.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Segundo bimestre

Resultados de aprendizaje 15 y 16

- Describe las funciones y componentes necesarios de un sistema operativo.
- Analiza evalúa y recomienda el uso de los sistemas operativos para una tarea u objetivo determinado.

Contenidos, recursos y actividades recomendadas



Semana 9



Unidad 7. Principios de los sistemas operativos

En esta unidad, se revisan los fundamentos de sistemas operativos, que servirán de base para el estudio del segundo bimestre de esta asignatura.



Cabe recalcar que los sistemas operativos están evolucionando continuamente, dependiendo del progreso del *hardware*, por lo cual se le invita a que revise las tendencias de los mismos.

7.1. Sistema operativo

Empezaremos revisando la definición de qué es un sistema operativo de algunos autores, debido a que no hay una definición universal. Un sistema operativo:

- Es considerado como un “gerente ejecutivo”, es la parte del sistema de cómputo que administra el *hardware* y el *software* (Flynn y McHoes, 2001).
- Es el *software* que se ejecuta en modo kernel, se encarga de cumplir con las funciones de: a) proporcionar a los programadores de aplicaciones un conjunto abstracto de recursos simples y b) administrar estos recursos de *hardware* (Tanenbaum, 2009).
- Un sistema operativo es un programa que gestiona el *hardware* de una computadora; además proporciona una base para los programas de aplicación y actúa como intermediario entre el usuario y el *hardware* de la computadora (Silberschatz, Galvin y Gagne, 2013).
- Es un programa, o más bien un conjunto de programas o algoritmos, que actúa como intermediario entre el usuario (en su sentido amplio) de una computadora y el *hardware* de una computadora (Silva, 2015).

Con estos conceptos los invitamos a que realicen una definición propia de un sistema operativo.

Como se ha visto en todas las definiciones anteriores, se puede concluir que el sistema operativo es el encargado de administrar el *hardware* y *software* del sistema informático.

Como usuario usted no utiliza directamente el sistema operativo, sino que se interactúa con aplicaciones, que le brindan una interfaz amigable; en la figura 70, se puede observar la ubicación del sistema operativo.



Figura 70. Ubicación del sistema operativo. Fuente: Silva (2015)

Como se puede observar en la figura 70, se encuentra el *hardware* conformado por todos los componentes físicos; encima se encuentra el *software* o las aplicaciones que sirven de interfaz a los usuarios.

Según Tanenbaum (2009), el *software* se puede ejecutar en dos modos:

- **Modo kernel (modo supervisor):** el sistema operativo, tiene acceso completo a todo el *hardware* y puede ejecutar cualquier instrucción que la máquina sea capaz de ejecutar.

- **Modo usuario:** se ejecuta el resto del *software*, en el cual solo un subconjunto de las instrucciones de máquina es permitido.

Según Stallings (2005), se puede considerar que un sistema operativo tiene tres objetivos.

- **Facilidad de Uso:** facilita el uso de una computadora.
- **Eficiencia:** permite que los recursos de un sistema de computación se puedan utilizar de una manera eficiente.
- **Capacidad para evolucionar:** se debe construir de tal forma que se puedan desarrollar, probar e introducir nuevas funciones en el sistema sin interferir con su servicio.

Como se había mencionado anteriormente, el *hardware* ha evolucionado y por ende los sistemas operativos han tenido que acoplarse a estos nuevos requerimientos, de tal forma que tengan la capacidad de ser portables; en la siguiente sección se revisan las generaciones de sistemas operativos.

7.2. Generaciones de los sistemas operativos

En esta sección se ofrece una panorámica del proceso de la evolución de los sistemas operativos por generaciones.

7.2.1. Primera generación

Esta generación abarca los años 1940-1955, en la época de los tubos al vacío (bulbos) y de computadoras que tenían el tamaño de los salones de clase.

Según Silva (2015), las primeras computadoras no tenían sistema operativo y era el usuario quien las operaba cumpliendo los roles de usuario, operador, programador y administrador; por lo cual debía

conocer exactamente el *hardware* ya que a cada programa había que darle todas las especificaciones físicas, mediante clavijas y paneles, para que se ejecutara correctamente.

Los programadores operaban las máquinas desde consola principal, era un proceso en el que ellos debían hacer todo: para eliminar los errores de un programa el programador detenía al procesador, leía el contenido de cada registro, efectuaba las correcciones en las localizaciones de memoria y reanudaba la operación (Flynn y McHoes, 2001).

A estas computadoras se las conoce con el nombre de Computadoras controladas por programa, a continuación, en la figura 71, se muestra la arquitectura Von Neumann como un ejemplo de computadoras de la primera generación.

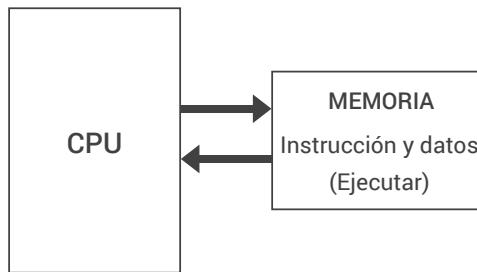


Figura 71. Arquitectura de Von Neumann

Fuente: elaboración propia.

Los computadores de esta generación, por lo general, se utilizaban para cálculos numéricos simples, como la obtención de logaritmos, senos y cosenos.

7.2.2. Segunda generación

Esta generación involucra el periodo desde 1955 hasta 1965, se conoce como la generación de los transistores y de sistemas de procesamiento por lotes.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Con la introducción de los transistores a mediados de 1950, cambió el panorama de las computadoras, debido a que se volvieron más confiables; además había una separación clara entre los roles de diseñadores, constructores, operadores, programadores y el personal de mantenimiento.

Tanenbaum (2009), menciona que estas máquinas se las conoce como mainframes, estaban encerradas en cuartos especiales con aire acondicionado y grupos de operadores profesionales para manejarlas; sin embargo, eran solo accesibles para empresas grandes, universidades o instituciones gubernamentales debido a su costo millonario.

El trabajo se escribía por parte del programador en papel, luego se pasaban a las tarjetas perforadas; a continuación, el conjunto de tarjetas perforadas al cuarto de entrada de datos y lo entregaba a uno de los operadores y se debían esperar a que los resultados estén listos. A continuación, los operadores retiraban las hojas de los resultados y las llevaban al cuarto de salida de datos para que el programador pueda retirarlas.

El tiempo empleado en este proceso era mucho, por lo cual se buscaban alternativas para reducir el tiempo desperdiciado, para esto se adoptó como una solución el sistema de procesamiento por lotes; en la figura 72 se representa un sistema de procesamiento por lotes.

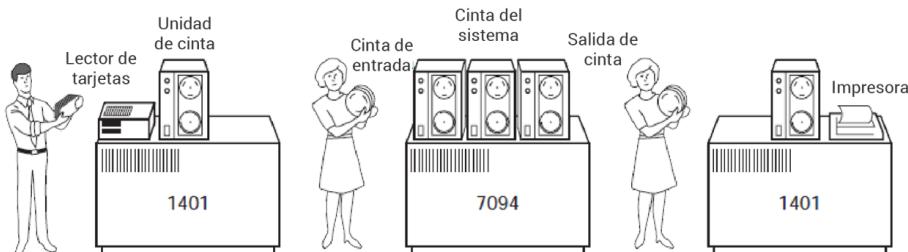


Figura 72. Sistema de procesamiento por lotes.

Fuente: Tanenbaum (2009)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Las computadoras de procesamiento por lotes consistían en recolectar en el cuarto de entrada de datos la bandeja llena de trabajos, luego pasarlo a una unidad de cinta magnética, mediante el uso de una pequeña computadora, posteriormente se pasaba a otra computadora especial para llevar los cálculos numéricos; finalmente el operador llevaba a la computadora de salida y se imprimían los resultados.

Le invitamos a que revisen las características de las computadoras IBM 1401 en *Computer History Museum* (2015) y de IBM 7094 en *Computer History Museum* (2014), para que pueda tener como referencia las características técnicas de las primeras computadoras de procesamiento por lotes.

7.2.3. Tercera generación

Esta generación abarca desde 1965 a 1980, se caracteriza por la aparición de los circuitos integrados y la multiprogramación.

Las computadoras en esta generación fueron creadas con CPU más rápidos; sin embargo, tenían problemas de velocidad al interactuar con dispositivos de entrada/salida que eran lentos. Este problema se solucionó con la multiprogramación al cargar muchos programas en una sola vez y compartir el uso de la CPU.

Flynn y McHoes (2001), mencionan que los primeros sistemas operativos permitían que se diera servicio a cada programa por turno, siendo el mecanismo más común para implementar la multiprogramación el uso de las interrupciones, que es cuando se notifica al CPU de los sucesos que necesitan los servicios de los sistemas operativos.

Tanenbaum (2009), recalcan que la solución para evitar que la CPU esté mucho tiempo inactiva era particionar la memoria en varias piezas, con un trabajo distinto en cada partición, con lo cual

mientras se esperaba que se completara una operación de E/S otra podía estar usando la CPU. En la figura 73, se muestra un sistema multiprogramado con tres trabajos en memoria.

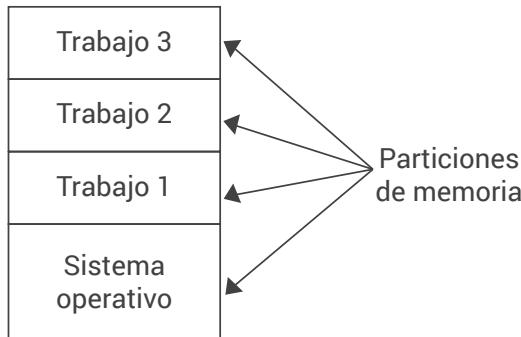


Figura 73. Sistema de multiprogramación.

Fuente: Tanenbaum (2009)

Otra de las características de los sistemas de esta generación es el uso de la técnica de spooling (*Simultaneous Peripheral Operation On Line*) y de tiempo compartido.

Entre los sistemas operativos de esta generación se encuentran a: OS/360, TSS/360, CTSS, Unics, UNIX, VAX-11/VMS.

7.2.4. Cuarta generación

Esta generación abarca desde 1980 hasta la actualidad, se la conoce como la generación de las computadoras personales.

Según Tanenbaum y Andrew (2009), en términos de arquitectura, las computadoras personales no eran del todo distintas de las minicomputadoras, pero en términos de precio sin duda eran distintas.

Además, surgieron los sistemas operativos en red y sistemas operativos distribuidos.

Entre los sistemas operativos de esta generación se pueden mencionar a DOS (Disk Operating System), MS-DOS (Microsoft Disk Operating System), OS/2, Lisa OS, MacOS, Minix, Windows 3.0, Windows 98, Windows NT, Windows Me, Windows XP, FreeBSD, etc.

Le invitamos a ver la película *Piratas de Silicon Valley* (2011), para que analice los cambios que se dieron en la última generación y la evolución de los distintos sistemas operativos.

Stallings (2005), considera que un sistema operativo debe evolucionar en el tiempo por las siguientes razones:

- **Actualizaciones de hardware y la aparición de nuevos tipos de hardware:** a medida que el hardware evoluciona, los sistemas operativos se deben acoplar a estos cambios.
- **Nuevos servicios:** debe ofrecer nuevos servicios como respuesta a la demanda del usuario o de las necesidades de los gestores del sistema.
- **Resolución de fallos:** cualquier sistema operativo tiene fallos, los cuales deben resolverse en el transcurso del tiempo y resolverse; claro que esto también implica que se introduzcan nuevos fallos.

7.3. Estructura de los sistemas operativos

En esta sección se revisará la organización interna de los sistemas operativos su clasificación, de los cuales se mencionarán las principales características.

Para gestionar la complejidad de los sistemas operativos se ha hecho hincapié en la estructura del sistema operativo, el cual debe ser modular. A continuación, se mencionan las principales estructuras.

7.3.1. Sistemas monolíticos

En este diseño que se considera como la organización más común, todo el sistema operativo se ejecuta como un solo programa en modo *kernel*. El sistema operativo se escribe como una colección de procedimientos, enlazados entre sí en un solo programa binario ejecutable extenso (Tanenbaum y Andrew, 2009).

En los sistemas monolíticos no existe una estructura propiamente dicha o es mínima. Según Silberschatz, Galvin, y Gagne (2013), existen numerosos sistemas comerciales que no cuentan con una estructura bien definida, dichos sistemas iniciaron como sistemas sencillos, pequeños y limitados que luego crecieron más allá de su alcance original.

De acuerdo con Tanenbaum y Andrew (2009), los sistemas operativos monolíticos, tienen una estructura básica compuesta por:

- Un programa principal que invoca al procedimiento del servicio solicitado.
- Un conjunto de procedimientos de servicio que lleva a cabo las llamadas al sistema.

Un conjunto de procedimientos utilitarios que ayudan a los procedimientos de servicio.

En la figura 74, se representa una estructura monolítica.

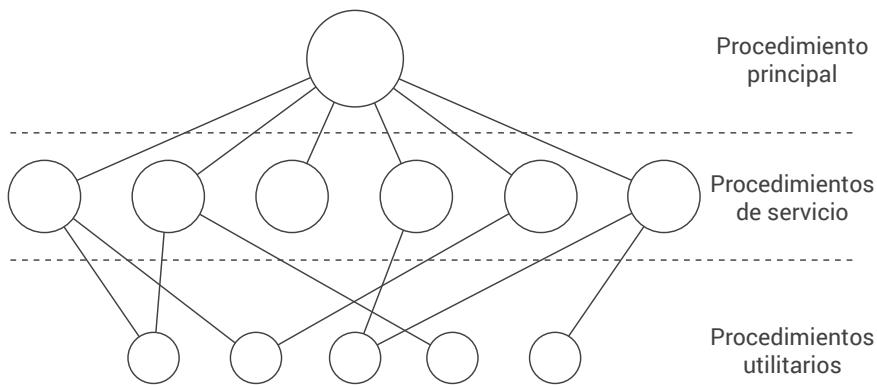


Figura 74. Modelo de estructura de un sistema monolítico.

Fuente: Tanenbaum y Andrew (2009)

Entre los sistemas operativos que tiene una estructura simple, se puede mencionar a: MS-DOS, UNIX original.

7.3.2. Sistemas de capas

El sistema operativo se descompone en varios niveles, cada uno de ellos está construido sobre capas inferiores; en donde la capa inferior (capa 0) es el *hardware* y la capa más alta (capa N) es la interfaz del usuario (Silberschatz, Galvin, y Gagne, 2013).

Las capas se seleccionan de forma que cada una utilice funciones y servicios exclusivamente de capas de niveles inferiores; de tal forma que se simplifica la depuración y la verificación del sistema. En la figura 75, se puede observar la estructura de un sistema operativo por capas.

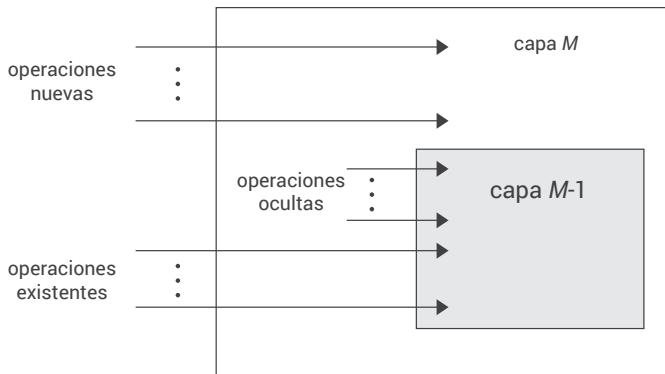


Figura 75. Sistema por capas.

Fuente: Stallings (2005)

Cada capa se implementa solo con aquellas operaciones proporcionadas por capas de nivel inferior; por lo tanto, una capa no necesita saber cómo están implementadas estas operaciones; solo requiere conocer qué hacen dichas operaciones.

De lo antes mencionado se concluye que, la principal ventaja de estos sistemas es la modularidad.

Como ejemplos de estos sistemas se pueden mencionar a OS/2, Windows NT 4.0.

7.3.3. Máquinas virtuales

Según Silberschatz, Galvin, y Gagne (2006), la idea fundamental que subyace a una máquina virtual es la de abstraer el *hardware* de la computadora, formando varios entornos de ejecución, creando la ilusión de que cada entorno de ejecución está operando en su propia computadora privada.

Las máquinas virtuales separan totalmente las funciones de multiprogramación y de máquina extendida, siendo el principal

elemento el monitor de máquina virtual que cumple con las funciones de:

- a. Ejecutarse en el hardware.
- b. Realizar la multiprogramación.
- c. Proporcionar varias máquinas virtuales a la capa superior.

Las máquinas virtuales instrumentan copias exactas del *hardware* simple, con su modo núcleo, usuario, interrupciones y todo lo demás que posee una máquina real. Pueden ejecutar cualquier sistema operativo que se ejecute en forma directa sobre el *hardware* (La Red, 2001).

Tanenbaum y Andrew (2009), mencionan como ejemplo de estos sistemas operativos a VM/370, el cual estaba basado en un sistema de tiempo compartido proporcionando: 1) multiprogramación y 2) una máquina virtual extendida con un interfaz más conveniente que el *hardware* por sí solo, la esencia VM/370 es separar por completo estas dos funciones. Es decir cada máquina virtual es idéntica al verdadero *hardware*, cada una puede ejecutar cualquier sistema operativo que se ejecute directamente solo en el *hardware*.

Una de las principales razones por las cuales se crea una máquina virtual es la compartición del *hardware* y a pesar de ello, operar con entornos de ejecución diferentes de forma concurrente.

7.3.4. Microkernel

Este método estructura el sistema operativo eliminando todos los componentes no esenciales del *kernel* e implementándolos como programas del sistema y de nivel de usuario, el resultado es un *kernel* más pequeño (Silberschatz, Galvin, y Gagne, 2006).

La principal función del *microkernel* es proporcionar un mecanismo de comunicaciones entre el programa cliente y los distintos servicios que se ejecutan en el espacio de usuario.

La idea básica detrás del diseño de *microkernel* es lograr una alta confiabilidad al dividir el sistema operativo en módulos pequeños y bien definidos, solo uno de los cuales (*microkernel*) se ejecuta en modo *kernel* y el resto se ejecuta como procesos de usuario ordinarios (Castellanos, 2014).

Entre los sistemas operativos que utilizan el método de *microkernel* se puede mencionar a: QNX, Tru64 Unix, Windows NT, PikeOS.

7.3.5. Modo cliente-servidor

Por lo general, la comunicación entre los clientes y servidores se la realiza utilizando el paso de mensajes.

Según Castellanos (2014), la esencia es la presencia de procesos cliente y procesos servidor. Un proceso cliente construye un mensaje indicando lo que desea y lo envía al servicio apropiado; después el servicio hace el trabajo y envía de vuelta la respuesta.

Este modelo puede ser utilizado tanto en un solo equipo como en una red de equipos.

En la figura 76, se representa un modelo cliente-servidor sobre una red de computadoras.

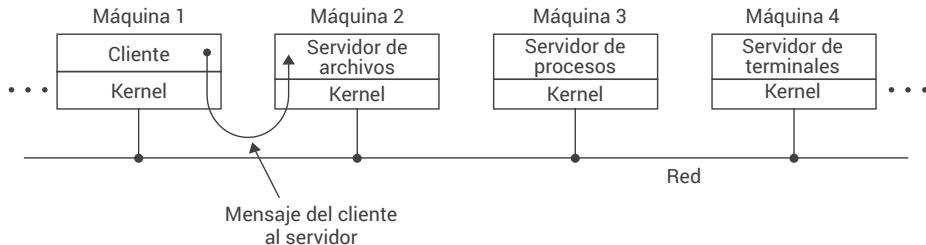


Figura 76. Modelo cliente-servidor sobre una red.

Fuente: Tanenbaum y Andrew (2009)

Como se puede observar en la figura anterior, la Máquina 1 envía un mensaje al servidor de archivos (Máquina 2), del cual esperará para

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

que le envíe un mensaje de respuesta cuando el servicio solicitado se haya completado.

7.4. Tendencias de los sistemas operativos

Al finalizar esta unidad, se han analizado las generaciones y las estructuras de los sistemas operativos; concluyendo que los mismos van cambiando de acuerdo con la demanda de nuevos requerimientos por parte de los usuarios y de la actualización del *hardware*.

Los sistemas operativos han experimentado cambios vertiginosos, los cuales han sido liderados por las grandes empresas como Windows, Apple, Google, IBM, etc., es así que los sistemas operativos actuales han sido diseñados para integrarse con el *hardware*. No solo se han diseñado sistemas operativos para computadoras sino para dispositivos móviles, dispositivos inteligentes, etcétera.

Tanenbaum y Andrew (2009), sugieren que para empezar el diseño de un sistema operativo, hay que determinar lo que debe hacer. La interfaz debe ser simple, completa y eficiente; el sistema debe estar bien estructurado y utilizar una de varias técnicas conocidas (estructura del SO). Además, no hay que olvidarse que el rendimiento es importante por lo cual las optimizaciones se deben seleccionar apropiadamente para no arruinar la estructura del sistema.

La empresa desarrolladora del sistema operativo Ubuntu, Canonical (2016) menciona que el *software* cada vez es más complejo, los desarrollos de *software* que antes eran sencillos y distribuidos en muchas máquinas, sistemas operativos, regiones y entornos; es por esto que las organizaciones están demandando una nueva generación de *software* para operar a una gran escala, con un alto grado de dependencia en integración y automatización para

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

desplegar nuevos servicios en cualquier lugar y en cualquier instante.

En Barrera (2016), se realiza un análisis de las tendencias de los sistemas operativos: de escritorio, basados en la web y para el internet de las cosas. En la tabla 7, se resumen las características de estos sistemas operativos.

Tabla 7. *Tendencias de los sistemas operativos*

Tipo Sistema Operativo	Sistema Operativo	Principales Características
Escritorio	Microsoft Windows 10	<ul style="list-style-type: none">▪ Asistentes personales de voz (Cortana).▪ Autenticación biométrica (Windows Hello).
	Ubuntu 16.04	<ul style="list-style-type: none">▪ Distribución con licencia libre.▪ Búsquedas locales y en línea (Unity).
	macOS Sierra	<ul style="list-style-type: none">▪ Sistema de archivos APFS, en el que se ha incorporado técnicas de encriptación, sellos de granularidad.▪ Handoff, se puede iniciar una tarea en un dispositivo y culminarla en otro.▪ Asistente personal de voz.
Basados en la Web	Chrome OS	<ul style="list-style-type: none">▪ Utiliza el núcleo de Linux.▪ Soporta aplicaciones escritas para Android.
Internet de las cosas	Ubuntu CORE	<ul style="list-style-type: none">▪ Utiliza paquetes de aplicaciones conocidos como snaps.▪ Utiliza el núcleo de Ubuntu.▪ Soporta una gran variedad de SoC.
	Windows 10 IoT	<ul style="list-style-type: none">▪ Plataforma de convergencia desde dispositivos de seguridad industrial hasta dispositivos conectados a la nube, conectividad nativa para comunicación máquina a máquina y de máquina a la nube.▪ Orientada a dispositivos pequeños embebidos que puedan o no tener pantallas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En los próximos años los sistemas operativos tendrán que cambiar para seguir nuevas tendencias y cumplir con nuevos retos; estos pueden ser sistemas basados en hipervisor, sistemas multinúcleo, espacios de direcciones de 64 bits, conectividad masiva, multiprocesadores, multicomputadoras de gran escala, multimedia, computadoras inalámbricas de bolsillo, los sistemas embebidos y nodos de monitoreo (Tanenbaum y Andrew, 2009).

Estimado estudiante:

Hemos finalizado la unidad 7, por ello se recomienda el desarrollo de las siguientes actividades.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Analice y compare la evolución de los sistemas operativos descritos. Diseño y evolución.

- **Procedimiento**

- Revisar artículos sobre tendencias de los sistemas operativos.
- Realizar un mapa conceptual de las tendencias de los sistemas operativos.

Estrategias de desarrollo:

Puede utilizar como base los siguientes recursos:

Barrera, K. (2016). *Tendencias actuales en los sistemas operativos de escritorio, basados en la web y para el internet of things (Iot)*. Recuperado de [Tendencias en los Sistemas Operativos Actuales](#)

Canonical. (2016). *Big software has arrived*. Recuperado de [It's time for IT to let software do more of the work.](#)

Piratas de Silicon Valley. (2011). [Película]. Recuperado de [Los Piratas De Silicon Valley Español Parte1](#)

Silberschatz, A., Galvin, A., y Gagne, G. (2006). *Fundamentos de sistemas operativos*. Madrid, España: McGraw-Hill Interamericana.

Tanenbaum, A. (2009). *Sistemas operativos modernos*. México: Pearson Prentice Hall.

Actividad 2:

- **Actividad de aprendizaje**

Basándose en los principales objetivos de los Sistemas Operativos indique al menos dos usos de acuerdo con cada objetivo.

- **Procedimiento**

Revise la Unidad 7 del texto guía: Castillo T. Cueva S. (2018). Texto guía de Arquitectura de Computadoras y Sistemas Operativos. Loja: Editorial Universidad Técnica Particular de Loja.



Autoevaluación 7

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. Un sistema operativo:
 - a. Proporciona comunicación entre los procesadores, la memoria principal y los módulos de salida.
 - b. Es un conjunto de programas destinados a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera eficiente.
 - c. Transfiere datos entre el computador y su entorno externo.
2. Los objetivos de un sistema operativo son:
 - a. Minimizar el uso de los recursos y brindar comodidad al usuario
 - b. Permitir el acceso a los recursos del sistema y garantizar seguridad a los procesos
 - c. Brindar comodidad al usuario y lograr la operación eficiente del sistema
3. Desde el punto de vista de usuario: Un sistema operativo es diseñado para:
 - a. Maximizar la utilización de recursos.
 - b. Usabilidad individual y maximizar la utilización de recursos.
 - c. Asignar Recursos.

4. El sistema operativo como programa de control
 - a. Actúa como el administrador de los recursos hardware de la computadora.
 - b. Evita las solicitudes conflictivas de los recursos hardware de la computadora
 - c. Gestiona la ejecución de los programas de usuario para evitar errores y mejorar el uso de la computadora.
5. La multiprogramación:
 - a. Incrementa el uso de la CPU organizando los trabajos de modo que la CPU siempre tenga uno que ejecutar.
 - b. Solo se ejecuta un proceso o trabajo a la vez.
 - c. El sistema operativo mantiene en memoria solo un trabajo o proceso.
6. ¿Qué tipo de procesamiento fue el predominante en la segunda generación sistemas informáticos?
 - a. Procesamiento por lotes.
 - b. Tiempo real.
 - c. Tiempo compartido.
7. Los núcleos de Linux son casos de:
 - a. Núcleos monolíticos.
 - b. Micronúcleos.
 - c. Sistemas por capas.
8. Los sistemas monolíticos:
 - a. Se descompone en varios niveles, cada uno de ellos está construido sobre capas inferiores.
 - b. No existe una estructura propiamente dicha o es mínima.
 - c. Abstraer el hardware de la computadora, formando varios entornos de ejecución.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

9. Los sistemas por capas:

- a. Abstraer el hardware de la computadora, formando varios entornos de ejecución.
- b. Se descompone en varios niveles, cada uno de ellos está construido sobre capas inferiores.
- c. No existe una estructura propiamente dicha o es mínima.

10. Los sistemas microkernels

- a. Instrumentan copias exactas del hardware simple.
- b. Proporcionan la comunicación entre los clientes y servidores utilizando el paso de mensajes.
- c. Proporcionan un mecanismo de comunicaciones entre el programa cliente y los distintos servicios que se ejecutan en el espacio de usuario.

[Ir al solucionario](#)

Resultados de aprendizaje 17 y 18

- Explica la concurrencia en relación con la planificación y despacho.
- Utiliza de forma correcta los scripts para automatizar tareas del sistema operativo.

Contenidos, recursos y actividades recomendadas



Semana 10



Unidad 8. Planificación de procesos

En esta unidad se analizan conceptos básicos de los procesos, así como la planificación de procesos. Los invitamos a que revisen con detalle cada uno de los conceptos brindados en la sección 8.1 para que pueda aplicarlos con facilidad en la sección 8.2 utilizando los algoritmos de planificación de procesos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8.1. Introducción a Procesos

En el capítulo 7, se había mencionado que una de las funciones de los sistemas operativos es gestionar los procesos. La mayoría de los requisitos que un sistema operativo debe cumplir se expresan con referencia a los procesos, Stallings (2005) menciona los siguientes.

- Intercalar la ejecución de múltiples procesos, para maximizar la utilización del procesador mientras se proporciona un tiempo de respuesta razonable.
- Reservar recursos para los procesos conforme a una política específica mientras que al mismo tiempo evita interbloqueos.
- Requerir dar soporte a la comunicación entre procesos y la creación de procesos, mediante las cuales ayuda a la estructuración de las aplicaciones.

Por otra parte, Silberschatz, Galvin y Gagne (2006), indican que el sistema operativo es el responsable de las siguientes actividades con respecto a la administración de procesos.

- Crear y eliminar procesos de usuario y procesos del sistema.
- Suspender y continuar procesos.
- Proporcionar mecanismos para la sincronización, comunicación e interbloqueo de procesos.

Pero ¿qué es un proceso? Les invitamos a revisar la siguiente sección en la cual se aborda el concepto de procesos.

8.1.1. Procesos

No existe una definición universal de procesos, por lo cual se mencionarán a continuación algunas de ellas:

- Unidad de trabajo en un sistema moderno de tiempo compartido (Silberschatz, Galvin, y Gagne, 2006).

- La entidad que se puede asignar y ejecutar en un procesador (Stallings, 2005).
- Programa en Ejecución (Tanenbaum y Andrew, 2009), (Stallings, 2005), (Silberschatz, Galvin y Gagne, 2006), (Castellanos, 2014).

En esta asignatura, se adoptará como definición de proceso la siguiente:

Un proceso es un programa en ejecución.

En un sistema informático se pueden estar ejecutando múltiples procesos; para listarlos se utiliza el comando “ps” en Linux y en Windows se usa el administrador de comandos.

En la siguiente sección se describen las operaciones que se realizan con los sistemas operativos.

8.1.2. Operaciones sobre procesos

El sistema operativo debe proporcionar mecanismos para que los procesos se puedan ejecutar en forma concurrente, es así que las operaciones que se realizan sobre los procesos son la creación y terminación; las mismas que se detallan a continuación.

8.1.2.1. Creación de procesos

Según Tanenbaum y Andrew (2009), la creación de un proceso surge por los siguientes eventos.

- Arranque del sistema.
- Ejecución, desde un proceso de una llamada al sistema para la creación de procesos.
- Petición de usuario para crear un proceso.
- Inicio de un trabajo por lotes.

Un proceso puede crear varios procesos nuevos, a través de una llamada al sistema para la creación de procesos, durante el curso de la ejecución. El proceso creador se denomina padre mientras que los procesos nuevos son denominados procesos hijos (Silberschatz, Galvin y Gagne, 2006).

8.1.2.2. Terminación de procesos

Un proceso puede terminar su ejecución en un momento determinado siempre y cuando se cumpla alguna condición.

Por lo general los procesos se terminan debido a que han concluido su trabajo; en este caso el compilador que ha compilado el proceso ejecuta una llamada al sistema para indicar al sistema operativo que ha terminado.

Para Tanenbaum y Andrew (2009), estas condiciones son:

- Salida normal (voluntaria).
- Salida por error (voluntaria).
- Error fatal (involuntaria).
- Eliminado por otro proceso (involuntaria).

Para Silberschatz, Galvin y Gagne (2006), un proceso padre puede terminar la ejecución de uno de sus hijos por varias razones, entre las que se encuentran:

- El hijo se ha excedido en el uso de alguno de los recursos que tiene asignados.
- Ya no se requiere la tarea asignada al hijo.
- El padre está saliendo, y el sistema no permite que un hijo continúe si su padre ha terminado.

A continuación, se detallan los estados de los procesos.

8.1.3. Estados de un proceso

En un sistema multiprogramado, pueden existir muchos procesos y un solo procesador, por lo cual podrá ocurrir que en un momento determinado solo un proceso se ejecute y los otros procesos deban esperar a que se les asigne la CPU; además, la interacción de un proceso con otros procesos puede generar una salida que puede ser utilizada por otro proceso como entrada; estos cambios producen los estados.

El estado del proceso se define por la actividad en que se encuentra en un momento determinado. Los estados de un proceso pueden ser:

- Nuevo: creación de un proceso.
- Preparado: el proceso está esperando a que se le asigne un procesador.
- Ejecución: el proceso está utilizando la CPU en ese instante.
- Bloqueado: el proceso está esperando a que se produzca un evento externo, como una llamada de E/S y pasaría a estado de ejecución.
- Finalizado: el proceso terminó su ejecución.

En la figura 77, se representan las transiciones de los estados de los procesos:

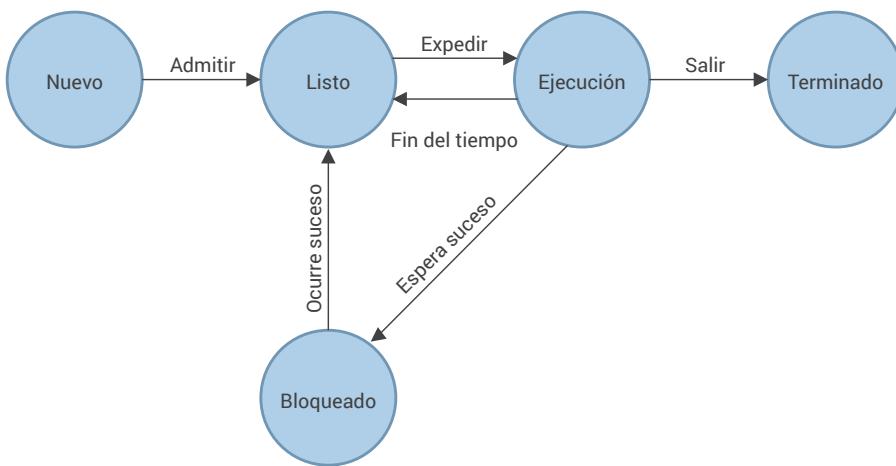


Figura 77. Estados de procesos.

Fuente: Stallings (2005).

Las transiciones de estados que pueden ocurrir son:

- **Nuevo a listo:** cuando se crea un proceso pasa inmediatamente al estado listo.
- **Listo a ejecución:** el proceso que se encuentra en estado listo, espera para que se le asigne un procesador para ejecutarse; al liberarse el procesador el planificador selecciona el proceso a ejecutarse de acuerdo al criterio definido o al algoritmo utilizado.
- **Ejecutado a listo:** cuando se origina una interrupción, el proceso puede perder el procesador y pasar al estado de listo. El planificador es el encargado de seleccionar el próximo proceso a ejecutar.
- **Ejecutado a bloqueado:** en la ejecución de los procesos puede solicitar algunos componentes (por ejemplo, dispositivos de

E/S). Considerando que la solicitud puede tardar en asignarse, el proceso es puesto en una cola de espera hasta que se complete su solicitud.

- **Bloqueado a listo:** cuando se asigna los recursos solicitados por el proceso, se lo coloca nuevamente en la cola de procesos listos.
- **Ejecutado a terminado:** ocurre cuando el proceso ejecuta su última instrucción y pasa al estado de terminado, liberando todas las estructuras utilizadas por el proceso.

Como se había comentado anteriormente existen procesos concurrentes que se ejecutan en el sistema operativo, los cuales pueden ser independientes o cooperativos, revisen el siguiente apartado para revisar las características de estos procesos.

8.1.4. Bloque de control de procesos

El bloque de control de procesos (PCB de *Process Control Block*), contiene elementos de información asociados a un proceso específico.

Según Silva (2015), el PCB está formado por los siguientes elementos.

- **Estado del proceso:** el estado actual en el que se encuentra el proceso.
- **Contador de programa:** indica la dirección de la siguiente instrucción que se ejecutará para el proceso.
- **Registros de CPU:** dependen de la arquitectura de la computadora, entre los que se pueden mencionar acumuladores, registros índices, punteros de pila y registros de propósito general.

- **Información de planificación de la CPU:** contiene la información de la prioridad del proceso y cualquier información adicional a la planificación del proceso.
- **Información de gestión de memoria:** incluye datos como los valores de los registros base, registro límite, las tablas de página, las tablas de segmentos, etc., dependiendo del esquema de memoria que utilice el sistema operativo.

8.1.5. Procesos Independientes y cooperativos

Los **procesos independientes** son los que no pueden afectar o verse afectados por los otros procesos que se están ejecutando en el sistema. Es decir, los procesos independientes NO comparten datos con ningún otro proceso.

Según Stallings (2005), aunque los procesos independientes no comparten datos, el sistema operativo tiene que encargarse de la competencia por los recursos; en este caso, los resultados de un proceso son independientes de las acciones de otros procesos, pero la ejecución de un proceso puede influir en el comportamiento de los procesos que compiten, es decir se pueden ver afectados sus tiempos de ejecución.

Silberschatz, Galvin, y Gagne (2013), mencionan que los **procesos cooperativos**, son los que si afectan o pueden verse afectados por los otros procesos que se están ejecutando en el sistema.

Cualquier proceso que comparta datos con otros procesos es un proceso cooperativo.

Las razones para proporcionar un ambiente que permita la cooperación entre procesos son:

- *Compartir información*: varios usuarios pueden necesitar el mismo recurso, por lo cual se debe proporcionar un ambiente que permita el acceso concurrente los recursos.
- *Aceleración de cálculos*: debido a la ejecución paralela de subtareas para la ejecución más rápida de una tarea en particular.
- *Modularidad*: construcción de un sistema en forma modular, para lo cual se dividen las funciones del sistema en procesos distintos o hilos.
- *Conveniencia*: cuando un usuario tiene varias tareas ejecutándose en un momento determinado.

8.1.6. Comunicación entre procesos

La comunicación entre procesos cooperativos requiere que comparten una reserva común de buffers y que el programador de la aplicación escriba de manera explícita el código para implementar el buffer. Otra forma es hacerlo a través del servicio de comunicación entre procesos (IPC) (Silberschatz, Galvin y Gagne, 2013).

8.2. Planificación de la CPU

El sistema operativo es el encargado de asignar los recursos de la computadora entre las necesidades de los múltiples procesos; por lo cual el procesador se asigna a través de la planificación.

Como ya se había mencionado, el implementar multiprogramación permite maximizar el aprovechamiento de la CPU; para ello, es necesario que el procesador mantenga en cada momento un

proceso en ejecución; sin embargo, es necesario tener en cuenta que en algún momento dado tendrá que ocurrir una conmutación de contexto (cambio de proceso) debido a la terminación de este, o por la necesidad de esperar a que termine de atenderse una solicitud de E/S.

Stallings (2005) indica que la planificación puede ser de diferentes tipos: planificación a largo, mediano y corto plazo; a continuación, se detallan cada una de ellas.

- **Planificación a largo plazo:** en la cual se decide añadir un proceso al conjunto de procesos a ser ejecutados.

Stallings (2005), menciona que el planificador a largo plazo determina que programas se admiten en el sistema para su procesamiento, con lo cual se controla el grado de multiprogamación.

- **Planificación a medio plazo:** decisión de añadir un proceso al número de procesos que están parcial o totalmente en la memoria principal. Constituye una parte de la función de intercambio.
- **Planificación a corto plazo:** decisión por la que un proceso disponible será ejecutado por el procesador. Es conocido también como activador, ejecuta mucho más frecuentemente y toma las decisiones de grano fino sobre qué proceso se debe ejecutar.

El propósito principal de la planificación es asignar procesos a ser ejecutados por el procesador.

Los algoritmos de planificación pueden ser en general:

- **No expropiativos:** denominada también como no apropiativa, significa que una vez que a un proceso se le ha asignado la CPU, este se ejecutará hasta que termine de ejecutarse o hasta que termine una solicitud de E/S.

- **Expropiativos o apropiativos:** consisten en que un proceso se está ejecutando en el procesador hasta que es sustituido por otro proceso de mayor prioridad a través de una interrupción, pasando a la cola de procesos listos, en espera de que se le asigne el procesador nuevamente.

Silva (2015) menciona que hay supuestos básicos en torno a los algoritmos de planificación los cuales se detallan a continuación.

- Existe un grupo de procesos ejecutables que compiten por el uso de la CPU.
- Los procesos son independientes y compiten por los recursos.
- El trabajo del planificador es el de distribuir el uso de la CPU a los diferentes procesos de manera equitativa y de tal forma que se optimice el rendimiento de la misma.

8.2.1. Criterios de planificación

Existen algunos criterios que se deben considerar para seleccionar el algoritmo de planificación adecuado, es importante que los comprenda para que tome la decisión adecuada cuando deba seleccionar un algoritmo.

En la tabla 8, se resumen los criterios que señalan algunos autores que se deben considerar:

Tabla 8. *Criterios de un algoritmo de planificación*

CRITERIO	DESCRIPCIÓN
Equidad	Garantizar que cada proceso obtenga su proporción justa de la CPU (La Red Martínez, 2001) (Tanenbaum y Andrew, 2009).
Eficacia	Mantener ocupada la CPU al 100% del tiempo (La Red Martínez, 2001).
Tiempo de respuesta	Minimizar el tiempo de respuesta para los usuarios interactivos. (La Red Martínez, 2001). Responder a las peticiones con rapidez (Tanenbaum y Andrew, 2009).

CRITERIO	DESCRIPCIÓN
Tiempo de espera	Minimizar el tiempo que deben esperar los usuarios por lotes (batch) para obtener sus resultados (La Red Martínez, 2001).
Tiempo de retorno	Minimizar el tiempo entre la entrega y la terminación (Tanenbaum y Andrew, 2009). Es el tiempo estadísticamente promedio desde el momento en que se envía un trabajo por lotes, hasta el momento en que se completa, con lo cual se mide el tiempo que debe esperar el usuario promedio la salida.
Rendimiento	Maximizar el número de tareas procesadas por hora (La Red Martínez, 2001), (Tanenbaum y Andrew, 2009).

Es importante aprovechar al máximo el uso de los recursos de cómputo por ello, al momento de diseñar un algoritmo de planificación se deberá observar un equilibrio entre los diferentes criterios, pues en muchos casos se tendrá que sacrificar alguno para obtener otro de ellos al 100%.

8.2.2. Algoritmos de planificación

A continuación, se van a analizar los algoritmos de planificación más utilizados en los diferentes sistemas operativos.

8.2.2.1. Algoritmo FIFO (Primero en entrar, primero en salir)

El algoritmo FIFO (*First in First out*), es no apropiativo debido a que asigna los procesos la CPU de acuerdo con el orden en el que se solicita. Es decir, según el orden de llegada a la cola de listos es asignada la CPU.

Al decir de Tanenbaum y Andrew (2009), es el más simple de todos los algoritmos de planificación, es fácil de comprender e igualmente sencillo de programar; también es equitativo.

El algoritmo FIFO tiene como desventaja sufrir el efecto convoy, es decir puede que un proceso que se ejecute necesite utilizar la

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

CPU, por un largo periodo de tiempo con lo cual los otros procesos quedarán por un largo periodo de tiempo en estado de inanición.

Otra de las desventajas es que puede resultar poco eficiente.

A continuación, se desarrollará un ejercicio de planificación de procesos utilizando el algoritmo FIFO.

Realice el diagrama de Gantt y calcule el tiempo de espera y tiempo de retorno de cada proceso.

Proceso	Ráfaga de CPU	Tiempo de Llegada
A	3	2
B	1	4
C	3	0
D	4	1
E	2	3

Se verifica cual es el proceso que ha llegado primero a la lista para poder ejecutar en este caso es el proceso C, luego se debe ejecutar el proceso D debido a que tiene el tiempo de llegada 1 y se sigue la misma secuencia con el resto de procesos. A continuación se muestra el diagrama de Gannt resultante.

C	D	A	E	B
---	---	---	---	---

0 3 7 10 12 13

Tiempo de espera

$$\begin{aligned} A &= (7-2) = 5 \\ B &= (12-4) = 8 \\ C &= (0-0) = 0 \\ D &= (3-1) = 2 \\ E &= (10-3) = 7 \end{aligned}$$

Tiempo de retorno

$$\begin{aligned} A &= 10 \\ B &= 13 \\ C &= 3 \\ D &= 7 \\ E &= 12 \end{aligned}$$

Tiempo media de espera

$$TEM = (5+8+0+2+7)/5 = 4,4$$

Tiempo de retorno medio

$$TRM = (10+13+3+7+12)/5 = 9$$

8.2.2.2. Algoritmo SJF (Primero el trabajo más corto)

El algoritmo SJF, o primero el trabajo más corto, consiste en seleccionar al proceso que tiene menos duración; esta planificación es teóricamente óptima para los tiempos medios de respuesta, retorno y espera.

Además, es un algoritmo no apropiativo, que supone que los tiempos de ejecución se conocen de antemano (Tanenbaum y Andrew, 2009).

Cuando hay procesos que tienen la misma ráfaga de CPU, se debe elegir el que llegó primero.

A continuación, se mostrará un ejemplo de cómo aplicar el algoritmo SJF.

Proceso	Tiempo de Llegada	Ráfaga de CPU
P1	3	3
P2	1	4
P3	5	2

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

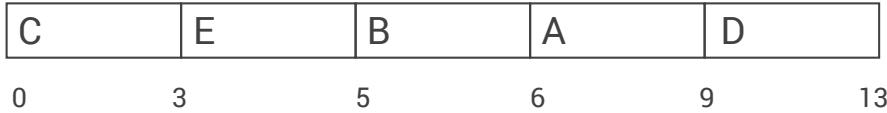
El diagrama de Gantt, resultante es el siguiente.



Tiempo de Espera:	Tiempo de Retorno
P1 (7-3)=4	P1= 10
P2 (1-1)=0	P2 = 5
P3 (5-5)=0	P3= 7
Tiempo de Espera Promedio= 1,33	Tiempo de Retorno Promedio= 7,33

Existe una variante, que se denomina SRTN (*Shortest Remaining Time Next* o menor tiempo restante), el cual es un algoritmo apropiativo donde el planificador siempre selecciona el proceso cuyo tiempo restante de ejecución sea el más corto (Castellanos, 2014).

A continuación, se muestra la ejecución del algoritmo SJF, considerándolo como apropiativo, es decir que si llega a la cola de listos un proceso que tenga menor tiempo de ráfaga de CPU se apropiá de la CPU.



Tiempo de espera Tiempo de retorno

$$\begin{array}{ll} A = (6-2) = 4 & A = 9 \\ B = (5-4) = 1 & B = 6 \\ C = (0-0) = 0 & C = 3 \\ D = (9-1) = 8 & D = 13 \\ E = (3-3) = 0 & E = 5 \end{array}$$

Tiempo media de espera Tiempo de retorno medio

$$TEM = (4+1+0+8+0)/5 = 2,6 \quad TRM = (9+6+3+13+5)/5 = 7,2$$

8.2.2.3. Algoritmo por prioridad

El algoritmo por prioridad consiste en asignar una prioridad a cada proceso, siendo el proceso que posee la prioridad más alta el que se ejecuta.

En el caso de que existan dos procesos con la misma prioridad se tendría que ejecutar el proceso que primero llegó a la cola de listos, es decir se aplica también la política de FIFO.

Proceso	Ráfaga de CPU	Tiempo de Llegada	Prioridad
A	3	2	2
B	1	4	3
C	3	0	1
D	4	1	3
E	2	3	4



Tiempo de espera

$$\begin{aligned}A &= (3-2) = 1 \\B &= (10-4) = 6 \\C &= (0-0) = 0 \\D &= (6-1) = 5 \\E &= (11-3) = 8\end{aligned}$$

Tiempo de retorno

$$\begin{aligned}A &= 6 \\B &= 11 \\C &= 3 \\D &= 10 \\E &= 13\end{aligned}$$

Tiempo media de espera

$$TEM = (1+6+0+5+8)/5 = 4$$

Tiempo de retorno medio

$$TRM = (6+11+3+10+13)/5 = 8,6$$

8.2.2.4. Algoritmo Round Robin (espera circular)

Este algoritmo también se conoce como planificación por turno circular.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

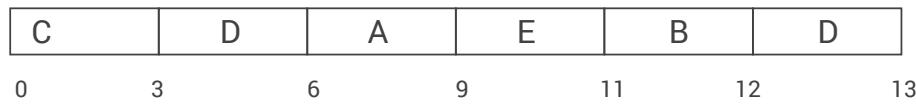
Según Tanenbaum y Andrew (2009), es uno de los algoritmos más antiguos, simples, equitativos y de mayor uso.

Consiste en que a cada proceso se le asigna un intervalo de tiempo, conocido como quantum (q), durante el cual el proceso puede ejecutarse, si al proceso le falta tiempo para ejecutarse se coloca de nuevo al final de la cola de listos, es decir los procesos se ejecutan en orden del algoritmo FIFO pero solo se le permite utilizar la CPU el tiempo del quantum.

Se recomienda que el quantum no sea muy pequeño, debido a que pueden ocurrir muchos cambios de contexto. Recuerde que un cambio de contexto es el cambio de la CPU de un proceso a otro.

A continuación, se realiza un ejemplo con el algoritmo RR, considerando un $q = 3$.

Proceso	Ráfaga de CPU	Tiempo de Llegada
A	3	2
B	1	4
C	3	0
D	4	1
E	2	3



Tiempo de espera

$$\begin{aligned}A &= (6-2) = 4 \\B &= (11-4) = 7 \\C &= (0-0) = 0 \\D &= (3-1) + 6 = 8 \\E &= (9-3) = 6\end{aligned}$$

Tiempo de retorno

$$\begin{aligned}A &= 9 \\B &= 12 \\C &= 3 \\D &= 13 \\E &= 11\end{aligned}$$

Tiempo de espera

$$TEp = 5$$

Tiempo de retorno

$$TRP = 9,6$$



Actividad de aprendizaje recomendada

Actividad 1

- **Actividad de aprendizaje**

Con base en los criterios de planificación existentes realice un análisis comparativo de los diferentes algoritmos de planificación.

- **Procedimiento**

- Realice un cuadro comparativo de la planificación de corto, medio y largo plazo.
- Desarrolle los ejercicios que se planteen a través de la plataforma web.

Estrategias de desarrollo

Se recomienda utilizar como base los siguientes recursos.

Silberschatz, A., Galvin, A., y Gagne, G. (2006). *Fundamentos de sistemas operativos*. Madrid: España: McGraw-Hill Interamericana.

Tanenbaum, A. (2009). *Sistemas Operativos Modernos*. Ciudad de México, México: Pearson Prentice Hall.



Autoevaluación 8

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. Un proceso es:
 - a. Un programa
 - b. El código de un programa
 - c. Un programa en ejecución
2. El planificador a largo plazo:
 - a. Controla el grado de multiprogramación del sistema.
 - b. Reduce la contienda por el uso de la CPU.
 - c. Se lo conoce como planificador de la CPU
3. La conmutación de contexto es:
 - a. Se la considera como tiempo útil.
 - b. Se la considera como tiempo muerto ya que implica un gasto extra de tiempo.
 - c. El proceso está esperando a que se produzca un suceso.
4. Los procesos que afectan o pueden verse afectados por los demás procesos se los conocen como:
 - a. Competitivos
 - b. Cooperativos
 - c. Paralelos

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. El bloque de control de proceso está formado por varios elementos entre ellos el de información de planificación de la CPU, el mismo que:
 - a. Es el contador que indica la dirección de la siguiente instrucción que va a ejecutar dicho proceso.
 - b. Incluye información de la prioridad del proceso, los punteros a las colas de planificación y otros parámetros de planificación que se requieren.
 - c. Incluye información acerca del valor de los registros base y límite, las tablas de páginas o de segmentos, dependiendo del mecanismo de gestión de la memoria utilizado por el SO.
6. Un proceso que está utilizando la CPU en un instante dado está:
 - a. En ejecución.
 - b. Bloqueado
 - c. Listo.
7. El algoritmo de planificación de procesos que realiza una asignación equitativa del tiempo de procesador es:
 - a. SJF
 - b. Round Robin
 - c. Por prioridad

8. Considere el siguiente conjunto de procesos, con el tiempo de ráfaga de CPU dada en milisegundos:

Proceso	Tiempo de ráfaga	Prioridad
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

¿Cuál es el tiempo de retorno del proceso P3 utilizando el algoritmo de planificación FIFO (Primero en entrar primero en salir)?

- a. 10 ut
 - b. 14 ut
 - c. 13 ut
9. El algoritmo de planificación de procesos que asocia con cada proceso la longitud de su siguiente ráfaga de CPU.
- a. Por prioridad.
 - b. FIFO
 - c. SJF
10. El tiempo de espera es:
- a. El intervalo que va desde el instante en que se ordena la ejecución de un proceso hasta el instante en que se completa.
 - b. Es la suma de los periodos invertidos en esperar en la cola de procesos preparados.
 - c. Es el tiempo que el proceso tarde en empezar a responder.

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Semana 11



Unidad 9. Sincronización de Procesos

9.1. Introducción

El propósito de esta unidad es identificar el problema de sincronización de procesos y los medios para realizarla.

La sincronización entre procesos es esencial para evitar los problemas de temporización relacionados con la concurrencia, por lo cual los procesos se deben sincronizar cuando se van a utilizar recursos compartidos, con espacios de direcciones comunes o dispositivos físicos como, por ejemplo, un servidor de impresión para varias terminales de red.

La concurrencia es fundamental en el diseño de un sistema operativo, abarca varios aspectos entre los cuales se puede mencionar la comunicación entre procesos y la compartición de los recursos; siendo este aspecto lo que conlleva a que el diseño de un sistema operativo contemple la sincronización de procesos para que no se queden bloqueados y puedan ser atendidos equitativamente.

Milenkovic (1988), menciona que debido a que el sistema operativo no conoce ni necesita conocer la semántica de las actividades del proceso, los procesos cooperativos son los que deben encargarse de sincronizar adecuadamente sus operaciones; sin embargo, esto no significa que los usuarios, el lenguaje de programación y hasta el mismo sistema operativo multitarea no proporcionen una sincronización entre procesos.

9.2. Problemas de la sección crítica

La sección crítica se refiere a la sección de código dentro de un proceso que requiere acceso a recursos compartidos y que no puede ser ejecutada, mientras otro proceso esté en una sección de código correspondiente (Stallings, 2005).

Silva (2015), denomina a la sección crítica como el segmento de código en el cual un hilo está accediendo en exclusividad a un recurso compartido (una variable, una estructura de datos, un dispositivo) y que no debe ser accedido concurrentemente por otro.

Además, la sección crítica delimita la actualización de una o más variables compartidas, de tal forma que cuando un proceso se encuentra en una región o sección crítica, se debe garantizar que ningún otro proceso entre a ejecutarse en esa sección mientras el proceso inicial completa todas las instrucciones.

El problema de la sección crítica según Silberschatz, Galvin, y Gagne (2006), consiste en elegir un protocolo que puedan usar los hilos para cooperar. Una solución a este problema es que puedan satisfacer los siguientes tres requerimientos:

- *Exclusión mutua:* si el hilo T_i se está ejecutando en su sección crítica, entonces ningún otro hilo puede estar en la misma ejecución.

- *Progreso:* si ningún hilo se está ejecutando en su sección crítica y existen algunos hilos que desean entrar a sus secciones críticas, entonces solo aquellos hilos que no se están ejecutando en su sección no crítica pueden participar en la decisión acerca de cuál entrará a su sección crítica a continuación, y esta selección no puede posponerse de manera indefinida.
- *Espera limitada:* existe un límite sobre el número de veces que otros hilos tienen permitido entrar a su sección crítica después de que un hilo ha hecho una solicitud para entrar a su sección crítica y antes de que se conceda dicha solicitud, con lo cual se evita la inanición de cualquier hilo individual.

Los requerimientos para solucionar el problema de la sección crítica son: exclusión mutua, progreso, espera limitada.

Se puede dar el caso de que, en un tiempo determinado, pueden estar activos muchos procesos en modo *kernel*; como resultado, el código que implementa el sistema operativo está sujeto a varias posibles condiciones de competencia. En la figura 78 se muestra la estructura general de un proceso.

```
Do {
    Sección de entrada
        Sección crítica
    Sección de salida
        Sección restante
} while TRUE;
```

Figura 78. Estructura general de un proceso típico Pi.
Fuente: Silberschatz, Galvin y Gagne (2006)

A continuación, se analizará el *hardware* de sincronización, por lo cual le invitamos a que revise la siguiente sección.

9.3. Hardware de sincronización

En esta sección se presentan algunas instrucciones de *hardware* que se pueden usar para resolver el problema de la sección crítica, considerando que el soporte del *hardware* puede facilitar cualquier tarea de programación y mejorar la eficiencia del sistema.

Una opción es la instrucción TestAndSet para leer y modificar atómicamente una variable, esta instrucción se caracteriza porque se ejecuta atómicamente. Según Silberschatz, Galvin, y Gagne (2006), si dos instrucciones TestAndSet se ejecutan simultáneamente (cada una en una CPU diferente), se ejecutarán secuencialmente en un orden arbitrario. A continuación, se muestra el código de la instrucción.

```
boolean test_and_set(boolean*target) {  
    boolean rv = *target;  
    *target = true;  
  
    return rv;  
}
```

Si la máquina soporta la instrucción TestAndSet, se puede implementar la exclusión mutua declarando una variable booleana *lock*, la cual se inicializa con el valor de *false*, a continuación se muestra el código de esta instrucción proporcionada por (Silberschatz, Galvin y Gagne, 2013).

```
do {  
    while (test_and_set(&lock))  
        /*do nothing */  
  
    /* critical section */  
  
    lock = false;  
  
    /* remainder setion */  
} while (true);
```

Además, Stallings (2005) indica otra solución de *hardware* a la exclusión mutua que es la deshabilitación de interrupciones, la cual para garantizar la exclusión mutua, debe impedir que un proceso sea interrumpido, esta técnica se puede proporcionar en forma de primitivas definidas por el núcleo del sistema para deshabilitar y habilitar interrupciones. De la siguiente forma un proceso puede cumplir la exclusión mutua.

```
while (true)
{
    /*deshabilitar interrupciones */;
    /*sección crítica */;
    /*habilitar interrupciones */;
    /*resto*/;
```

En algunas ocasiones hay dificultad para utilizar las soluciones de *software* y *hardware* propuestas, por lo que se puede utilizar otra alternativa que son los semáforos, los cuales se analizan en la siguiente sección.

9.4. Semáforos

Un semáforo S es una variable entera a la que, solo se accede mediante dos operaciones atómicas estándar: *wait ()* y *signal ()* (Silberschatz, Galvin y Gagne, 2013).

Un semáforo es una variable especial o un tipo abstracto de datos, que constituye el método clásico para restringir o permitir el acceso a recursos compartidos en un entorno de multiprocesamiento en el que se ejecutarán varios procesos concurrentemente (Castellanos, 2014).

Los semáforos pueden ser de dos tipos.

- **Contador:** puede variar en un dominio no restringido (Silberschatz, Galvin y Gagne, 2013).

- **Binario:** es un semáforo simple que puede tomar solo los valores de 0 o 1; se inicializan en 1 y son usados cuando solo un proceso puede acceder a un recurso a la vez; cuando el recurso está disponible, un proceso accede y decrementa el valor del semáforo con la opción P. El valor queda entonces en 0, lo que hace que si otro proceso intenta decrementarlo tenga que esperar. Cuando el proceso que decrementó el semáforo realiza una operación V, otro proceso que estaba esperando comienza a utilizar el recurso (Castellanos, 2014).

En algunos sistemas los semáforos binarios también son conocidos como cerrojos mútex, ya que son cerrojos que proporcionan exclusión mutua.

Una de las desventajas de los semáforos es que requiere una espera activa.

Los semáforos son una herramienta sencilla para su implementación pero bastante potentes para asegurar la exclusión mutua, la sincronización entre procesos y también la señalización.

9.5. Problemas clásicos de sincronización

En esta sección se examinarán algunos de los problemas de sincronización de procesos más conocidos.

9.5.1. El problema del búfer limitado h5

En el problema de búfer limitado existen dos tipos de procesos: productores cuyo objetivo es generar algún tipo de información y almacenarla en un búfer compartido por todos los procesos y procesos consumidores, cuyo objetivo es extraer la información almacenada en el búfer por los procesos productores para llevar a cabo algún tipo de procesamiento sobre la misma (Candela, García, Quesada, Santana y Santos, 2007).

Según (Silberschatz, Galvin, y Gagne, 2013), este problema es utilizado para demostrar la potencia de las primitivas de sincronización, suponga que la cola de procesos consta de n búferes, siendo cada uno capaz de almacenar un elemento, el cual es colocado por un productor método *enter* (), los consumidores remueven elementos invocando al método *remove* ().

En la figura 79, se muestra el código de los procesos productor y consumidor.

Estructura del proceso productor

```
do {
    . . .
    /*produce an item in next_produced */
    . . .
    wait (empty);
    wait (mutex);
    . . .
    signal (mutex);
    signal (full);
}while (true);
```

Estructura del proceso consumidor

```
do {
    wait (full);
    wait (mutex);
    . . .
    /* remove an item from buffer to next_consumed */
    . . .
    signal (mutex);
    signal (empty);
    . . .
    /* consume the item in next_consumed */
    . . .
} while (true);
```

Figura 79. Estructura del proceso productor y consumidor.

Fuente: Silberschatz, Galvin y Gagne (2013)

9.5.2. El problema de los filósofos comensales

Dijkstra en el año 1965, propuso y resolvió el problema de sincronización de procesos conocido como el problema de los filósofos comensales o comedores, el cual consiste en:

Hay cinco filósofos que están sentados alrededor de una mesa circular, cada filósofo tiene un plato de espagueti; el espagueti es tan resbaloso, que un filósofo necesita dos tenedores para comerlo; entre cada par de platos hay un tenedor (Tanenbaum y Andrew, 2009).

Se debe considerar que la vida de un filósofo consiste en dos períodos alternos de comer y pensar, cuando un filósofo tiene hambre, tiene que adquirir el tenedor izquierdo y derecho, uno a la vez, en cualquier orden; si ha podido conseguir los tenedores puede comer por un momento, después deja los tenedores y continúa pensando.

En la figura 80, se representa la distribución de la mesa de los filósofos comensales

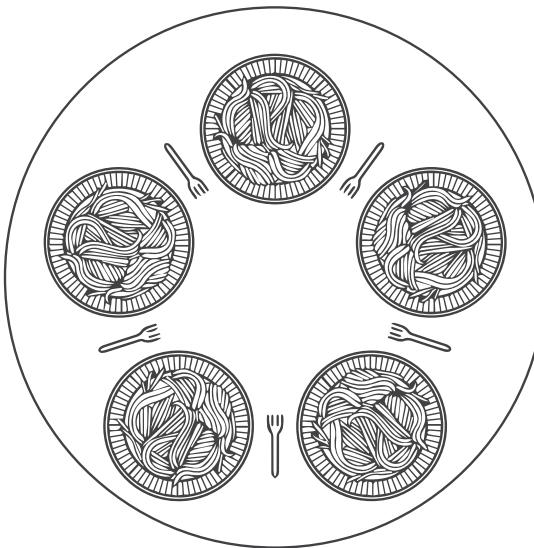


Figura 80. Problema de los filósofos comensales.

Fuente: Tanenbaum (2009)

9.5.3. El problema de los lectores escritores

Consiste en que existe algún determinado objetivo al cual pueden acceder a utilizarlo y compartirlo una serie de procesos concurrentes. Algunos procesos ingresarán y solo accederán al objeto sin modificarlo en cambio otros modificarán su contenido; esta actualización implica leerlo, modificar el contenido y escribirlos; de ahí que los primeros procesos se denominan lectores y los segundos escritores.

Las restricciones a este problema según Silva (2015) son:

- Solo se permite que un escritor tenga acceso al objeto al mismo tiempo, mientras que el escritor esté accediendo al objeto, ningún otro proceso lector ni escritor podrá acceder a él.
- Sin embargo, es permitido que múltiples lectores tengan acceso al objeto, ya que ellos no modifican el contenido del mismo.

9.5.4. El problema del peluquero dormilón

En una peluquería se tiene una sala de espera separada del salón de peinados; la sala de espera tiene una entrada y cerca de ella una puerta que conduce a la habitación donde se corta y peina; ambas comparten la misma puerta deslizante por lo cual siempre se cierra una de ellas.

Cuando el peluquero ha finalizado su corte de cabello, abre la puerta a la sala de espera y la inspecciona; si la sala de espera está vacía, invita al próximo cliente, sino se va a dormir en una de las sillas de espera. Por su parte los clientes cuando se encuentran cero o más clientes esperan en la sala su turno, pero si se encuentran al peluquero dormido lo despiertan.

9.6. Monitores

Un monitor es un módulo consistente en uno o más procedimientos, una secuencia de inicialización y datos locales.

Según Stallings (2005), las principales características de un monitor son las siguientes:

- Las variables locales de datos son solo accesibles por los procedimientos del monitor y no por ningún procedimiento.
- Un proceso entra en el monitor invocando uno de sus procedimientos.
- Solo un proceso puede estar ejecutándose dentro del monitor al tiempo; cualquier otro proceso que haya invocado al monitor se bloquea, en espera de que el monitor quede disponible.

Silva (2015), menciona que los monitores son más fáciles de controlar que los semáforos y se implementan como bibliotecas de programas.

Un problema con los monitores es que están diseñados para resolver el problema de exclusión mutua en una o más CPU que tengan acceso a una memoria común (Castellanos, 2014).



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Cite los principales problemas de sincronización de procesos e indique brevemente sus características

- **Procedimiento**

Revise la unidad 9 del texto-guía Vista previa del documento:
Castillo, T., y Cueva, S. (2018). *Arquitectura de computadoras y sistemas operativos*. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Actividad 2:

- **Actividad de aprendizaje**

Realice un cuadro comparativo entre semáforos y monitores; un aspecto de la comparación debe considerar scripts para semáforos y monitores.

- **Procedimiento**

Utilizar como base los siguientes recursos:

Silberschatz, A., Galvin, A., y Gagne, G. (2006). *Fundamentos de sistemas operativos*. España: McGraw-Hill Interamericana.

Silva, M. (2015). *Sistemas Operativos*. Buenos Aires, Argentina: Alfaomega.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Stallings, W. (2005). *Sistemas operativos Aspectos internos y principios de diseños*. Madrid, España: Pearson Prentice Hall.

Tanenbaum y Andrew. (2009). *Sistemas Operativos Modernos*. México: Pearson Prentice Hall.

Actividad 3:

- **Actividad de aprendizaje**

Autoevaluación.

- **Procedimiento**

Analice las preguntas propuestas en la autoevaluación 9, si tiene dudas, puede revisar la retroalimentación de la misma al final de esta guía virtual.



Autoevaluación 9

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. La región crítica es:

- a. El conjunto de instrucciones que delimitan la actualización de una o más variables compartidas.
- b. Cuando un proceso permite temporalmente a los demás procesos utilizar la sección compartida.
- c. El conjunto de instrucciones que permiten trabajar con secciones compartidas.

2. El requisito de exclusión mutua se refiere a:

- a. Existe un límite en el número de veces que se permite que otros procesos entren en sus secciones críticas y después de que un proceso haya hecho una solicitud para entrar en sus secciones críticas y antes de que la misma haya sido concedida.
- b. Si ningún proceso está ejecutando su sección crítica y algunos procesos desean entrar en sus correspondientes secciones críticas, sólo aquellos procesos que no estén ejecutando sus secciones restantes pueden participar en la decisión de cuál será el siguiente que entre en su sección crítica y esta selección no se puede posponer indefinidamente.
- c. Si el proceso Pi está ejecutándose en su sección crítica, los demás procesos no pueden estar ejecutando sus secciones críticas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

3. Si el proceso se está ejecutando en su sección crítica los demás procesos no pueden estar ejecutándose en sus secciones críticas, significa que se encuentra en:
 - a. Exclusión mutua.
 - b. Sincronización.
 - c. Bloqueo mutuo.
4. El requisito de progreso se refiere a:
 - a. Existe un límite en el número de veces que se permite que otros procesos entren en sus secciones críticas y después de que un proceso haya hecho una solicitud para entrar en sus secciones crítica y antes de que la misma haya sido concedida.
 - b. Si ningún proceso está ejecutando su sección crítica y algunos procesos desean entrar en sus correspondientes secciones críticas, sólo aquellos procesos que no estén ejecutando sus secciones restantes pueden participar en la decisión de cuál será el siguiente que entre en su sección crítica y esta selección no se puede posponer indefinidamente.
 - c. Si el proceso P_i está ejecutándose en su sección crítica, los demás procesos no pueden estar ejecutando sus secciones críticas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. El requisito de espera limitada se refiere a:
 - a. Existe un límite en el número de veces que se permite que otros procesos entren en sus secciones críticas y después de que un proceso haya hecho una solicitud para entrar en sus secciones crítica y antes de que la misma haya sido concedida.
 - b. Si ningún proceso está ejecutando su sección crítica y algunos procesos desean entrar en sus correspondientes secciones críticas, sólo aquellos procesos que no estén ejecutando sus secciones restantes pueden participar en la decisión de cuál será el siguiente que entre en su sección crítica y esta selección no se puede posponer indefinidamente.
 - c. Si el proceso Pi está ejecutándose en su sección crítica, los demás procesos no pueden estar ejecutando sus secciones críticas.
6. Un semáforo es:
 - a. Una variable entera a la que, dejando a parte la inicialización, sólo se accede mediante dos operaciones atómicas estándar; wait () y signal ().
 - b. Tiene un conjunto de operaciones definidas por el programador que gozan de las características de exclusión mutua.
 - c. Asegura que sólo un proceso esté activo cada vez dentro del monitor.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

7. Un monitor es:
- Una variable entera a la que, dejando aparte la inicialización, sólo se accede mediando dos operaciones atómicas estándar; wait () y signal ().
 - Tiene un conjunto de operaciones definidas por el programador que gozan de las características de exclusión mutua.
 - Sirven para resolver diversos problemas de sincronización
8. El semáforo contador:
- Sólo puede ser 0 o 1.
 - Puede variar en un dominio.
 - Se conocen como cerrojos mútex.
9. La condición para resolver el problema de la sección crítica, que se exige que no puede haber más de un proceso trabajando en la sección crítica, es la condición de:
- Exclusión mutua.
 - Progreso.
 - Espera limitada.
10. ¿Cuál de estas afirmaciones es correcta?
- El problema de la sección crítica solo puede darse en sistemas monolíticos.
 - El problema de la sección crítica solo puede darse en sistemas multiprocesadores.

[Ir al solucionario](#)



Semana 12



Unidad 10. Interbloqueo de Procesos

10.1. Fundamentos de interbloqueo

Un interbloqueo o bloqueo mutuo de procesos se produce por la competición de procesos por los recursos del sistema.

Según Stallings (2005), un conjunto de procesos está interbloqueado cuando cada proceso del conjunto está bloqueado esperando un evento por lo general la liberación de algún recurso requerido por parte del proceso.

En la figura 81, se ilustra un caso en donde cuatro automóviles que han llegado aproximadamente al mismo tiempo a una intersección donde confluyen cuatro caminos, estos cuatro cuadrantes de la intersección son los recursos que hay que controlar. Considere que la circulación normal es que un automóvil en un cruce de cuatro caminos debe dar preferencia a otro automóvil que está a su derecha; lo cual funciona si existen solo dos o tres automóviles en la intersección.

Pero que sucede si los coches olvidan esta normativa y todos llegan en el mismo instante, cada uno posee un cuadrante y se abstendrá de cruzar la intersección, produciéndose un interbloqueo.

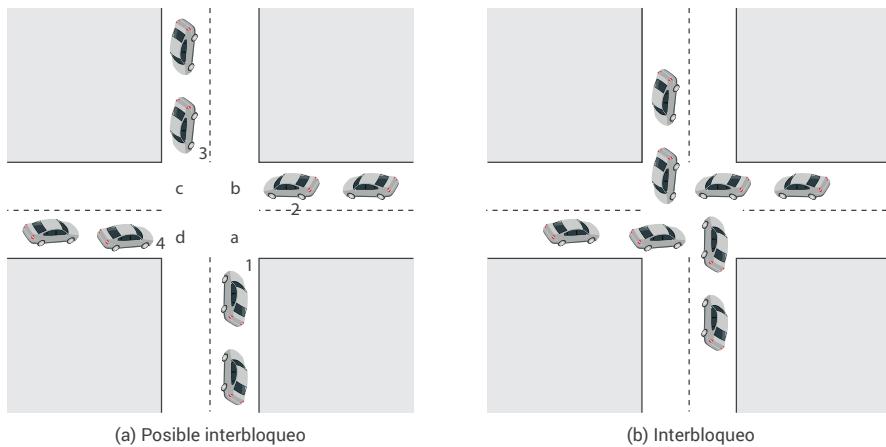


Figura 81. Interbloqueo de procesos.

Fuente: Stallings (2005)

Un ejemplo de bloqueo mutuo es la ley aprobada por la legislatura de Kansas a principios del siglo, en una parte de la misma decía: "Si dos trenes se aproximan uno al otro en un cruce, ambos harán alto total y ninguno arrancará de nuevo hasta que el otro se haya ido" (Silberschatz, Galvin, y Gagne, 2013).

Un sistema computacional está conformado por un número finito de recursos que deben distribuirse entre varios procesos que compiten;

Según Stallings (2005) dichos recursos se clasifican en:

- Recursos reutilizables: son aquellos que solo puede utilizar de forma segura un proceso en cada momento y que no se destruye después de su uso. Por ejemplo: procesadores, canales de E/S, memoria principal y secundaria, dispositivos, estructuras de datos, bases de datos, etcétera.

- Recursos consumibles: son los que pueden crearse (producirse) y destruirse (consumirse), no existe un límite en el número de recursos consumibles de un determinado tipo. Por ejemplo, las interrupciones, las señales, los mensajes y la información de *buffers* de E/S.

Cuando un sistema tiene dos CPU, entonces el tipo de recurso tiene dos instancias, las cuales pueden ser asignadas a satisfacer cualquier solicitud.

Un proceso debe solicitar cada recurso antes de utilizarlo y debe liberarlo después de haberlo usado, el número de recursos solicitados depende de los que necesite para llevar a cabo las tareas que tienen asignadas. El modo de operación normal de solicitud de recursos según Silberschatz, Galvin y Gagne (2006) es el siguiente.

1. **Solicitud:** si la solicitud no puede ser concedida inmediatamente, entonces el proceso solicitante tendrá que esperar hasta que pueda adquirir el recurso.
2. **Uso:** el proceso puede operar sobre el recurso.
3. **Liberación:** el proceso libera el recurso.

10.2. Condiciones para que se cumpla el interbloqueo

Según Silberschatz, Galvin y Gagne (2006), para que ocurra un interbloqueo, se deben cumplir las siguientes condiciones.

- **Exclusión mutua:** al menos un recurso debe estar en modo no compartido; es decir, solo un proceso puede usarlo cada vez. Si otro proceso solicita el recurso, debe esperar hasta que el recurso sea liberado.

- **Retención y espera:** un proceso debe estar reteniendo al menos un recurso y esperando para adquirir otros recursos adicionales que actualmente estén retenidos por otros procesos.
- **Sin desalojo:** los recursos no pueden ser desalojados; es decir; un recurso solo puede ser liberado voluntariamente por el proceso que le retiene, después de que dicho proceso haya completado su tarea.
- **Espera circular:** debe existir un conjunto de procesos $\{P_0, P_1, \dots, P_n\}$ en espera, de tal forma que P_0 está esperando por un recurso retenido por P_1 , P_1 está esperando a un recurso retenido por P_2, \dots, P_{n-1} está esperando un recurso que fue por P_0 . Por lo cual existen una cadena de dos o más procesos de manera tal que cada proceso en la cadena tiene un recurso solicitado por el próximo proceso en la misma.

Para que se produzca un interbloqueo se deben producir las cuatro condiciones: exclusión mutua, retención y espera, sin desalojo y espera circular.

10.3. Grafos de asignación de recursos

El grafo de asignación de recursos es un grafo dirigido, conformado por un conjunto de vértices (V) y un conjunto de aristas (E).

Los vértices están formados por dos tipos de nodos que son los procesos (P) y los recursos del sistema (R).

En el caso de las aristas tienen el siguiente significado:

- **Arista de solicitud:** si la arista está dirigida desde el proceso al recurso significa que el proceso ha solicitado una instancia del recurso y que actualmente está esperando el recurso. Ejemplo: $p_i \rightarrow R_j$.

- **Arista de asignación:** si la arista está dirigida desde el recurso hasta el proceso, significa que se ha asignado una instancia del recurso al proceso. Ejemplo: $r_j \rightarrow P$

En la figura 82, se representan las aristas:

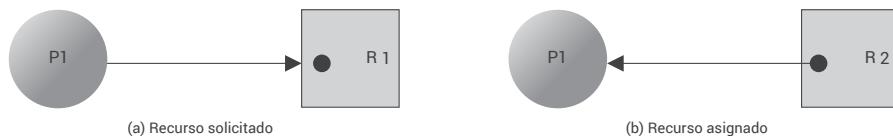


Figura 82. Aristas de un grafo de asignación de recursos.

Fuente: XXXXX

En la figura 83, se representa un ejemplo de interbloqueos, en el cual solo hay una unidad del recurso R1 y R2, en donde el proceso P1 mantiene a R2 y solicita R1; mientras que el proceso P2 mantiene al recurso R1 y pide R2.

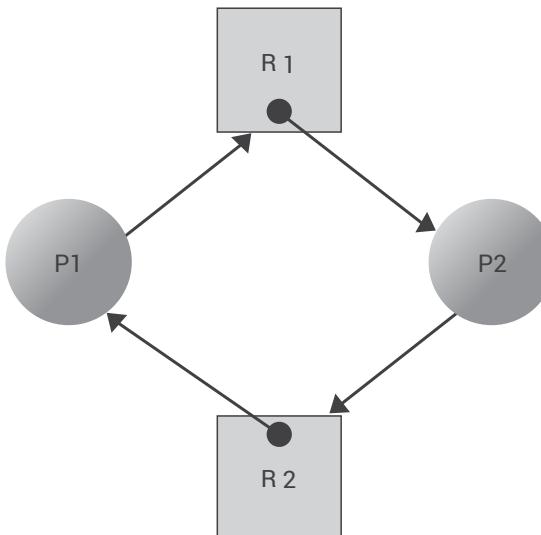


Figura 83. Ejemplo de interbloqueo de procesos.

Fuente: XXXXX

En la figura 84 se presenta un ejemplo donde el recurso R1 tiene tres instancias y R2 tiene dos instancias; por lo cual no se produce un interbloqueo.

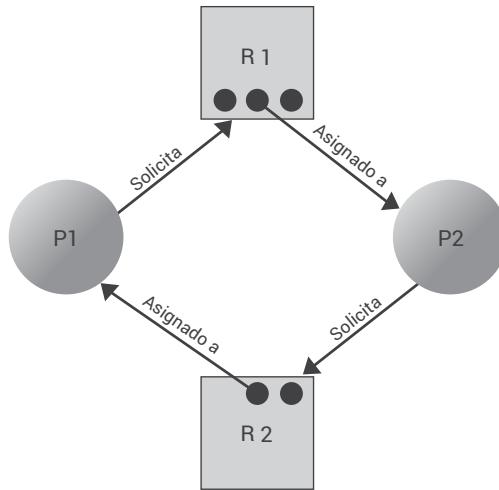


Figura 84. Grafo de asignación de recursos sin Interbloqueo

Fuente: XXXXX

Para determinar si existe un interbloqueo es necesario verificar que se cumplan las cuatro condiciones mencionadas en la sección 10.2.

Al respecto Stallings (2005), indica lineamientos para diferenciar cuando existe interbloqueo, se los detalla en la tabla 9.

Tabla 9. Lineamientos para diferenciar interbloqueos

Posibilidad de Interbloqueo	Existencia de interbloqueo
<ul style="list-style-type: none"> ▪ Exclusión mutua. ▪ Sin expropiación ▪ Retención y espera 	<ul style="list-style-type: none"> ▪ Exclusión mutua ▪ Sin expropiación ▪ Retención y espera ▪ Espera circular

Fuente: Stallings (2005)

Para tratar los interbloqueos existen tres estrategias: prevenir, predecir y detectar; los invitamos a revisar cada una de ellas en las siguientes secciones.

10.4. Prevención de interbloqueos

La estrategia de prevención de interbloqueo consiste en no permitir que ocurra alguna de las cuatro condiciones mencionadas en la sección 10.2.

Stallings (2005), indica que los métodos de prevención de interbloqueos se pueden clasificar en:

- **Directos:** impide que se produzca una espera circular (cuarta condición).
- **Indirectos:** impide la aparición de una de las tres condiciones: exclusión mutua, sin expropiación y retención y espera.

A continuación, se examinan las técnicas relacionadas con las cuatro condiciones:

- **Exclusión mutua:** esta condición no se puede eliminar, debido a que si un recurso requiere exclusión mutua el sistema operativo lo debe proporcionar.
- **Retención y espera:** puede eliminarse estableciendo que un proceso debe solicitar al mismo tiempo todos sus recursos requeridos, bloqueándolo hasta que le puedan asignar simultáneamente todas las peticiones.
- **Sin expropiación:** esta condición se puede impedir, si a un proceso que mantiene varios recursos se le deniega una petición posterior, para lo cual el proceso debe liberar sus recursos originales y si es necesario los debe solicitar nuevamente junto con el recurso adicional.

- **Espera circular:** se puede impedir que se produzca definiendo un orden lineal entre los distintos tipos de recursos.

10.5. Predicción de interbloqueos

Consiste en proporcionar de antemano al sistema operativo información sobre los recursos que solicitará el proceso durante su tiempo de ejecución. Con estos datos adicionales, se puede decidir para cada solicitud si el proceso tiene que esperar o no. Para decidir si la solicitud actual puede satisfacerse o debe retardarse, el sistema necesita considerar qué recursos hay disponibles en ese momento, qué recursos están asignados a cada proceso y las futuras solicitudes y liberaciones de cada proceso (Silberschatz, Galvin y Gagne, 2013).

Los diferentes algoritmos que usan este método difieren en la cantidad y el tipo de información requerida; siendo el más simple y útil el que declara el máximo número de recursos de cada tipo que puede necesitar.

Los algoritmos más utilizados son: el algoritmo de estado seguro, el grafo de asignación de recursos y el algoritmo de banquero. Le invitamos a que revise material complementario en el EVA y en bases de datos científicas sobre las características de estos algoritmos.

10.6. Detección y Recuperación de Interbloqueos

Cuando se produce un interbloqueo, el sistema debe proporcionar:

- Un algoritmo que examine el estado del sistema para determinar si se ha producido un interbloqueo.
- Un algoritmo para recuperarse del interbloqueo.

Una de las desventajas de estos algoritmos son el coste significativo asociado que no solo incluye el tiempo de ejecución, sino también el coste asociado con el mantenimiento de la información necesaria y la ejecución del algoritmo de detección, sino también las potenciales pérdidas inherentes al proceso de recuperación de interbloqueo (Silberschatz, Galvin y Gagne, 2013).

El algoritmo de detección se debe utilizar de acuerdo con:

- Frecuencia con la que se producirá probablemente un interbloqueo.
- Cantidad de procesos que se verían afectados por el interbloqueo cuando se produzca.

Según Silva (2015), se tienen tres enfoques para la recuperación a partir de un interbloqueo.

- **Desalojo automático:** el sistema operativo expropia un subconjunto de los recursos asignados, considerando criterios como qué recursos seleccionar, la consecuencia y la inanición de estos procesos. Silberschatz, Galvin, y Gagne, (2006), menciona otros factores que influyen en qué proceso seleccionar entre los que se consideran:
 - La prioridad del proceso.
 - Durante cuánto tiempo ha estado el proceso ejecutándose y cuánto tiempo adicional necesita el proceso para completar sus tareas.
 - Cuántos y qué tipos de recursos ha utilizado el proceso.
 - Cuántos más recursos necesita el proceso para completarse.
 - Cuántos procesos hará falta terminar.
 - Si se trata de un proceso interactivo o de procesamiento por lotes.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

- **Finalización automática:** elimina el interbloqueo mediante la finalización de los mismos, se pueden seleccionar todos los procesos bloqueados o solo un subconjunto de ellos.
- **Intervención manual:** se delega la responsabilidad de resolver el problema al operador del sistema.

En la tabla 10, se resumen las estrategias de interbloqueos de sistemas operativos realizada por Stallings (2005).

Tabla 10. *Resumen de las estrategias de interbloqueos de sistemas operativos*

Estrategia	Política de reserva de recursos	Esquemas Alternativos	Principales ventajas	Principales desventajas
Prevención	Conservador, infrautilizar los recursos	Solicitud simultánea de todos los recursos	<ul style="list-style-type: none">• Adecuada para procesos que realizan una sola ráfaga de actividad.• No es necesaria la expropiación	<ul style="list-style-type: none">• Ineficiente• Retrasa la iniciación del proceso.• Los procesos deben conocer sus futuros requisitos de recursos.
		Expropiación	<ul style="list-style-type: none">• Conveniente cuando se aplica a recursos cuyo estado se puede guardar y restaurar fácilmente	<ul style="list-style-type: none">• Expropia con más frecuencia de lo necesario
		Ordenamiento de recursos	<ul style="list-style-type: none">• Es posible asegurarlo mediante comprobaciones en tiempo de compilación.• No necesita cálculos en tiempo de ejecución ya que el problema se resuelve en el diseño del sistema	<ul style="list-style-type: none">• Impide solicitudes graduales de recursos

Estrategia	Política de reserva de recursos	Esquemas Alternativos	Principales ventajas	Principales desventajas
Predicción	A medio camino entre la detección y la prevención	Asegura que existe al menos un camino seguro	<ul style="list-style-type: none"> • No es necesaria la expropiación 	<ul style="list-style-type: none"> • El SO debe conocer los futuros requisitos de recursos de los procesos. • Los procesos se pueden bloquear durante largos períodos
Detección	Muy liberal; los recursos solicitados se conceden en caso de que sea posible.	Se invoca periódicamente para comprobar si hay interbloqueo	<ul style="list-style-type: none"> • Nunca retrasa la iniciación del proceso. • Facilita la gestión en línea 	<ul style="list-style-type: none"> • Pérdidas inherentes por expropiación

Fuente. (Stallings, 2005)



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Analice los métodos para tratar interbloqueos.

- **Procedimiento**

Como estrategia de desarrollo se recomienda utilizar como base los siguientes recursos:

Silberschatz, A., Galvin, A., y Gagne, G. (2006). *Fundamentos de sistemas operativos*. España: McGraw-Hill Interamericana.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Silva, M. (2015). *Sistemas Operativos*. Buenos Aires, Argentina: Alfaomega.

Stallings, W. (2005). *Sistemas operativos Aspectos internos y principios de diseños* Madrid, España: Pearson Prentice Hall.

Tanenbaum A., (2009). *Sistemas Operativos Modernos*. México: Pearson Prentice Hall.

Actividad 2:

- **Actividad de aprendizaje**

Autoevaluación.

- **Procedimiento**

Analice las preguntas propuestas en la autoevaluación 10, si tiene dudas, puede revisar la retroalimentación de la misma al final de esta guía virtual.



Autoevaluación 10

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. La exclusión mutua en Interbloqueos se refiere a:
 - a. Un recurso sólo puede ser liberado voluntariamente por el proceso que le retiene, después de que dicho proceso haya completado la tarea.
 - b. Al menos un recurso debe estar en modo no compartido.
 - c. Un proceso debe estar reteniendo al menos un recurso y esperando para adquirir otros recursos adicionales que actualmente estén retenidos por otros procesos
2. Una situación de interbloqueo puede surgir si se dan simultáneamente las siguientes condiciones:
 - a. Exclusión mutua, retención y espera, sin desalojo.
 - b. Exclusión mutua, retención y espera, espera circular.
 - c. Exclusión mutua, retención y espera, sin desalojo, espera circular
3. El grafo de asignación de recursos del sistema
 - a. Está formado por un conjunto de vértices V y de un conjunto de aristas E.
 - b. Está formado solo por el conjunto de procesos activos del sistema.
 - c. Está formado sólo por el conjunto de procesos inactivos del sistema.

Índice

Primer bimestre

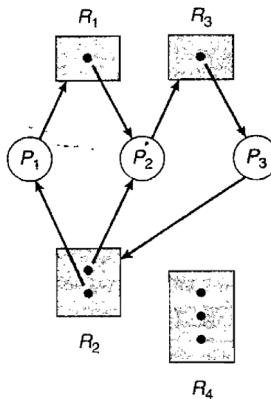
Segundo bimestre

Solucionario

Referencias bibliográficas

4. Una condición no necesaria para la aparición del interbloqueo es:
 - a. La existencia de recursos con número finito de instancias.
 - b. La existencia de recursos que exigen exclusión mutua.
 - c. La aparición de un estado inseguro.
5. Una de las principales desventajas de la estrategia de prevención es:
 - a. Los procesos deben conocer sus futuros requisitos de recursos.
 - b. Hay pérdidas inherentes por expropiación.
 - c. Los procesos se pueden bloquear durante largos periodos.
6. Una de las principales desventajas de la estrategia de detección es:
 - a. Los procesos deben conocer sus futuros requisitos de recursos.
 - b. Hay pérdidas inherentes por expropiación.
 - c. Los procesos se pueden bloquear durante largos periodos.
7. Una de las principales desventajas de las técnicas de prevención de interbloqueos consiste:
 - a. Los procesos deben conocer sus futuros requisitos de recursos.
 - b. Impide solicitudes graduales de recursos.
 - c. Pérdidas inherentes por expropiación.

8. Los grafos de asignación de recursos son una técnica de:
- Predicción de interbloqueos.
 - Prevención de interbloqueos.
 - Detección de interbloqueos
9. El estado de interbloqueos se produce cuando se dan simultáneamente las siguientes condiciones:
- Exclusión mutua, retención y espera, existencia de expropiación y espera circular.
 - Sección crítica, retención y espera, existencia de expropiación y espera circular.
 - Exclusión mutua, retención y espera, no existencia de expropiación y espera circular.
10. Analice el siguiente grafo de asignación de recursos:



- Corresponde a un grafo de asignación de recursos con un interbloqueo.
- Corresponde a un grafo de asignación de recursos sin interbloqueo.
- Corresponde a un grafo de asignación de recursos con solo una instancia.

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Resultado de aprendizaje 19

Compara y contrasta las estrategias de gestión de memoria de diferentes sistemas operativos.

Contenidos, recursos y actividades recomendadas



Semana 13



Unidad 11. Gestión de memoria

11.1. Antecedentes

En un sistema computacional para que se puedan ejecutar programas junto con los datos a los que acceden deberán estar ubicados en memoria principal. A esto hay que añadir la característica de multiprogramación que mantienen muchos sistemas con la finalidad de mejorar el aprovechamiento de la CPU de la computadora que deberá mantener varios procesos en la memoria; es decir, varios procesos deberán compartir esta memoria física.

La gestión de la memoria es la encargada de asignar espacios de memoria a los procesos que así lo requieren para su ejecución; este proceso es sumamente importante, pues de la efectividad de esta asignación dependerá mucho el rendimiento de todo el sistema, ya que mantiene una influencia e interacción directa con el planificador, debido a que este realiza su trabajo con base en la utilización del espacio de memoria física. Es decir, la gestión de memoria es:

- Una tarea llevada a cabo por el sistema operativo y el *hardware* para acomodar múltiples procesos en memoria principal.
- Si solo unos pocos procesos pueden ser mantenidos en memoria, entonces los procesos estarán esperando por E/S y la CPU estará desocupada.
- Luego, la memoria necesita ser alojada eficientemente en orden para mantener la mayor cantidad de procesos activos.
- En muchos esquemas, el *kernel* ocupa algunas porciones fijas de memoria principal y el resto es compartido por múltiples procesos

Por lo antes mencionado las funciones del gestor de memoria según Silberschatz, Galvin, y Gagne (2013) son:

- Llevar el registro de qué partes de la memoria están en uso.
- Decidir qué procesos se van a cargar en la memoria cuando el espacio de la misma esté disponible.
- Asignar y liberar espacio de la memoria según se necesite.

Además, Stallings (2005) analiza los requisitos que la gestión de la memoria debe satisfacer, los cuales se detallan a continuación.

- **Reubicación:** la memoria principal disponible se comparte entre varios procesos, por lo que el sistema operativo busca cargar y descargar los procesos activos en la memoria principal para

maximizar el uso del procesador; por lo cual se mantiene una gran cantidad de procesos listos para ejecutarse. Una vez que se descargó el programa a disco, cuando vuelva a ser cargado se puede necesitar reubicar el proceso en un área distinta de memoria (Stallings, 2005).

Es decir, la reubicación se refiere al hecho de cargar y ejecutar un programa en una posición arbitraria de memoria.

Según Candela, García, Quesada, Santana y Santos (2007), existen dos tipos de reubicación:

Reubicación estática: cuando un proceso que ha sido bajado a memoria auxiliar al ser cargado nuevamente ocupará el mismo lugar en memoria principal donde se ubicaba al inicio; aun cuando el área esté ocupada deberá esperar para poderse colocar. Este esquema no ofrece muchas ventajas.

Reubicación dinámica: cuando el proceso se va a subir en memoria principal puede ser colocado en cualquier partición libre. En este caso es necesario realizar una reasignación de direcciones cada vez que el programa se ejecute. Este esquema resulta costoso por el *hardware* que se utiliza, así como aumenta el tiempo de acceso a memoria.

En la figura 85, se muestra un ejemplo de traducción de direcciones utilizando reubicación dinámica.

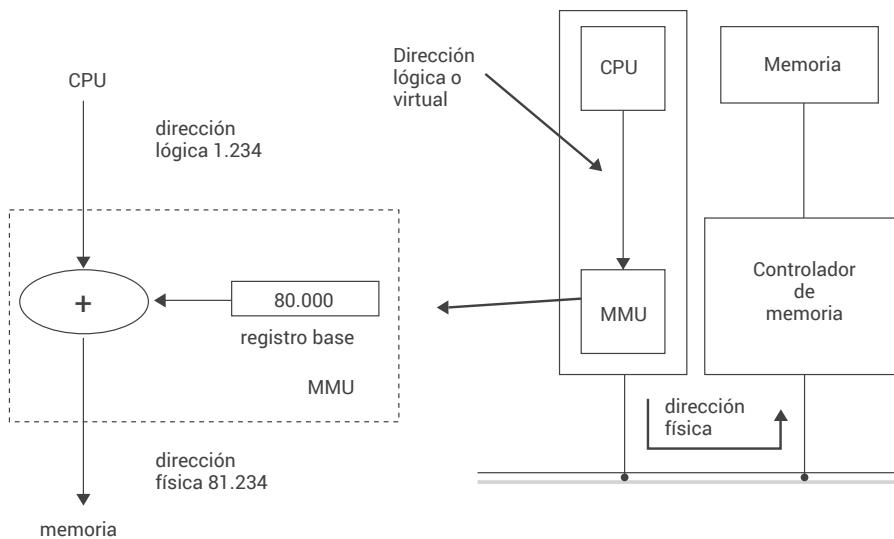


Figura 85. Reubicación Dinámica.

Fuente: Candela, García, Quesada, Santana y Santos (2007)

- **Protección:** cada proceso se debe proteger contra interferencias no deseadas por parte de otros procesos, sean accidentales o intencionadas.

Por lo cual todas las referencias de memoria generadas por un proceso se deben comprobar en tiempo de ejecución para poder asegurar que se refieren solo al espacio de memoria asignado a dicho proceso.

Según Stallings, (2005) es difícil implantar la protección de memoria en tiempo de compilación debido a que no se puede conocer de antemano la dirección física que el proceso ocupará en tiempo de ejecución.

Es el procesador el que debe satisfacer las exigencias de protección de memoria, ya que el SO no puede anticiparse a todas las referencias de memoria que hará el programa.

Al respecto La Red, (2001) menciona que el sistema operativo debe estar protegido contra el proceso de usuario a través del uso de registro de límites.

La protección de memoria se consigue utilizando los registros base y registro límite, que delimitan la zona de memoria en la que el usuario está autorizado a trabajar; el sistema está dotado de un *hardware* específico para implementar estos registros y el mecanismo de protección, el cual se activa cuanto está en modo usuario para verificar que la dirección de memoria es legal (Candela, García, Quesada, Santana y Santos, 2007).

Para implementar la política de protección, se utiliza la Unidad de Manejo de Memoria (MMU) con un registro base y un registro límite.

En la figura 86, se muestra el esquema de protección de memoria.

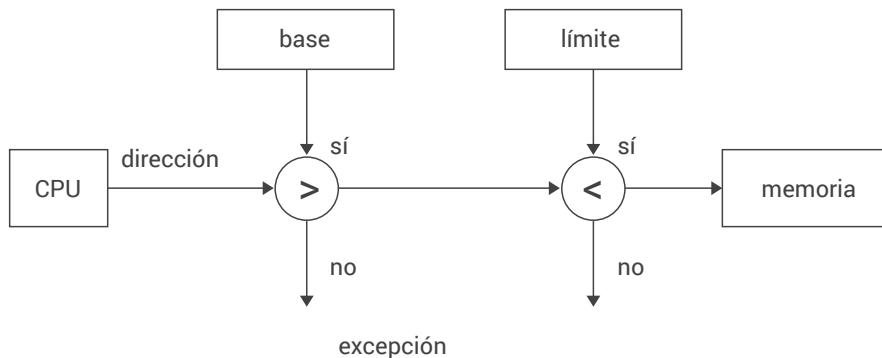


Figura 86. Protección de acceso a memoria.

Fuente: Candela, García, Quesada, Santana y Santos (2007)

- **Compartición:** cuando hay procesos que están cooperando en la misma tarea pueden necesitar compartir el acceso a la misma estructura de datos; por lo cual el sistema de gestión de la memoria debe permitir el acceso controlado a áreas de memoria compartidas sin comprometer la protección esencial (Stallings, 2005).

- **Organización lógica:** es el espacio de almacenamiento lineal o unidimensional, compuesto por bytes o palabras.

Según Silberschatz, Galvin y Gagne (2006), la dirección generada por la CPU se denomina dirección lógica, también se la puede conocer como dirección virtual.

El conjunto de todas las direcciones lógicas generadas por un programa es lo que se denomina un espacio de direcciones lógicas.

Candela, García, Quesada, Santana y Santos (2007), mencionan que las direcciones lógicas son aquellas a las que accede el programa, tal y como se generó el archivo ejecutable.

- **Organización física:** las direcciones físicas son aquellas donde se han ubicado realmente el programa en memoria principal. A cada dirección lógica le corresponderá una dirección física que no necesariamente tiene que coincidir (Candela, García, Quesada, Santana y Santos, 2007).

11.2. Técnicas de gestión de memoria

En esta sección se analizan las técnicas de gestión de memoria: partición estática, partición dinámica, paginación, segmentación y memoria virtual. Les invitamos a revisar con detenimiento cada una de ellas.

11.2.1. Partición estática

La memoria se divide en dos partes: sistema operativo y programa de usuario. La forma más simple es considerar que solo se cargará un proceso en la memoria; en la figura 87 se muestra una partición simple de memoria.

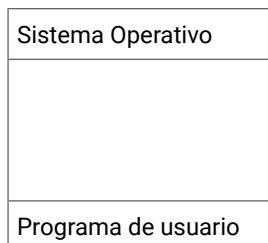


Figura 87. Memoria con una sola partición.

Fuente: elaboración propia.

Una de las funciones del gestor de memoria es conocer las áreas que están libres y ocupadas y cuál es el proceso que está ocupando la zona en particular. Estas zonas se conocen como particiones de memoria, las cuales pueden ser fijas o variables.

En la gestión de memoria, se puede asumir que el sistema operativo ocupa una porción fija de memoria principal y que el resto de la memoria está disponible para múltiples procesos; siendo el esquema más simple repartirla en regiones o particiones con límites fijos (Stallings, 2005).

La memoria se divide en un conjunto de particiones de tamaños preestablecidos, en cada una de las particiones se ubica un único programa. Según Stallings (2005), menciona que una dirección física o dirección absoluta, es una posición real en la memoria principal.

El particionamiento fijo de memoria según La Red (2001), se caracteriza por:

- Las particiones son de tamaño fijo.
- Alojan a un proceso en cada partición.
- La CPU se cambia rápidamente entre los procesos creando la ilusión de simultaneidad.

Entre una de las ventajas de la partición fija se puede mencionar que requieren un mínimo soporte por parte del sistema operativo por ser un esquema relativamente sencillo.

Según Stallings (2005), las principales desventajas del uso de las particiones fijas del mismo tamaño son:

- Los procesos pequeños no utilizan eficientemente el espacio de las particiones.
- La utilización de la memoria principal es extremadamente ineficiente, cualquier programa sin importar lo pequeño que sea, ocupa una partición entera.
- El número de particiones especificadas en tiempo de generación del sistema limita el número de procesos activos del sistema.

11.2.2. Partición dinámica

En un sistema de particiones variables, inicialmente toda la memoria se encuentra libre, pero cuando un programa se empieza a ejecuta, se le concede un bloque de memoria y cuando este termine el bloque pasa a la zona de bloques libres, lo que en muchas ocasiones genera que hayan huecos libres de memoria.

Los procesos ocupan tanto espacio como necesitan, pero no deben superar el espacio disponible de memoria, no hay límites fijos de memoria, es decir que la partición de un trabajo es su propio tamaño.

La fragmentación de memoria ocurre en todos los sistemas independientemente de su organización de memoria.

Según La Red, (2001), la fragmentación en sistemas operativos de partición fija se produce cuando:

- Los trabajos de usuario no llenan completamente sus particiones designadas.
- Una partición permanece sin usar porque es demasiado pequeña para alojar un trabajo de espera.

La memoria principal puede sufrir dos tipos de fragmentación:

- **Fragmentación interna:** es la memoria que se desperdicia a causa de las unidades de asignación de tamaño fijo.

Por ejemplo: una memoria de 32 megabytes cuyas particiones tienen una longitud de 8 megabytes y hay un proceso que tiene una longitud de 2 Megabytes cuando se lleva a la memoria ocuparía una de las particiones de 8 Megabytes, con lo cual se está desperdiando memoria.

- **Fragmentación externa:** es la memoria que no se puede asignar por no estar contigua.

Según Stallings (2005), la fragmentación externa es el fenómeno en el cual existen muchos huecos pequeños en la memoria.

La fragmentación externa se la puede manejar utilizando la técnica de compactación.

La compactación es la combinación de los huecos en la memoria, a través del desplazamiento de los mismos a la zona más baja de la memoria.

11.2.2.1. Algoritmos de ubicación

En el esquema de particiones dinámicas se pueden considerar los algoritmos de primer, mejor y peor ajuste, cuyo propósito es seleccionar entre los bloques de memoria libres para cargar un proceso. A continuación, se detallan cada uno de ellos:

1. **Algoritmo de primer ajuste:** el sistema operativo revisa en todas las secciones de memoria libre. El proceso es asignado al primer hueco encontrado que sea mayor que el tamaño del proceso. A menos que el tamaño del proceso coincida con el tamaño del hueco, el agujero continúa existiendo, reducido por el tamaño del proceso (Silva, 2015).

Por lo general la exploración del agujero empieza al inicio del conjunto de agujeros y se puede detener en cuanto se encuentre un agujero libre que sea lo suficientemente grande como para alojar al proceso.

1. **Algoritmo de mejor ajuste:** según Silberschatz, Galvin y Gagne (2013) consiste en asignar el agujero más pequeño que tenga el tamaño suficiente, para lo cual se debe explorar la lista completa de agujeros.
2. **Algoritmo de peor ajuste:** este algoritmo consiste en asignar el agujero de mayor tamaño, para lo cual se debe explorar toda la lista de agujeros.

Según Silberschatz, Galvin y Gagne, (2013), no está claro cuál de las estrategias de primer y mejor ajuste, es mejor en términos de utilización del espacio de almacenamiento, pero la estrategia de primer ajuste es generalmente más fácil de implementar.

Considere un proceso $P_n = 70 \text{ Kb}$, que se debe ubicar en el mapa de memoria de la figura 88.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

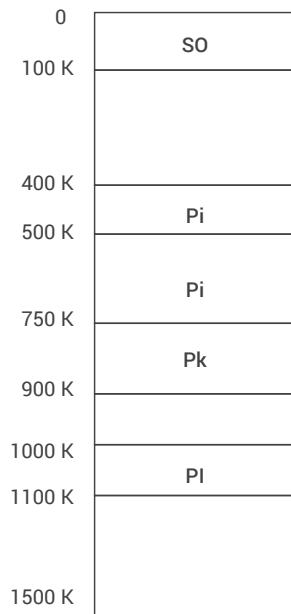


Figura 88. Mapa de memoria.

Fuente: XXXX

Aplicando los algoritmos de primer, mejor y peor ajuste para ubicar el Pn, tiene como resultado los siguientes mapas de memoria representados en la figura 89.

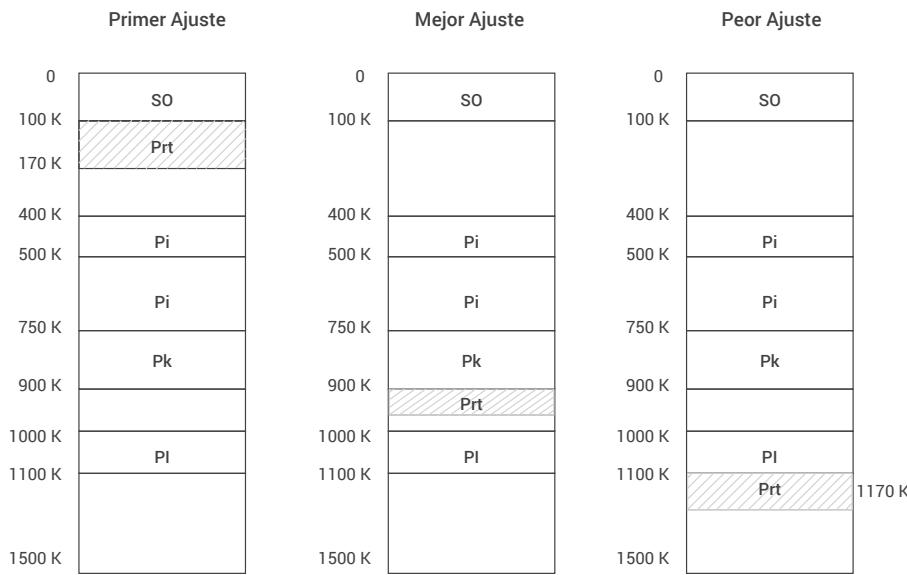


Figura 89. Aplicación de los algoritmos de primer, mejor y peor ajuste.

Fuente: XXXX

Le invitamos a que realice ejercicios similares para que comprenda el funcionamiento de los mismos.

11.2.3. Paginación

La paginación es un esquema de gestión de memoria que permite que el espacio de direcciones físicas de un proceso sea no contiguo, evita el problema de encajar fragmentos de memoria de tamaño variable en el almacén de respaldo (Silberschatz, Galvin y Gagne, 2013).

La memoria física se divide en porciones o particiones de tamaño fijos llamados marcos (frames) y la memoria lógica en bloques del mismo tamaño llamados páginas. Para lo cual el sistema operativo cuenta con una tabla de páginas (TP), donde se relaciona cada página con el marco de página correspondiente.

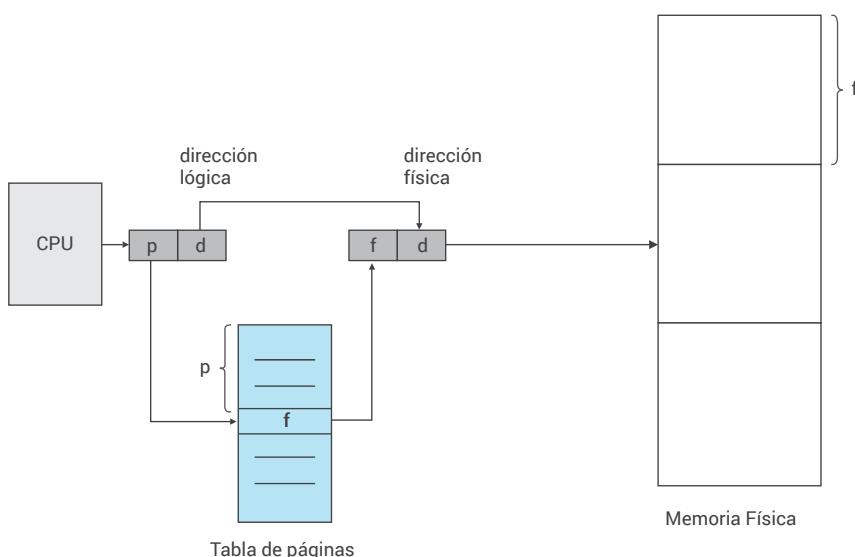


Figura 90. Hardware de Paginación.

Fuente: Silberschatz, Galvin y Gagne (2013)

Cabe resaltar que el tamaño de las páginas son potencias de dos y, por ende, el tamaño del marco de página también.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Para Silberschatz, Galvin y Gagne (2013), si el tamaño del espacio de direcciones lógicas es 2^m y el tamaño de página es 2^n unidades de direccionamiento, entonces los $m-n$ bits de mayor peso de cada dirección lógica designarán el número de página, mientras que los n bits de menor peso indicarán el desplazamiento de página por lo tanto la dirección lógica tiene la estructura de la figura 91.

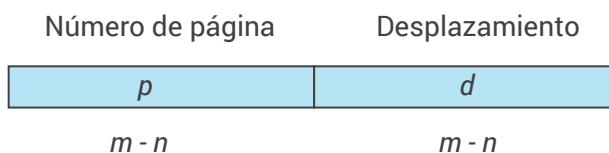


Figura 91. Estructura de la dirección lógica de paginación.

Fuente: Silberschatz, Galvin y Gagne (2013)

Silberschatz, Galvin y Gagne (2013), muestran un ejemplo en donde se utiliza una memoria física de 32 bytes; el tamaño de página es de 4 bytes; por lo tanto la memoria física se encuentra dividida en 8 páginas. Como se puede observar la dirección lógica 0 representa la página 0, desplazamiento 0; al realizar la indexación en la tabla de páginas se puede observar que la página 0 se encuentra en el marco de página 5; por lo cual la dirección física de la página 0 es 20 que se obtiene de la siguiente manera:

$$\text{Página (p)} = 0$$

$$\text{Tamaño de Página (tp)} = 4$$

$$\text{Marco de Página (mp)} = 5$$

$$\text{Desplazamiento (d)} = 0$$

$$\text{De donde: dirección Física} = ((\text{mp} * \text{tp}) + \text{d}) = ((5 * 4) + 0)$$

Este proceso se debe realizar con todas las páginas, para obtener las direcciones físicas. Les invitamos a que realicen este mismo ejercicio utilizando la página 9, página 15, etc. Puede revisar los resultados en la figura 92.

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

0	5
1	6
2	1
3	2

Tabla de páginas

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

Memoria Física

Figura 92. Ejemplo de paginación.

Fuente: Silberschatz, Galvin y Gagne (2013)

Con este esquema de paginación, se evita la fragmentación externa, pero sí se puede tener un cierto grado de fragmentación interna.

A continuación se detallará el esquema de segmentación.

11.2.4. Segmentación

Este esquema al igual que el de paginación tiene el objetivo de dividir el programa en porciones más pequeñas para que sea más fácil ubicarlo en memoria, pero, la diferencia es que es el usuario quien divide el programa en bloques contiguos, a los cuales se les denomina segmentos.

Al utilizar segmentos de distinto tamaño se tiene como consecuencia que no hay relación simple entre direcciones lógicas y direcciones físicas; cada entrada también debería proporcionar la longitud del segmento para asegurar que no se utilizan direcciones no válidas (Stallings, 2005).

Las direcciones lógicas en segmentación constituyen los segmentos y cada segmento tiene un nombre y una longitud, las direcciones especifican tanto el nombre del segmento como el desplazamiento dentro de ese segmento (Silberschatz, Galvin y Gagne, 2013).

Silberschatz, Galvin y Gagne (2013) mencionan que para la traducción de direcciones se utiliza un *hardware* especial, para la traducción de direcciones lógicas a física se utiliza una tabla de segmentos para cada proceso y una lista de bloques libres en la memoria principal; en la tabla de segmentos cada entrada contiene la dirección de comienzo del segmento correspondiente de la memoria principal, en la figura 93 se representa el *hardware* de segmentación.

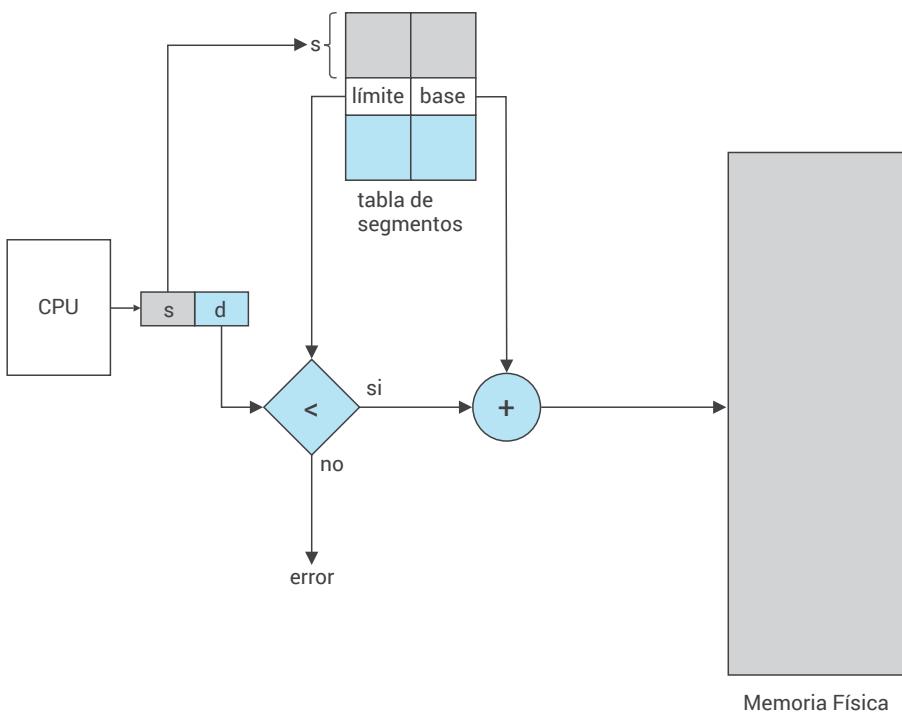


Figura 93. Hardware de segmentación.

Fuente: Silberschatz, Galvin y Gagne (2013)

El mapeo de las direcciones se lleva a cabo mediante una tabla de segmentos, la cual está formada por una dirección base del segmento y un límite del segmento; la dirección base contiene la dirección física inicial del lugar donde el segmento reside dentro de la memoria, mientras que el límite del segmento especifica la longitud del mismo.

En la figura 94, se puede observar que hay 5 segmentos, identificados desde el 0 al 4; a manera de ejemplo considere el segmento 2 que tiene una longitud de 400 bytes y comienza su ubicación en la dirección 4300; por lo cual una referencia al byte 53 del segmento 2 corresponde a la posición 3200 (la base del segmento 3) + 852 = 4052.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

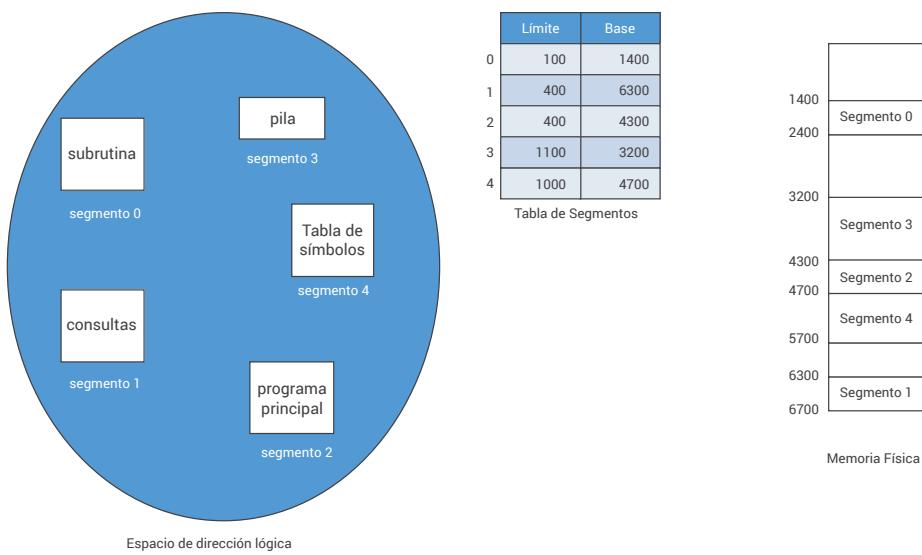


Figura 94. Ejemplo de segmentación.

Fuente: Silberschatz, Galvin y Gagne (2013)

La segmentación elimina la fragmentación interna pero, al igual que el particionamiento dinámico, sufre de fragmentación externa (Stallings, 2005).

En la siguiente sección se analizarán los principios de memoria virtual.

11.2.5. Memoria virtual

El esquema de memoria virtual tiene como objetivo eliminar la restricción del tamaño de los programas que se tiene respecto a las limitaciones de la memoria principal; es decir, permite que los procesos estén parcialmente cargados en memoria lo que permitirá trabajar con memoria de mayor capacidad que la que realmente se tiene.

Según Candela, García, Quesada, Santana y Santos (2007), el fundamento de la memoria virtual no es más que aplicar el principio

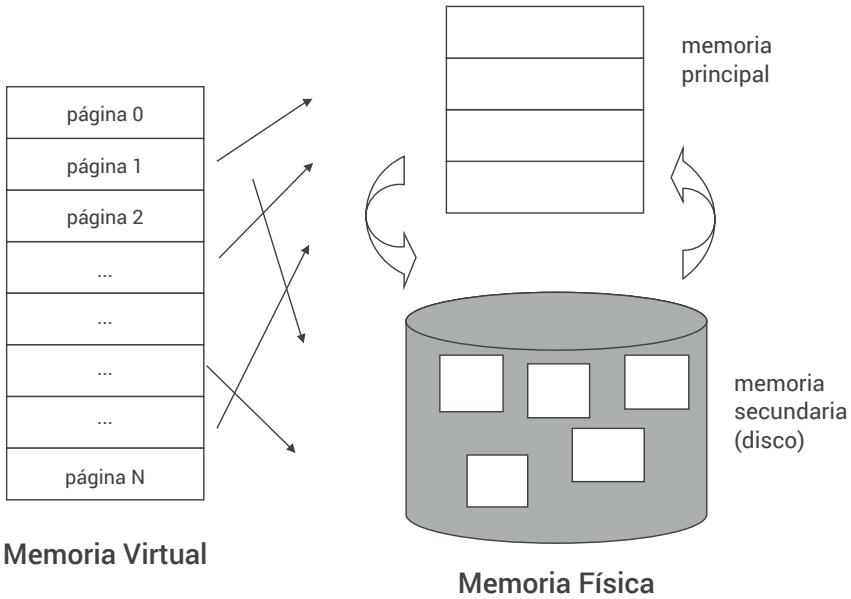


Figura 95. Organización de memoria virtual.

Fuente: Candela, García, Quesada, Santana y Santos (2007)

El esquema de memoria virtual surge de la necesidad de ejecutar programas que son demasiados grandes como para caber en la memoria y sin duda también existe la necesidad de tener sistemas que puedan soportar varios programas ejecutándose simultáneamente (Stallings, 2005).

Además, el esquema consiste en que cada programa tiene su propio espacio de direcciones, el cual se divide en trozos llamados páginas, cada página es un rango continuo de direcciones, estas páginas se asocian a la memoria física, pero no todas deben estar en la memoria física al mismo tiempo para ejecutarse; a este esquema se lo conoce como paginación por demanda.

11.2.5.1. Algoritmos de reemplazo de páginas

El sistema operativo es el encargado de seleccionar una página para desalojarla de memoria principal y hacer un espacio para la página entrante cuando ocurre un fallo de página, es decir, cuando se necesita que una página esté cargada en memoria y esta no se encuentra ahí.

La gestión de memoria virtual necesita estar guiada por ciertas políticas como:

- **Políticas de asignación:** cuánta memoria real se asigna a cada proceso activo.
- **Políticas de búsqueda:** paginación por demanda, prepaginación. Las páginas de procesos pueden introducirse por demanda o introduciendo varias páginas a la vez.
- **Políticas de acceso:** cuáles elementos se incorporan y cuándo se les incorpora desde el almacenamiento secundario a la memoria principal.
- **Políticas de sustitución:** decidir qué elemento se desaloja con el fin de dejar espacio para el nuevo.
- **Políticas de ubicación:** dónde se coloca el nuevo elemento.
- **Control de carga:** nivel de multiprogramación, es decir, el número de procesos que residirán en la memoria principal.

A continuación, se describen los algoritmos de remplazo de páginas más utilizados.

■ Algoritmo de remplazo de página FIFO

Es el más simple de entender y de implementar se basa en la política de la primera página que ingresa es la primera en salir, es decir, asocia cada página al instante en que dicha página fue cargada en memoria.

Considere que tiene la siguiente cadena de referencias de páginas: **a b c d e f g a b c d e f g h a b c d**, en donde los 4 marcos de página se encuentran inicialmente vacíos, las cuatro primeras referencias (a, b, c, d) provocan fallos de página y esas páginas se cargan en los marcos vacío. La siguiente referencia es la **página e**, pero como no hay marcos vacíos y no se encuentra en memoria principal se produce un fallo de página, por lo cual hay que aplicar la política del algoritmo en este caso la primera página en ingresar es la **página a**, a la cual se la baja a memoria secundaria y se carga **la página e**, lo mismo sucede con la **página f** que se carga en lugar de la **página b**. En la figura 96, se muestra la ejecución de paginación por demanda aplicando el algoritmo FIFO.

a	B	c	d	e	f	g	a	b	c	d	e	f	g	h	a	b	c	d
a	a	a	a	e	e	e	e	b	b	b	b	f	f	f	f	f	b	B
b	b	b	b	b	f	f	f	f	c	c	c	c	G	g	g	g	c	c
c	c	c	c	c	g	g	g	g	d	d	d	d	h	h	h	h	d	
				d	d	d	d	a	a	a	a	e	e	e	e	a	a	a

fp=19

Figura 96. Ejemplo de algoritmo FIFO.

Fuente: XXXXX

■ Algoritmo de remplazo de páginas óptimo

Este algoritmo es fácil de describir, pero difícil de implementar; consiste en remplazar la página que no se usará durante más tiempo, es decir, la que más tarde en ser accedida.

La política óptima de selección tomará como remplazo la página para la cual el instante de la siguiente referencia se encuentra más lejos. Se puede ver que para esta política, los resultados son el menor número de posibles fallos de página (Stallings, 2005).

En la figura 97, se muestra la ejecución del algoritmo óptimo de la cadena de referencia: **a b c d e f g a b c d e f g h a b c d**. Las primeras cuatro páginas se cargan en los marcos vacíos, produciendo en cada uno de ellos un fallo de página debido a que no se encuentran cargados en memoria principal. La **página e** no se encuentra en memoria por lo cual se produce un fallo de página; además no hay marcos de página vacíos en este caso se aplica la política de revisar la página que se utilizará en el tiempo más lejano (revisar la cadena de referencia a la derecha), en este caso la página que se utilizará en el tiempo más lejano es la **página d**.

Se necesita cargar en memoria la **página f**, se revisa la cadena de referencias hacia la derecha, determinando que la **página e** es la que se utilizará en el tiempo más lejano y esta es remplazada, para el resto de cadenas se sigue el mismo procedimiento.

a	b	c	d	e	f	g	a	b	c	d	e	f	g	h	a	b	c	d
a	a	a	a	a	a	a				A	a	A		A			c	c
b	b	b	b	b	b	b				B	b	B		B			b	d
	c	c	c	c	c	c				D	e	F		f			f	f
		d	e	f	g					G	g	g		h			h	h

Fp=13

Figura 97. Ejemplo de algoritmo óptimo.

Fuente: XXXXX

- **Algoritmo de remplazo de páginas LRU**

El algoritmo LRU, o el menos recientemente utilizado, se caracteriza por seleccionar la página que no haya sido utilizada durante el periodo de tiempo más largo.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En este algoritmo se considera revisar la cadena de referencias hacia atrás. El principal problema es cómo implementar ese mecanismo de sustitución debido a que puede requerir una considerable asistencia de *hardware* para lo cual se puede utilizar dos técnicas: contadores y pila (Silberschatz, Galvin y Gagne, 2013).

Consideré el ejemplo de la cadena de referencias: **a b c d e f g a b c d**. Al igual que los algoritmos anteriores los primeros marcos están vacíos por lo cual se cargan las cuatro primeras páginas produciendo un fallo de página; para cargar la **página e** se verifica cuál es la página menos recientemente utilizada determinando que es la **página a** por la cual se remplaza, se produce un fallo de página. La siguiente página que se necesita cargar en memoria principal es la **página f**, como no hay marcos de página vacío se remplaza por la página menos recientemente utilizada, en este caso la **página b**, se produce un fallo de página al cargar en la **página f**, para el resto de páginas se prosigue con la misma política. En la figura 98, puede revisar la ejecución completa del algoritmo de remplazo de páginas utilizando el algoritmo LRU.

a	b	c	d	e	f	g	a	b	c	d	e	f	g	h	a	b	c	d
a	a	a	a	E	E	E	E	B	B	B	B	F	F	F	F	B	B	B
b	b	b	B	F	F	F	F	C	C	C	C	G	G	G	G	C	C	C
c	c	C	C	G	G	G	G	D	D	D	D	H	H	H	H	D		
		d	D	D	D	A	A	A	A	E	E	E	E	A	A	A	A	

Fp=19

Figura 98. Ejemplo de algoritmo LRU.

Fuente: XXXXX

Incluso en un sistema de memoria virtual, el sistema operativo tendrá que expulsar de vez en cuando algunos procesos de forma explícita y completa para mejorar el rendimiento.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Realice un cuadro comparativo entre los algoritmos de reemplazo de página.

- **Procedimiento**

Estrategias de desarrollo se recomienda utilizar como base los siguientes recursos:

Candela, S., García, C., Quesada, A., Santana, F., y Santos, J. (2007). Fundamentos de Sistemas Operativos teoría y ejercicios resueltos. Madrid: Thomson.

Silberschatz, A., Galvin, P., y Gagne, G. (2013). Operating Systems Concepts. Estados Unidos: Wiley.

Silva, M. (2015). Sistemas Operativos. Buenos Aires, Argentina: Alfaomega.

Stallings, W. (2005). Sistemas operativos Aspectos internos y principios de diseños. Madrid, España: Pearson Prentice Hall.

Tanenbaum, A. (2009). Sistemas Operativos Modernos. México: Pearson Prentice Hall.



Autoevaluación 11

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. Si al realizar la asignación de memoria se cuenta con suficiente espacio para alojar el proceso pero este no es contiguo se trata de un problema de:
 - a. Fragmentación interna.
 - b. Fragmentación externa.
 - c. Compactación.
2. La técnica de reorganizar los contenidos de la memoria para colocar junta toda la memoria libre en un bloque grande se llama:
 - a. Reubicación.
 - b. Compactación.
 - c. Protección.
3. La dirección lógica es:
 - a. La dirección generada por la CPU.
 - b. La dirección que se carga en el registro de direcciones de memoria.
 - c. La dirección que llega al chip de memoria.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4. Con el esquema de gestión de memoria mediante particiones fijas se produce:

- a. Fragmentación interna.
- b. Fragmentación externa.
- c. No existe fragmentación.

5. Indique cuál es la dirección física de la dirección virtual (0,128), utilizando la siguiente tabla de segmentos:

Nº de segmento	Base	longitud
0	500	300
1	1800	600
2	100	320
3	2634	650
4	900	45

- a. 628
- b. 3192
- c. Error

6. La compactación de memoria es necesaria en un esquema de gestión de memoria:

- a. De particiones fijas.
- b. De particiones dinámicas.
- c. De paginación

7. ¿Cuál es la función del gestor de memoria virtual?

- a. Permitir la ejecución de procesos que están parcialmente en memoria principal.
- b. Permitir la ejecución de procesos que están completamente en memoria secundaria.
- c. Permitir la ejecución de procesos que están completamente en memoria principal.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8. El algoritmo que sustituye la página que no haya sido utilizada durante el periodo más largo de tiempo se conoce como:
- FIFO
 - LRU
 - Óptimo
9. El número de procesos en memoria principal depende de:
- Capacidad física de memoria.
 - Capacidad de la memoria Caché.
 - Tamaño del proceso.
10. ¿Cuántos fallos de página se producen para la siguiente cadena de referencias, si se utiliza 4 marcos de páginas y el algoritmo de sustitución LRU?

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

- 10 fallos de página
- 8 fallos de página.
- 7 fallos de página.

Ir al solucionario

**Resultado de aprendizaje
20 y 21**

- Comparar la organización de los sistemas de archivos.
- Conocer los principales componentes del subsistema de entrada y salida.

Contenidos, recursos y actividades recomendadas**Semana 14****Unidad 12. Sistemas de archivos****12.1. Introducción al sistema de archivos**

Todas las computadoras realizan la función de almacenar y recuperar información; esta información se almacena a través de los archivos.

Un archivo es un conjunto de información relacionada, que está almacenada en memoria secundaria, al cual se puede acceder mediante un nombre.

Para Tanenbaum (2009), los archivos son unidades lógicas de información creada por los procesos.

Un archivo posee atributos que permiten identificarlo con el objetivo de facilitar la realización de operaciones con archivos a través de las llamadas al sistema operativo. Entre los atributos que un archivo posee se pueden mencionar:

- **Nombre:** es el nombre simbólico del archivo, es la única información legible por parte de los usuarios del sistema.
- **Identificador:** es una etiqueta única, permite identificar el archivo en el sistema de archivos, este no es legible por el usuario del sistema.
- **Tipo:** es la información necesaria para los sistemas que soporten diferentes tipos de archivos.
- **Ubicación:** es un puntero a un dispositivo y a la ubicación del archivo dentro del dispositivo.
- **Tamaño:** expresa el tamaño actual del archivo y posiblemente el tamaño compartido.
- **Protección:** determina los permisos de acceso al archivo; estos permisos pueden ser de lectura, escritura y ejecución, para el propietario, el grupo del propietario y para otros usuarios del sistema. A continuación se describen los usuarios del sistema:

Propietario: es el usuario que creó el archivo.

Grupo propietario: un conjunto de usuarios que comparten el archivo y tienen los mismos privilegios sobre el archivo.

Otros usuarios: son el resto de usuarios del sistema.

- **Fecha:** es la información con respecto a la hora y fecha de creación, última modificación y último uso del archivo.

Según Silberschatz, Galvin y Gagne (2013), el gestor de archivos cumple con las siguientes funciones:

- Crear y eliminar archivos.
- Crear y eliminar directorios.
- Dar soporte a primitivas para la manipulación de archivos y directorios.
- Hacer un mapa de los archivos en el almacenamiento secundario.
- Respaldar archivos en medios de almacenamiento estables (no volátiles).

Las operaciones que se pueden realizar con los archivos se detallan a continuación.

- **Crear:** es necesario encontrar espacio para el archivo dentro del sistema de archivos, el archivo se crea sin datos y se establecen los atributos.
- **Abrir:** antes de poder utilizar un archivo, un proceso debe abrirlo a través de la invocación de una llamada al sistema, con lo cual se sube a memoria principal los atributos y la lista de direcciones del archivo.
- **Escribir:** los datos se escriben en el archivo, desde la posición actual.
- **Leer:** se leen los datos desde la posición actual del archivo.
- **Renombrar:** se cambia el nombre del archivo a través de una llamada al sistema.
- **Borrar:** se lo realiza cuando ya es necesario el archivo y se necesita eliminar espacio en disco.

- **Truncar archivos:** cuando es necesario eliminar la información del archivo, pero se mantienen los atributos.

12.2. Métodos de acceso

Existen algunos métodos que permiten el acceso a la información almacenada en un archivo, así tenemos:

- **Acceso secuencial:** resulta un método de acceso sencillo, la información del archivo se procesa en orden, un registro tras otro. Este esquema es muy útil en archivos de cinta.

Este acceso se proporcionaba en los primeros sistemas operativos en donde un proceso podía leer todos los bytes o registros en un archivo en orden, empezando desde el principio. El medio de almacenamiento utilizado era la cinta magnética.

- **Acceso directo:** mediante este acceso se permite el acceso aleatorio, puesto que el archivo se presenta como una secuencia numerada de registros. Estos son muy utilizados para acceder rápidamente a grandes cantidades de información.

Este tipo de acceso es esencial en muchas aplicaciones, como por ejemplo los sistemas de bases de datos.

12.3. Estructura del sistema de archivos

Los archivos se pueden estructurar de varias maneras, Tanenbaum y Andrew (2009), La Red, (2001) mencionan como estructuras de archivos a:

- a. **Sucesión de bytes:** consiste en una serie no estructurada de bytes, posee máxima flexibilidad.

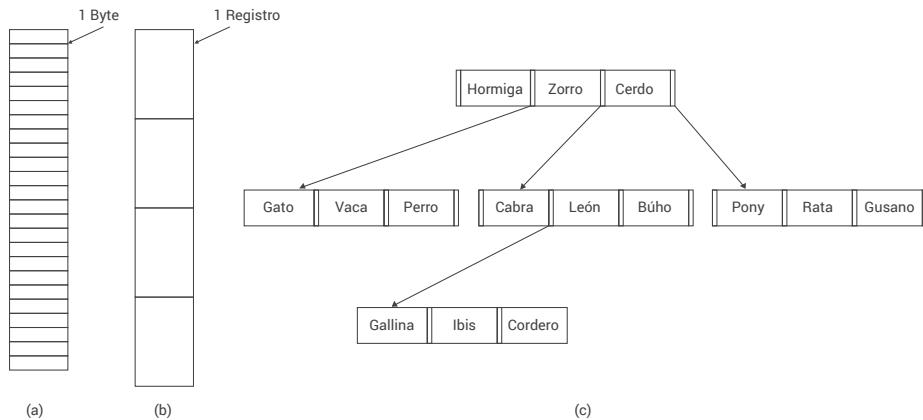


Figura 99. Estructura del sistema de archivos: a) Sucesión de bytes, b) Sucesión de Registros, c) Árbol.

Fuente: Tanenbaum (2009)

Algunos sistemas operativos soportan varios tipos de archivos y por ende de directorios, por lo cual en la siguiente sección se describe la estructura de directorios.

12.4. Estructura de los directorios

Un directorio es un sistema de archivos para mantener la estructura del sistema de archivos (Tanenbaum, 2009).

A continuación se describen los sistemas de directorios que existen.

12.4.1. Sistema de directorios de un solo nivel

La forma más simple de un sistema de directorios es tener un directorio que contenga todos los archivos; se lo conoce como directorio raíz, pero como es el único el nombre no importa (Tanenbaum, 2009).

Silberschatz, Galvin y Gagne (2013), mencionan que los directorios tienen limitaciones significativas, cuando el número de archivos se incrementa o cuando el sistema tiene más de un usuario, por lo tanto, los archivos deberán tener nombres distintos. En la figura 100, se representa un ejemplo de este tipo de directorio.

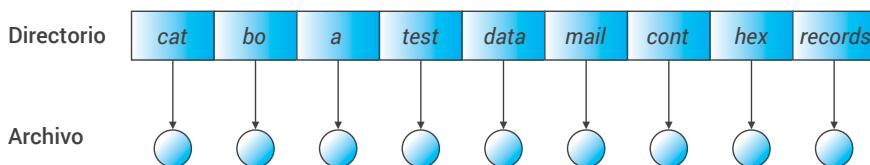


Figura 100. Directorio de único nivel.

Fuente: Silberschatz, Galvin y Gagne (2013)

12.4.2. Sistema de directorios jerárquico

Esta estructura se caracteriza porque cada usuario tiene su propio directorio de archivos, por lo cual cuando un usuario hace referencia a un archivo concreto, solo explora en su propio directorio de archivos. En la figura 101 se muestra un ejemplo de un directorio de dos niveles.

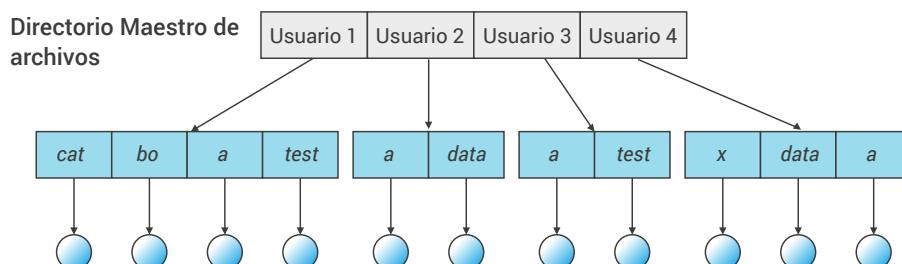


Figura 101. Directorio de dos niveles.

Fuente: Silberschatz, Galvin y Gagne (2013)

12.4.3. Sistema de directorios en árbol

Esta estructura permite a los usuarios crear sus propios subdirectorios y organizar sus archivos correspondientemente, es la estructura de directorio más común.

Según Silberschatz, Galvin y Gagne (2013), para accesar a un archivo se puede hacer a través de dos tipos de rutas: absoluta y relativa.

La ruta absoluta consiste en nombrar desde la raíz y sigue la ruta especificada hasta descender al archivo especificado, mientras que la ruta relativa consiste en definir una ruta a partir del directorio actual. En la figura 102, se muestra un ejemplo de un directorio con estructura de árbol.

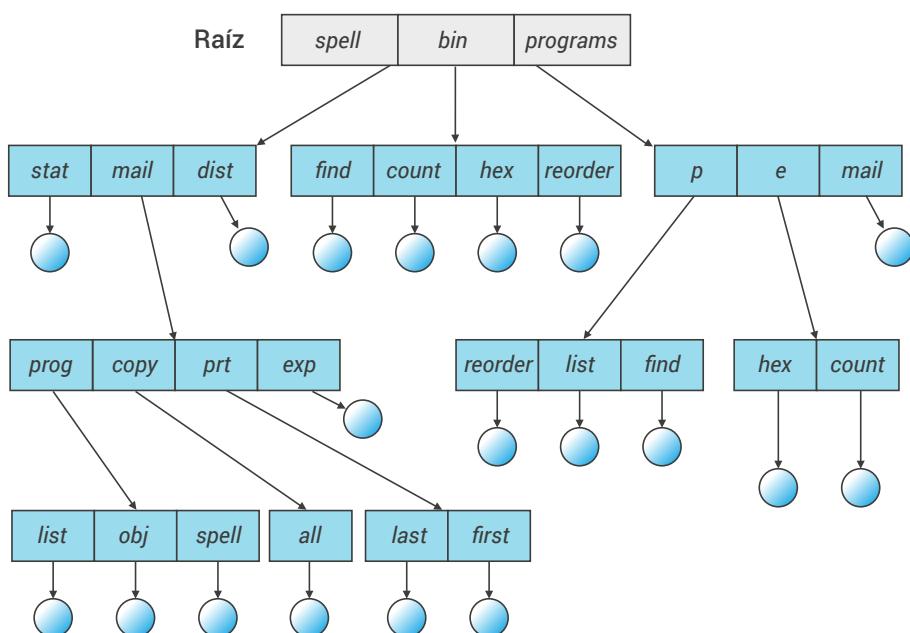


Figura 102. Directorio con estructura de árbol.

Fuente: Silberschatz, Galvin y Gagne (2013)

12.4.4. Sistema de directorios en un gráfico acíclico

La estructura de grafo acíclico permite que los directorios comparten subdirectorios de archivos, el mismo archivo o subdirectorio puede estar en dos directorios diferentes, constituye una generalización del esquema de directorio con estructura de árbol. En la figura 103, se representa una estructura de grafo acíclico.

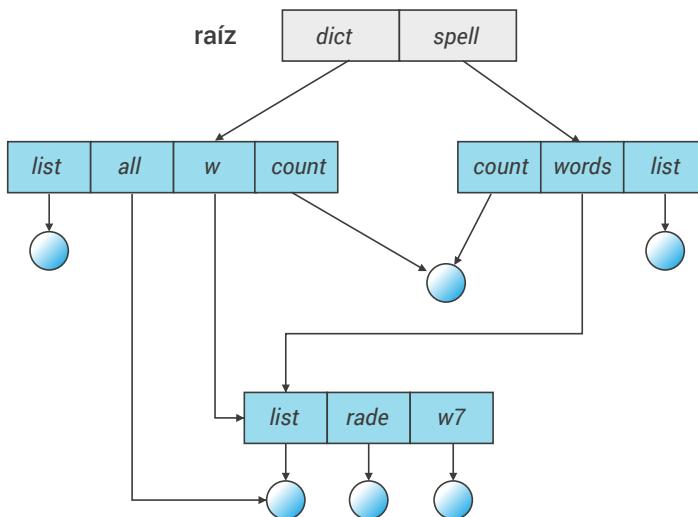


Figura 103. Directorio con estructura de grafo acíclico.

Fuente: Silberschatz, Galvin y Gagne (2013)

12.5. Métodos de asignación

Es muy importante lograr un aprovechamiento eficaz del espacio en disco, que permita un acceso rápido a los archivos. Los métodos de asignación de espacio que son ampliamente usados son: asignación contigua, enlazada e indizada.

A continuación, se describen estos métodos de asignación.

12.5.1. Asignación contigua

La asignación contigua de un archivo está definida por la dirección en disco y la longitud del primer bloque. Para la asignación de espacios libres se trabaja con estrategias de primer ajuste, mejor ajuste y peor ajuste, aunque las más utilizadas son las dos primeras.

Cada fichero ocupa un conjunto de bloques contiguos en el disco. La entrada de directorio para cada fichero tiene la dirección del bloque inicial y la longitud del área asignada al archivo. En la figura 104, se representa la asignación contigua de archivos.

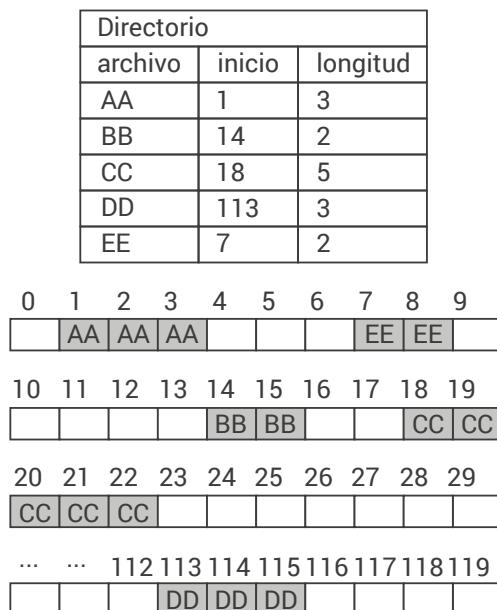


Figura 104. Asignación contigua.

Fuente: Candela, García, Quesada, Santana y Santos (2007)

El método de asignación contigua resulta sencillo, pero presenta inconvenientes al momento de encontrar espacio para un archivo nuevo, ocasionando fragmentación externa e incluso fragmentación interna y, en la mayoría de las ocasiones, es muy difícil determinar cuál será el tamaño del archivo.

12.5.2. Asignación enlazada

La asignación enlazada, consiste en que cada archivo es una lista enlazada de bloques de disco, pudiendo estar dichos bloques dispersos por todo el disco; el directorio contiene un puntero al primer y al último bloque de cada archivo. En la figura 105, se representa la asignación enlazada.

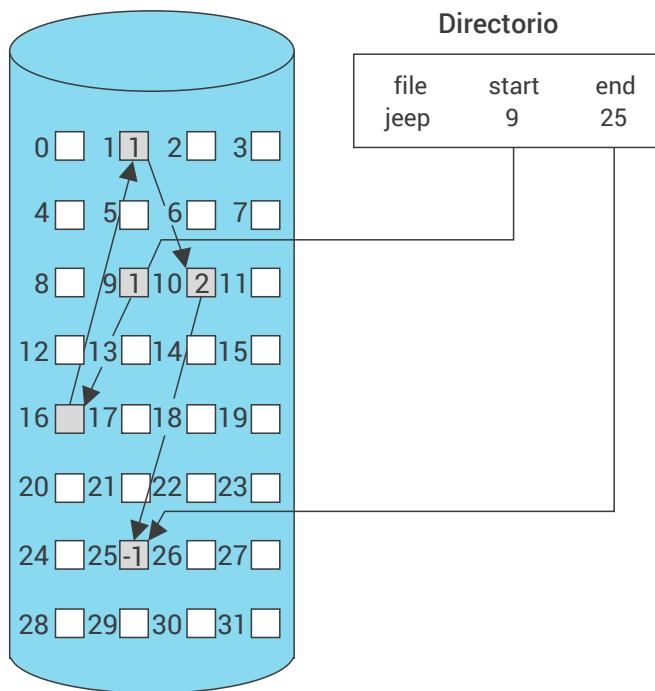


Figura 105. Asignación enlazada.

Fuente: Silberschatz, Galvin y Gagne (2013)

Una de las desventajas de esta asignación es que se necesita espacio adicional para almacenar los punteros, otra desventaja es que solo se lo puede utilizar de manera efectiva para archivos de acceso secuencial.

12.5.3. Asignación indexada

La asignación indexada agrupa todos los punteros en una única ubicación denominada como bloque de índice, como se puede observar en la figura 106, el bloque de índice contiene todos los punteros del bloque.

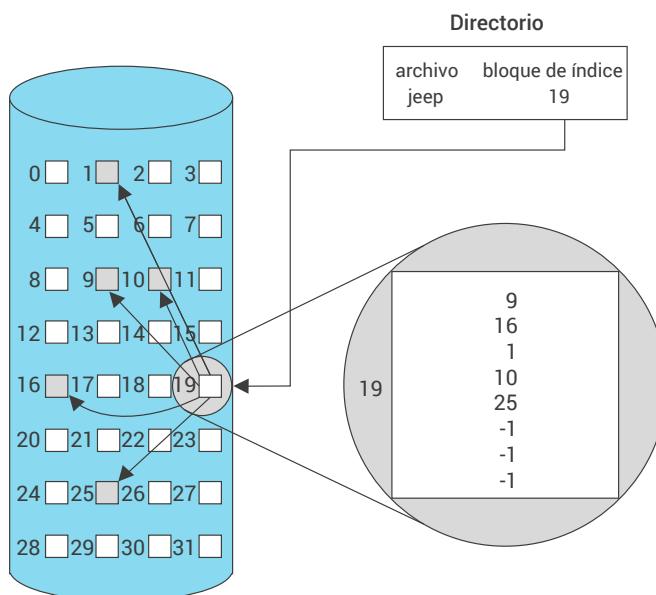


Figura 106. Asignación indizada.

Fuente: Silberschatz, Galvin y Gagne (2013)

Con esta temática se ha finalizado la unidad de gestión de archivos, se sugiere realizar las actividades recomendadas.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Realice un cuadro comparativo de los métodos de administración de espacio libre.

- **Procedimiento**

Como estrategia de desarrollo se recomienda utilizar como base los siguientes recursos:

Candela, S., García, C., Quesada, A., Santana, F., y Santos, J. (2007). *Fundamentos de Sistemas Operativos teoría y ejercicios resueltos*. Madrid: Thomson.

Silberschatz, A., Galvin, P., y Gagne, G. (2013). *Operating Systems Concepts*. Estados Unidos: Wiley.

Stallings, W. (2005). *Sistemas operativos Aspectos internos y principios de diseños*. Madrid, España: Pearson Prentice Hall.

Tanenbaum, A. (2009). *Sistemas Operativos Modernos*. México: Pearson Prentice Hall.



Autoevaluación 12

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. El nombre de un archivo se divide en dos partes:
 - a. Un nombre y una extensión.
 - b. Obligatorios o sugeridos.
 - c. Bloqueo compartido, bloqueo exclusivo.
2. El atributo de un archivo que mantiene la fecha de creación, la última modificación el último uso del mismo, es:
 - a. Tipo.
 - b. Ubicación.
 - c. Hora, fecha e identificación del usuario.
3. La colección de información relacionada, con un nombre, que se graba en almacenamiento secundario se conoce como:
 - a. Archivo.
 - b. Directorio.
 - c. Proceso.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

4. Los directorios con estructuras de árboles:
 - a. Permiten a los usuarios crear sus propios subdirectorios y organizar sus archivos.
 - b. Crean directorios separados para cada usuario, cada usuario tendrá su propio directorio que contendrá sus propios archivos.
 - c. Todos los archivos están contenidos en un mismo directorio y resulta fácil de mantener y comprender.
5. Un directorio de un único nivel se caracteriza por:
 - a. Permitir a los usuarios crear sus propios subdirectorios.
 - b. Crear directorios separados para cada usuario, cada usuario tendrá su propio directorio que contendrá sus propios archivos
 - c. Todos los archivos están contenidos en un mismo directorio.
6. La estructura de directorios de gráfica acíclica:
 - a. Cada usuario tiene su propio directorio.
 - b. Permite el compartimiento de archivos o directorios.
 - c. Tiene la forma de un árbol y es la estructura de directorios más común.
7. La asignación contigua:
 - a. Tiene todos los apuntadores juntos en una sola ubicación (bloque índice).
 - b. Requiere que cada archivo ocupe un conjunto de bloques contiguos en el disco.
 - c. Cada archivo es una lista enlazada de bloques de discos, los bloques pueden estar dispersos en cualquier parte del disco.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8. Para acceder a los archivos que se encuentran en una cinta magnética se lo realiza utilizando el acceso:
 - a. Acceso Secuencial.
 - b. Acceso Enlazado.
 - c. Acceso Directo.
9. El atributo “Identificador” se refiere a:
 - a. El nombre del archivo simbólico es la única información que se mantiene en un formato legible por parte de las personas.
 - b. Una etiqueta unívoca, que usualmente es un número, identifica el archivo dentro del sistema de archivos.
 - c. Es un puntero a un dispositivo y a la ubicación del archivo dentro de dicho dispositivo.
10. Los mecanismos de protección proporcionan un acceso controlado limitando los tipos de accesos a archivos. Podemos controlar varios tipos de operaciones diferentes:
 - a. Propietario, Grupo de usuarios, Otros usuarios.
 - b. Nombre, Tipo, Ubicación, Tamaño, Protección, Fecha, hora e identificador del usuario.
 - c. Lectura, Escritura, Ejecución, Adición, Borrado, Listado.

[Ir al solucionario](#)



Semana 15



Unidad 13. Gestión de dispositivos de E/S

13.1. Fundamentos de gestión de dispositivos

Según Silberschatz, Galvin y Gagne (2013) el gestor de dispositivos de E/S cumple con las siguientes funciones:

- Administrar la memoria que incluye manejo de buffers, asignación de caché y *spooling*.
- Proporcionar una interfaz general de manejadores de dispositivos.
- Brindar controladores para dispositivos de *hardware* específicos.

Los dispositivos varían en aspectos como el modo de transferencia de datos, método de acceso, planificación de transferencia, velocidad del dispositivo, la dirección de E/S o el estado de compartimiento. Aparentemente todas estas diferencias entre los dispositivos complicarían el trabajo del sistema operativo; de ahí que trabajar con el subsistema de E/S independiente del *hardware* simplifica la tarea tanto de los creadores del sistema operativo como de los fabricantes

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

de *hardware*. De esta forma, se pueden manejar distintos tipos de dispositivos, incluso a nivel de sockets para el trabajo con redes.

Silberschatz, Galvin y Gagne (2006), mencionan que en el campo de la tecnología de dispositivos de E/S se experimentan dos tendencias que están en conflicto mutuo. Por un lado, la creciente estandarización de las interfaces *software* y *hardware*. Esta tendencia apoya a incorporar generaciones mejoradas de dispositivos dentro de las computadoras de sistemas operativos existentes; por otro lado, se tiene la tendencia que cada vez más hay una variedad amplia de dispositivos de E/S constituyendo un desafío incorporarlos en las computadoras y sistemas operativos.

De lo anteriormente mencionado, se puede concluir que el papel del sistema operativo en la E/S de una computadora juega un papel importantísimo para gestionar y controlar las operaciones y dispositivos de E/S.

Los conceptos que se deben diferenciar y tener clara su definición en este capítulo son:

1. **Puerto:** es el punto de conexión a través de los cuales los dispositivos se comunican con la máquina a través del envío de señales que pueden ser a través de un cable o del aire.
2. **Controlador:** colección de componentes electrónicos que permiten controlar un puerto, un *bus* o un dispositivo.
3. **Driver:** es el *software* que permite que el sistema operativo pueda controlar un dispositivo de *hardware*.

13.2. Tipos de dispositivos de E/S

Existe una gran variedad de dispositivos de E/S, dependiendo del aspecto que se considere los mismos, según Silberschatz, Galvin y Gagne (2013), se clasifican de acuerdo con el detalle de la tabla 11.

Tabla 11. *Tipos de dispositivos de E/S*

Aspecto	Variación	Ejemplo
Modo de transferencia de datos	<ul style="list-style-type: none"> ▪ Carácter. ▪ Bloque 	<ul style="list-style-type: none"> ▪ Terminal ▪ Disco
Método de acceso	<ul style="list-style-type: none"> ▪ Secuencial ▪ Aleatorio 	<ul style="list-style-type: none"> ▪ Módem ▪ CD-ROM
Planificación de transferencia	<ul style="list-style-type: none"> ▪ Síncrono ▪ Asíncrono 	<ul style="list-style-type: none"> ▪ Teclado ▪ Cinta Magnética
Compartición	<ul style="list-style-type: none"> ▪ Dedicado ▪ Compartible 	<ul style="list-style-type: none"> ▪ Teclado ▪ Cinta Magnética
Velocidad de operación	<ul style="list-style-type: none"> ▪ Latencia ▪ Tasa de transferencia ▪ Retardo entre las operaciones 	
Dirección de E/S	<ul style="list-style-type: none"> ▪ Solo lectura ▪ Solo escritura ▪ Escritura-Lectura 	<ul style="list-style-type: none"> ▪ CD-ROM ▪ Controlador gráfico ▪ Disco

Fuente: Silberschatz, Galvin y Gagne (2013)

Adicionalmente, la E/S puede ser bloqueante y no bloqueante de acuerdo con los siguientes criterios.

- **E/S bloqueante:** se suspende la ejecución de la aplicación, cuando se ejecuta una llamada al sistema bloqueante.
- **E/S no bloqueante:** como su nombre lo indica, la aplicación no se bloquea y se puede seguir utilizando los dispositivos de E/S.

Las solicitudes de E/S pueden ser atendidas a través de DMA, interrupciones y escrutinio. A continuación, se detalla la operación de la DMA debido a que los temas de interrupciones y escrutinio se han revisado anteriormente.

13.3. Acceso directo a memoria (DMA)

Un aspecto importante en la gestión de E/S es el acceso directo a memoria, en donde la CPU puede solicitar datos de un controlador de E/S un *bit* a la vez, pero al hacerlo se desperdicia el tiempo de la CPU, por lo que a menudo se utiliza un esquema distinto, conocido como DMA (Acceso Directo a Memoria). El sistema operativo solo puede utilizar DMA si el *hardware* tiene un controlador de DMA; por lo general, todos los sistemas lo tienen. En la figura 107, se representa el funcionamiento del uso de la DMA.

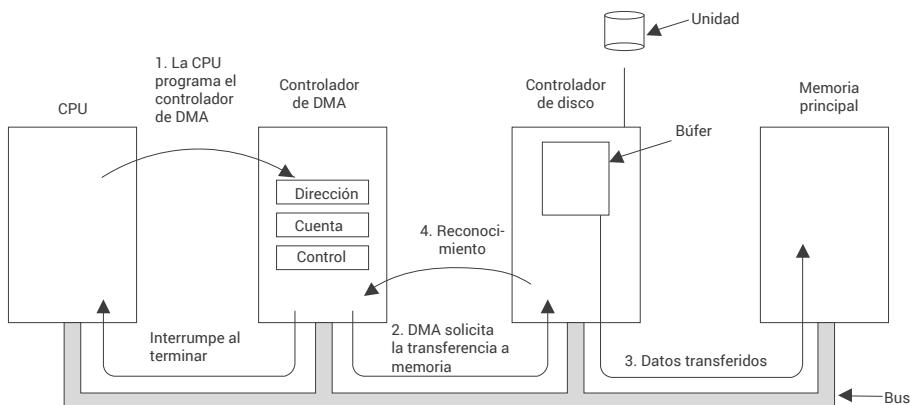


Figura 107. Operación de transferencia de DMA

Fuente: Tanenbaum (2009)

13.4. Subsistema de E/S del kernel

El núcleo ofrece algunos servicios relacionados con la E/S a fin de mejorar la eficiencia de la computadora. Estos servicios son:

- a. **Planificación de las operaciones de E/S:** planificar un conjunto de solicitudes de E/S significa determinar un orden adecuado en el que se puedan ejecutar, con lo cual se logra mejorar el rendimiento del sistema al compartir equitativamente el acceso a los dispositivos entre los procesos solicitantes.

Para lo cual los desarrolladores de sistemas operativos implementan mecanismos de planificación a través de una cola de espera de solicitudes.

- b. **Almacenamiento en búfer:** el área de memoria que almacena datos mientras se están transfiriendo entre dispositivos o en entre dispositivo y aplicación se conoce como búfer.

Las razones para almacenar en un búfer son por:

- Adaptación de velocidad de transferencia entre el productor y consumidor de los datos.
- Adaptación entre dispositivos que tienen diferentes tamaños de transferencia de datos.
- Soportar la semántica de copia en la E/S de aplicaciones.

- c. **Almacenamiento en caché:** partiendo de que una caché es una región de memoria rápida en la que alojan copias de ciertos datos, el acceso a una copia es más eficiente que el acceso a la original.

Se diferencia con el almacenamiento en búfer debido a que se pueden almacenar varias copias en un dispositivo de almacenamiento más rápido.

- d. **Gestión de colas y reserva de dispositivos:** la cola de un dispositivo es un búfer que almacena la salida dirigida a un dispositivo, en este caso el sistema operativo proporciona una interfaz de control que permite a los usuarios y a los administradores del sistema para visualizar la cola de solicitudes.

- e. **Tratamiento de errores:** cuando un sistema operativo utiliza memoria protegida debe defenderse de los tipos de errores de

hardware y software, de modo que cualquier error pequeño no provoque un fallo en el sistema completo. Los dispositivos de E/S pueden fallar por razones transitorias o permanentes.

- f. **Protección de E/S:** los errores están estrechamente relacionados con los aspectos de protección; para lo cual se utilizan mecanismos. Para evitar que los usuarios realicen operaciones de E/S ilegales se deben definir todas las instrucciones de E/S como instrucciones privilegiadas.

Según Silberschatz, Galvin y Gagne (2013), el subsistema de E/S del *kernel* se encarga de los siguientes procedimientos:

- Gestión del espacio de nombres para archivos y dispositivos.
- Control de acceso a los archivos y dispositivos.
- Control de operaciones.
- Asignación de espacio en el sistema de archivos.
- Asignación de dispositivos.
- Almacenamiento en búfer.
- Almacenamiento en caché y gestión de colas de impresión.
- Planificación de E/S.
- Monitorización del estado de dispositivos, tratamiento de errores y recuperación de fallos.
- Configuración e inicialización de controladores de dispositivos.



Actividades de aprendizaje recomendadas

Actividad 1:

- **Actividad de aprendizaje**

Realice un cuadro de los principales aspectos que diferencian cada tipo de dispositivo; con el objetivo de que identifique a qué componente del sistema de entrada y salida pertenece.

- **Procedimiento**

Como estrategia de desarrollo puede utilizar como base los siguientes recursos:

Candela, S., García, C., Quesada, A., Santana, F., y Santos, J. (2007). Fundamentos de Sistemas Operativos teoría y ejercicios resueltos. Madrid: Thomson.

Silberschatz, A., Galvin, P., y Gagne, G. (2013). Operating Systems Concepts. Estados Unidos: Wiley.

Stallings, W. (2005). Sistemas operativos Aspectos internos y principios de diseños. Madrid, España: Pearson Prentice Hall.

Tanenbaum, A. (2009). Sistemas Operativos Modernos. México: Pearson Prentice Hall.



Autoevaluación 13

Arquitectura de Computadoras y Sistemas Operativos.

Desarrolle las autoevaluaciones, responda correctamente a las cuestiones plateadas.

1. Uno de los objetivos del sistema de E/S es:
 - a. Proporcionar la información que el sistema necesita para iniciar un SO a partir de dicho volumen.
 - b. Proporcionar manejadores (drivers) para los dispositivos concretos.
 - c. Proporcionar información sobre la memoria disponible.
2. Un controlador es:
 - a. Una interfaz uniforme de acceso a dispositivos con el subsistema de E/S.
 - b. Un punto de conexión llamado puerto.
 - c. Un conjunto de componentes electrónicos que pueden operar un puerto, un bus o un dispositivo.
3. Un driver es:
 - a. Una interfaz uniforme de acceso a dispositivos con el subsistema de E/S.
 - b. Un punto de conexión llamado puerto.
 - c. Un conjunto de componentes electrónicos que pueden operar un puerto, un bus o un dispositivo.

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4. La E/S con bloqueo:
 - a. Cuando una aplicación emite una llamada al SO con bloqueo, se suspende la ejecución de dicha aplicación.
 - b. Se utiliza solo para dispositivos de entrada.
 - c. Cuando una aplicación emite una llamada al SO con bloqueo, no se suspende la ejecución de dicha aplicación.
5. Los dispositivos que pueden ser usados de manera concurrente por varios procesos son:
 - a. Dispositivos asíncronos.
 - b. Dispositivos síncronos.
 - c. Dispositivos Compartidos
6. De las diferentes técnicas de E/S, ¿cuál libera de más trabajo de E/S a la CPU?
 - a. Técnica de Polling.
 - b. Interrupciones.
 - c. DMA
7. El controlador de E/S y la memoria intercambian datos directamente, sin la intervención de la CPU, cuando se tiene:
 - a. E/S controlada por programa.
 - b. E/S por interrupciones.
 - c. DMA

8. El subsistema de E/S, es:
- Un driver de dispositivo que contiene el código que permite a un sistema operativo controlar un determinado tipo de dispositivo de E/S.
 - El componente del sistema operativo que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos independientes de los mismos.
 - Una función del núcleo encargada de atender una determinada interrupción.
9. Los manejadores de interrupciones son:
- Un driver de dispositivo que contiene el código que permite a un sistema operativo controlar un determinado tipo de dispositivo de E/S.
 - El componente del sistema operativo que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos independientes de los mismos.
 - Una función del núcleo encargada de atender una determinada interrupción.
10. Una interrupción es el mecanismo en el que:
- El controlador indica su estado mediante el bit busy en el registro status.
 - El procesador se comunica con el controlador leyendo y escribiendo patrones de bits.
 - Le permite al controlador de hardware notificar a la CPU cuando un dispositivo está listo para servicio.

[Ir al solucionario](#)

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas



Actividades finales del bimestre:



Semana 16

Evaluación presencial.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



4. Solucionario

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1	a	Son los atributos que el programador debe conocer:
2	c	El Conjunto de instrucciones, bits utilizados para representar datos, mecanismos de E/S y técnicas para direccionamiento de memoria.
3	a	La arquitectura básica no cambia, lo que asegura la compatibilidad del software entre generaciones de procesadores, lo que cambia año a año son características referentes a la organización, velocidad, tamaño de memorias, tecnología de materiales, etc.
4	a	La estructura nos indica los componentes del computador en cuanto a su relación, en la otra mano tendríamos la Operación, que describe el funcionamiento de cada estructura.
5	b	ALU, Unidad de control, memoria, E/S, son las estructuras básicas que tiene todo computador.
6	b	El número de bits usados para representar instrucciones y datos es una característica arquitectónica.
7	c	Los literales a y b son incorrectos ya que son funciones complejas que realizan programas especializados. En resumen, sólo hay cuatro operaciones básicas.
8	a	La unidad de control decodifica las instrucciones y emite las señales hacia los componentes como ALU y registros para el funcionamiento del computador.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 1

Pregunta	Respuesta	Retroalimentación
9	b	El literal b es el correcto ya que nombra los principales componentes estructurales. Los literales a y c nombran unidades funcionales.
10	c	O también llamado buses.

Ir a la
autoevaluación



Autoevaluación 2		
Pregunta	Respuesta	Retroalimentación
1	F	El bloque de instrucciones se encuentra en la RAM, la ALU sólo se encarga de realizar las operaciones aritméticas y lógicas.
2	V	La ALU ejecuta las operaciones físicamente (hardware), a menor complejidad de este componente el procesador se simplifica y los fabricantes pueden enfocarse en el incremento de velocidad en las operaciones, en lugar de la complejidad de las mismas.
3	V	Las compuertas lógicas permiten operar dos bits a la vez, es necesario generar un sistema digital compuesto de varias compuertas para poder realizar operaciones como la suma binaria.
4	F	Un sumador completo permite sumar hasta 3 bits a la vez, el tercer bit es generalmente el acarreo de la operación anterior.
5	V	El algoritmo de Booth permite utilizar operaciones sencillas como la suma y el desplazamiento de registros, que se lo hace sobre hardware, para realizar la operación de la multiplicación.
6	V	Según la arquitectura de Von Neumann la ALU podría comunicarse directamente con la memoria. Sin embargo en la práctica interviene siempre la unidad de control.
7	V	La ALU sólo realiza operaciones con datos numéricos. Es necesaria la intervención de la unidad de control para controlar qué operación debe realizar la ALU con dichos datos.
8	b	Si el resultado sobrepasa el ancho de palabra máximo, se emite una bandera llamada "overflow" o desborde matemático.
9	a	Las compuertas lógicas están formadas a su vez por transistores.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
10	a	La ALU es en esencia sólo hardware, las operaciones las realiza una estructura formada por compuertas lógicas y Flip Flops.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 3		
Pregunta	Respuesta	Retroalimentación
1	a	En esencia, la caché es una copia de la RAM, pero sólo de las instrucciones prioritarias.
2	b	La caché, aunque más pequeña que la RAM, es mucho más rápida, es decir, el tiempo de acceso es mucho menor.
3	b	Este proceso se puede extender a toda la jerarquía de memoria, por ejemplo: si la instrucción no se encuentra en la RAM, se puede recurrir a buscarla en la memoria virtual.
4	a	Un menor tiempo de acceso significa una memoria más rápida, esta, tendrá mejor tecnología, y por ende será más costosa.
5	a	Menor tamaño y mayor densidad (número de bits por unidad de área), es más costoso debido a la tecnología.
6	b	Se prefiere las memorias más costosas en el primer nivel de jerarquía, debido a que son costosas se diseñan de poca capacidad. Para almacenar grandes volúmenes de datos, se prefieren las memorias más lentas y por ende más baratas, como los discos duros.
7	b	La correspondencia directa es la técnica más simple de implementar, pero es ineficiente.
8	c	El menos recientemente usado. (LRU). Se ha demostrado que tiene un nivel de eficiencia ligeramente superior a los: LFU, y FIFO.
9		Chip del Bios; Tablas de funciones; Programas del sistema
10		EPROM: Borrado por UV. EEPROM: Borrado eléctrico. Flash: Borrado eléctrico pero mucho más rápido de las anteriores.

Ir a la
autoevaluación

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
1		Código de operación, operando fuente, operando destino, dirección siguiente instrucción.
2	d	Los operandos pueden estar distribuidos en distintas estructuras como en memoria principal, registros, E/S, etc.
3	a	Dentro de las 4 funciones básicas del procesador, está el Procesamiento de datos, que no es más que operar aritméticamente o lógicamente datos.
4	a	La ventaja del utilizar el acumulador como registro direccionable, es el de simplificar la instrucción (más pequeña)
5	c	Esto se traduce en que el trabajo de la decodificación de las instrucciones es más simple, y por ende, la unidad de control tendrá menos hardware, también disminuye el consumo de energía del mismo.
6		Tres direcciones: MUL Y, A, B SUB T,C,D ADD T,T,Y SUB Y,E,F DIV T,T,Y
7		Dos direcciones: MOVE Y,A MUL Y,B MOVE T,C SUB T,D ADD T,Y MOVE Y,E SUB Y,F DIV T,Y



Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
8		Una dirección. LOAD E SUB F STOR Y LOAD A MUL B STOR X LOAD C SUB D ADD X DIV Y
9		El de una dirección.
10		Mayor, debido a que contiene instrucciones más complejas.

Ir a la
autoevaluación

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)

Autoevaluación 5

Pregunta	Respuesta	Retroalimentación
1	c, b	La unidad de control no almacena datos, decodifica las instrucciones, temporiza señales y secuencia las operaciones con la ALU.
2	b	El contador de programa PC, está en la categoría de control y estado, es difícilmente manipulable por el programador ya que un error en este registro puede detener al procesador.
3		T1: MAR <- IR T2: MBR<-memoria. T3: R1<- (R1)*MBR.
4		T1: MAR <- IR T2: MBR<-R1. T3: memoria<- MBR.
5	b	Este proceso siempre es secuencial, la interrupción se comprueba luego de cada ejecución.
6	c	En la captación se decodifica la instrucción, separando el código de operación de los operandos o direcciones.
7		T1: MAR<-PC T2: MBR<-memoria. PC<- (PC)+1 T3: IR<-(MBR)
8		Programa, tiempo, E/S, fallo de hardware.
9	F	En el caso de detectar una interrupción el contador de programa guarda en memoria la cuenta que llevaba y se le transfiere una nueva secuencia de conteo del programa que atenderá la interrupción.
10	V	Siempre luego de ejecutar una instrucción, el procesador dedica un ciclo de reloj para revisar si algún dispositivo a habilitado la línea de interrupción.

[Ir a la autoevaluación](#)

Autoevaluación 6		
Pregunta	Respuesta	Retroalimentación
1	a	El módulo de E/S contiene la lógica necesaria para traducir la información de los periféricos a un formato y velocidad con los que el procesador pueda trabajar.
2	a	El DMA puede acomodarse en una arquitectura de bus, de tipo único, donde se conecta un módulo DMA.
3	b	El DMA es tanto hardware como software.
4	b	Sobre el bus de datos se ubica la información extraída desde las localidades de la memoria, o se transfiere los resultados desde el procesador hacia memoria o E/S.
5	c	Debe tener al menos 16 bits, en el caso de tener menos ancho, por ejemplo 8 bits, los ciclos de reloj necesarios para ejecutar una instrucción deberán al menos duplicarse.
6	a	Los buses pueden ser dedicados (sólo comunicar dos estructuras) o multiplexados (necesarios para comunicar varias estructuras, por ejemplo: varios módulos de E/S).
7	b	El arbitraje en un bus que conecta varias estructuras debe controlarse. Dependiendo de la arquitectura puede darse un ambiente centralizado donde un solo dispositivo controla el tráfico de datos, o distribuido, donde el control se comparte entre todos los dispositivos.
8	c	La temporización es la ventana de tiempo que se asigna a un dispositivo para transmitir o recibir datos. Puede ser síncrona (controlada por el reloj del sistema), o asíncrona, (controlada por señales de control).
9	F	El bus del sistema se reserva sólo para los dispositivos vitales: memoria, ALU y CPU.
10	V	El bus PCI tiene la velocidad suficiente para operar como bus del sistema.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 7

Pregunta	Respuesta	Retroalimentación
1	b	De acuerdo a las definiciones de la sección 7.1; en donde se menciona que el SO es el programa que gestiona el hardware y el software de la computadora; además que actúa como intermediario entre el usuario y el hardware de la computadora.
2	c	Corresponde a los objetivos del SO, de facilidad de uso y eficiencia del uso de los recursos del sistema computacional.
3	b	Desde el punto de vista de un usuario el sistema operativo debe cumplir con los requisitos de usabilidad y maximizar la utilización de los recursos.
4	c	Considerando que un SO es un programa de control, este debe gestionar los programas del usuario; con el objetivo de evitar errores y mejorar el uso de la computadora.
5	a	La principal característica de la multiprogramación es el incremento del uso de la CPU, organizando los trabajos para que la CPU siempre esté ocupada.
6	a	Como se puede observar en la sección 7.2.2, los sistemas de procesamiento por lotes fueron los predominantes en esta generación
7	a	En la sección 7.3.1, se menciona como ejemplo de un sistema monolítico a los SO basados en Unix.
8	b	En la sección 7.3.1, se menciona que este tipo de sistemas se caracteriza por tener estructura básica o mínima.
9	b	En la sección 7.3.2, se menciona que los sistemas por capas, se descomponen en varias capas y cada una utiliza las funciones y servicios de la capa anterior.
10	c	En la sección 7.3.4, se indica, que la principal función del microkernel es proporcionar un mecanismo de comunicaciones entre el programa cliente y los distintos servicios que se ejecutan en el espacio de usuario.

Ir a la
autoevaluación

Autoevaluación 8		
Pregunta	Respuesta	Retroalimentación
1	c	En la sección 8.1.1, se menciona que un proceso es la unidad de trabajo en un sistema; por lo cual es considerado como un programa en ejecución.
2	a	En la sección 8.2, se menciona que el planificador a largo plazo determina que programas se admiten en el sistema para realizar el procesamiento; es decir que se encarga de controlar el grado de multiprogramación.
3	b	El cambio de contexto o cambio de proceso en el uso de la CPU, se considera un tiempo muerto, debido a los cambios de procesos que se producen en el uso de la CPU.
4	b	Los procesos que afectan y se pueden ver afectados por otros procesos se los conoce como cooperativos (Revise la sección 8.1.5)
5	b	En la sección 8.1.4; se menciona que elemento de información de planificación de CPU del BCP, contiene la información de prioridad del proceso e información de planificación adicional.
6	a	Se considera que un proceso está en ejecución cuando está haciendo uso de la CPU y de los otros recursos de hardware en el caso de que amerite.
7	b	El algoritmo Round Robin, es considerado como un algoritmo equitativo en la asignación del uso de la CPU, ya que a todos los procesos les asigna el CPU mismo intervalo de tiempo.

Autoevaluación 8

Pregunta	Respuesta	Retroalimentación																																										
8	c	<p>En este caso no se han proporcionado tiempos de llegada; por lo cual se asume que todos los procesos llegaron en el tiempo cero en el orden: P1, P2, P3, P4, P5.</p> <p>Ejecutando el algoritmo FIFO, se obtiene como tiempo de retorno del proceso P3 13 unidades de tiempo, lo cual se lo muestra en el siguiente diagrama de gannt</p> <table border="1"> <thead> <tr> <th></th> <th>P1</th> <th>P2</th> <th>P3</th> <th>P4</th> <th>P5</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>10</td> <td>11</td> <td>13</td> <td>14</td> <td>20</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td colspan="2">Tiempo de Retorno</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>P1=10</td> <td></td> <td>P4=14</td> <td></td> <td></td> </tr> <tr> <td></td> <td>P2= 11</td> <td></td> <td>P5= 20</td> <td></td> <td></td> </tr> <tr> <td></td> <td>P3= 13</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		P1	P2	P3	P4	P5	0	10	11	13	14	20								Tiempo de Retorno						P1=10		P4=14				P2= 11		P5= 20				P3= 13				
	P1	P2	P3	P4	P5																																							
0	10	11	13	14	20																																							
	Tiempo de Retorno																																											
	P1=10		P4=14																																									
	P2= 11		P5= 20																																									
	P3= 13																																											
9	c	El algoritmo SJF es el selecciona al siguiente proceso que tiene la menor duración para asignarle la CPU. (Revise la sección 8.2.2.2)																																										
10	b	El tiempo de espera, es el tiempo que un proceso esperó para que se le asigne la CPU.																																										

[Ir a la autoevaluación](#)

Autoevaluación 9		
Pregunta	Respuesta	Retroalimentación
1	a	La sección crítica se refiere a la sección de código dentro de un proceso que requiere acceso a recursos compartidos y que no puede ser ejecutada mientras otro proceso esté en una sección de código correspondiente.
2	c	La exclusión mutua se refiere a que si el hilo TI, se está ejecutando en su sección crítica, entonces ningún otro hilo puede estar en la misma ejecución.
3	a	La exclusión mutua, tiene como política que mientras un proceso se está ejecutando en su sección crítica, ningún otro proceso se puede ejecutar.
4	b	El requisito de progreso se refiere a que si ningún hilo se está ejecutando en su sección crítica y existen algunos hilos que desean entrar en sus secciones críticas, entonces aquellos
5	a	Como se menciona en la sección 9.2; la espera limitada se refiere a que otros hilos tienen permitido entrar a su sección crítica, después de que un hilo ha realizado la solicitud para entrar en su sección crítica y antes de que se conceda la solicitud.
6	a	El semáforo es una variable entera, a la cual solo se accede mediante dos operaciones estándar: wait () y signal () .
7	b	Los monitores son considerados como una variable local que sólo son accesibles por los procedimientos del monitor y no por ningún otro procedimiento.
8	b	Como se menciona en la sección 9.4; los semáforos contadores sólo pueden variar en un dominio.
9	a	La condición de exclusión mutua, tiene como política que mientras un proceso se está ejecutando en su sección crítica, ningún otro proceso se puede ejecutar.
10	c	El problema de la sección crítica se produce en sistemas donde los procesos son concurrentes y existe competencia por el uso de la CPU.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 10		
Pregunta	Respuesta	Retroalimentación
1	b	En interbloqueos , la exclusión mutua se refiere que solo un proceso puede ser utilizado a la vez, si hay otro proceso solicita el recurso, debe esperar hasta que el recurso sea liberado.
2	c	Como se menciona en la sección 10.2, para que se produzca un interbloqueo se deben producir las cuatro condiciones: exclusión mutua, retención y espera, sin desalojo y espera circular.
3	a	Un grafo de asignación de recursos, está formado por un conjunto de vértices y de aristas.
4	c	Para que se produzca un interbloqueo, no es necesario que se produzca un estado seguro.
5	a	En la sección 10.4, se indica que una de las desventajas de la técnica de prevención es que se deben conocer los futuros requisitos de recursos.
6	b	En la sección 10.6, se indica que la principal desventaja de la estrategia de detección es que pueden ocurrir pérdidas por expropiación de los recursos.
7	a	Una de las desventajas de las técnicas de prevención, es que los procesos deben conocer los futuros requisitos de recursos que necesitarán para ejecutarse.
8	c	El grafo de asignación de recursos es utilizado en la predicción de interbloqueos.
9	c	Un interbloqueo se produce cuando se cumplen simultáneamente las condiciones de exclusión mutua, retención y espera, no existencia de expropiación y espera circular.
10	a	El grafo corresponde a una asignación de recursos con interbloqueo, debido a que se cumplen las condiciones necesarias para producirse un interbloqueo.

Ir a la
autoevaluación



[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)**Autoevaluación 11**

Pregunta	Respuesta	Retroalimentación
1	b	La Fragmentación externa es la memoria que no se puede asignar por no encontrarse contigua.
2	b	La compactación es la combinación de los huecos en la memoria, a través del desplazamiento de los mismos a la zona más baja de la memoria
3	a	Se considera como dirección lógica a la que es generada por la CPU, también se la conoce como dirección virtual.
4	a	La fragmentación interna se produce, en los esquemas de gestión de memoria con partición fija.
5	c	Resolución: Dirección Virtual: (0,128) Segmento (S): 0 Desplazamiento (D): 128 Límite (L): 300 $D \leq L$ $128 \leq 300$ Si, entonces calcular DF $DF = (B+D)$ $DF = (500+128) = 628$
6	b	En el esquema de paginación se hace necesario tener compactación de memoria.
7	a	Como se menciona en la sección 11.2.5, el esquema de memoria virtual se encarga de ejecutar programas que son demasiados grandes como para caber en la memoria principal; por lo cual su función es permitir la ejecución de procesos que están parcialmente en memoria principal.
8	b	El algoritmo de sustitución de páginas LRU, tiene como objetivo sustituir la página que no haya sido utilizada durante el periodo más largo de tiempo.
9	a	El número de procesos que se ejecutan en memoria principal dependen de la capacidad física de la memoria principal.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Autoevaluación 11

Pregunta	Respuesta	Retroalimentación																			
10	a	1	2	3	4	2	1	5	6	2	1	3	7	6	3	2	1	2	3	6	
		1	1	1	1			1	1			1	1	6			6				
			2	2	2			2	2			2	2	2			2				
			3	3				5	5			2	3	3			3				
			4				4	6			6	7	7			1					

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 12		
Pregunta	Respuesta	Retroalimentación
1	a	El nombre de un archivo, está formado por un nombre y una extensión.
2	c	El atributo Fecha, de un archivo es el que se encarga
3	a	Un archivo, es el conjunto de información con sentido y que se almacena en dispositivos de almacenamiento secundario.
4	a	La estructura de árboles de directorios, es en la cual los usuarios pueden crear sus propios subdirectorios y organizar los archivos correspondientes.
5	c	Los directorios de un único nivel se caracterizan porque los archivos de encuentran en un mismo directorio.
6	b	Los directorios de gráfica acíclica se caracterizan porque permiten la compartición de archivos y/o directorios.
7	c	En la asignación contigua cada archivo, constituye una lista enlazada de bloques de discos, que pueden estar en cualquier parte del disco.
8	a	El acceso utilizado en una cinta magnética, es el secuencial.
9	b	El atributo identificador, es un etiqueta única que se le asigna al archivo para distinguirlo de otros dentro del sistema de archivos.
10	c	Los tipos de acceso a archivos se controlan mediante las operaciones de: lectura, escritura, ejecución, adición, borrado, listado.

Ir a la
autoevaluación

Autoevaluación 13		
Pregunta	Respuesta	Retroalimentación
1	b	Uno de los principales objetivos del sistema de Archivos es proporcionar manejadores para los dispositivos concretos.
2	c	Un controlador es el conjunto de componentes electrónicos que pueden operar un puerto, un bus o un dispositivo.
3	a	El driver es una interfaz uniforme de acceso a dispositivos con el subsistema de E/S.
4	a	Se considera que la E/S con bloqueo se produce cuando una aplicación hace una llamada al SO, se suspende la ejecución de dicha aplicación.
5	c	Los dispositivos compartidos son los que pueden ser utilizados de manera concurrente por varios procesos.
6	c	La DMA es la técnica de E/S que más libera de trabajo a la CPU.
7	c	La técnica de DMA es la que intercambian datos directamente sin la intervención de la CPU.
8	b	El subsistema de E/S, es el que se encarga de realizar las tareas necesarias de las operaciones de E/S que son comunes a todos los dispositivos independientes de los mismos.
9	c	Los manejadores de interrupciones, son las funciones del núcleo encargados de atender una determinada aplicación.
10	c	La interrupción es el mecanismo, que permite al controlador de hardware notificar a la CPU cuando un dispositivo está listo para proporcionar el servicio.

Ir a la
autoevaluación



5. Referencias bibliográficas

Barrera, K. (2016). *Tendencias actuales en los sistemas operativos de escritorio, basados en la web y para el internet of things (iot)*. Quetzaltenango. Obtenido de <https://goo.gl/U5rTbR>

Candela, S., García, C., Quesada, A., Santana, F., y Santos, J. (2007). *Fundamentos de Sistemas Operativos teoría y ejercicios resueltos*. Madrid: Thomson.

Canonical. (2016). *Big software has arrived*. Canonical.

Castellanos, L. (2014). *Sistemas Operativos: una guía de estudios*. Maracaibo.

Castillo, T., y Cueva, S. (2017). *Texto guía Arquitectura de Computadoras y Sistemas Operativos*. Loja: Editorial de la Universidad Técnica Particular de Loja.

Computer History Museum. (1 de Febrero de 2014). *IBM 7094 Computer – console*. Obtenido de Computer History Museum: <https://goo.gl/oMnVqk>

Computer History Museum. (19 de Febrero de 2015). *About the Computer History Museum's IBM 1401 Machines*. Obtenido de Computer History Museum: [Computer History Museum](#).

Deka Kumar, J. (2006) Computer organization and architecture, curso web, IITG. <https://goo.gl/S38y87>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Flynn, I., y McHoes, A. (2001). *Sistemas Operativos*. México: Thomson Learning.

La Red Martínez, L. (2001). *Sistemas Operativos*. Argentina: U.N.N.E.

Milenkovic, M. (1988). *Sistemas operativos. Conceptos y diseño*. Madrid: McGraw-Hil.

Piratas de Silicon Valley (2011). [Película]. Recuperado el 1 de Julio de 2017, de <https://goo.gl/bpzYaE>

Silberschatz, A., Galvin, A., y Gagne, G. (2006). *Fundamentos de sistemas operativos*. España: McGraw-Hill Interamericana.

Silberschatz, A., Galvin, P., y Gagne, G. (2013). *Operating Systems Concepts*. Estados Unidos: Wiley.

Silva, M. (2015). *Sistemas Operativos*. Buenos Aires, Argentina: Alfaomega.

Stallings, W. (2005). *Sistemas operativos Aspectos internos y principios de diseños*. Madrid, España: Pearson Prentice Hall.

Tanenbaum, A. (2009). *Sistemas Operativos Modernos*. México: Pearson Prentice Hall.