

Fundamentos del Software

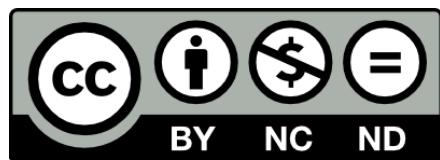


UNIVERSIDAD
DE GRANADA

**Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación**

Los Del DGIIM

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional (CC BY-NC-ND 4.0).

Eres libre de compartir y redistribuir el contenido de esta obra en cualquier medio o formato, siempre y cuando des el crédito adecuado a los autores originales y no persigas fines comerciales.

Fundamentos del Software

Los Del DGIIM

Granada, 2023

Índice general

1. Relaciones de ejercicios	5
1.1. Sistema de Cómputo	5
1.2. Introducción a los Sistemas Operativos	10
1.3. Compilación y Enlazado de Programas	22
1.4. Introducción a las Bases de Datos	28

1. Relaciones de ejercicios

1.1. Sistema de Cómputo

Ejercicio 1.1.1. El método de comunicación de E/S en el que la CPU está esperando hasta que la operación de E/S ha finalizado se conoce como:

- (a) E/S Programada.
- (b) E/S Dirigida por Interrupciones.
- (c) DMA.
- (d) E/S a Distancia.

Ejercicio 1.1.2. El método de comunicación de E/S en el que el dispositivo de E/S informa a la CPU en qué momento está preparado el dispositivo para la transferencia de datos se conoce como:

- (a) E/S Programada.
- (b) E/S Dirigida por interrupciones.
- (c) DMA.
- (d) E/S a Distancia.

Ejercicio 1.1.3. Cuál de las siguientes afirmaciones es correcta:

- (a) En algunas computadoras un programa puede ejecutarse sin necesidad de cargarlo en la memoria principal.
- (b) **Un programa, para que se ejecute, debe estar cargado en la memoria principal**
- (c) Un programa, para que se ejecute, basta con que esté en el disco duro
- (d) Un programa, para que se ejecute, si está en lenguaje máquina, puede estar en cualquier unidad.

Ejercicio 1.1.4. Dado el esquema de un computador elemental según se ha descrito en el tema, el puntero de pila (SP) indica:

- (a) La dirección de memoria donde debe saltar el programa después de ejecutarse la instrucción de retorno correspondiente.
- (b) **La dirección de memoria donde se encuentra la dirección donde debe saltar el programa después de ejecutarse la instrucción de retorno correspondiente¹.**
- (c) La dirección de memoria a donde se ha producido el último salto.
- (d) La dirección de memoria donde se encuentra la dirección a donde se ha producido la última llamada a una subrutina.

Ejercicio 1.1.5. Sea un ordenador elemental con una arquitectura tal y como se muestra en la figura 1.1, es decir, tres registros de propósito general, registro contador de programa (PC) y registro de instrucción (IR). El registro SP (Puntero de pila) contiene la dirección 35 y la pila crece hacia posiciones menores de memoria. La memoria principal dispone de 256 palabras donde cada palabra tiene la longitud necesaria para albergar la instrucción de mayor tamaño. Describa el estado final de ejecución del procesador a partir del estado actual de la CPU mostrado en la figura 1.1. Ponga todos los valores de los registros de cada ciclo de instrucción realizado por el procesador hasta llegar a dicho estado final.

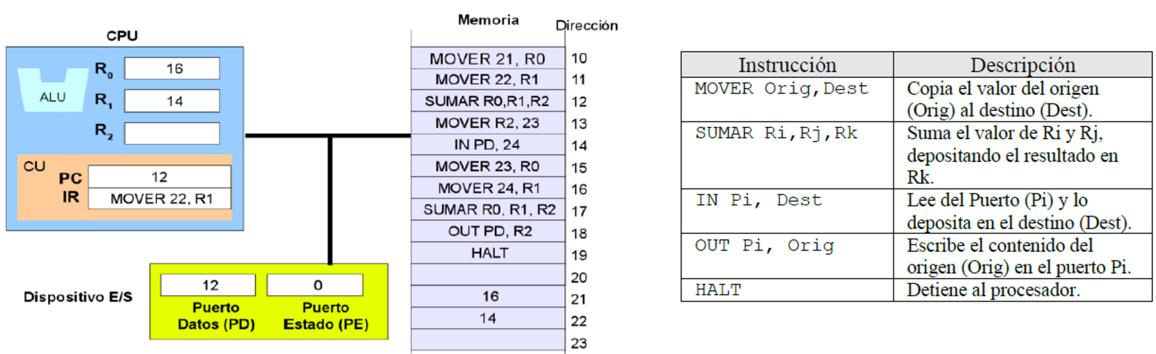


Figura 1.1: Computador del ejercicio 1.1.5

El estado del computador en cada ciclo es:

¹En realidad apunta a la dirección de memoria anterior.

PC	IR	SP	R0	R1	R2	PE	PD
12	MOVER 22, R1	35	16	14		0	12
13	SUMAR R0,R1,R2	35	16	14	30	0	12
14	MOVER R2,23	35	16	14	30	0	12
15	IN PD 24	35	16	14	30	0	12
16	MOVER 23, R0	35	30	14	30	0	12
17	MOVER 24, R1	35	30	12	30	0	12
18	SUMAR R0,R1,R2	35	30	12	42	0	12
19	OUT PD, R2	35	30	12	42	0	42
20	HALT	35	30	12	42	0	42

Tabla 1.1: Ejecución del programa del ejercicio 1.1.5

Memoria	
	:
16	21
14	22
30	23
12	24
	:

Tabla 1.2: Memoria del ejercicio 1.1.5

Ejercicio 1.1.6. Suponiendo que el lenguaje máquina de la arquitectura anterior dispone de 14 instrucciones distintas, muestre cuántos bits serían necesarios para codificar las instrucciones SUMAR R0,R1,R2 y MOVER 20,R0 respectivamente.

Puesto que dispone de 14 instrucciones distintas, se necesitaran 4 bits para codificarlas.

- **SUMAR R0, R1, R2:** Son necesarios además 2 bits para codificar cada uno de los 3 registros. Luego $4 + 3 \cdot 2 = 10$ bits
- **MOVER 20, R0:** Son necesarios además 2 para el registro R0. Para codificar la dirección, puesto que hay $256 = 2^8$ palabras de memoria principal, se necesitan 8 bits por dirección. Luego $4 + 2 + 8 = 14$ bits

Ejercicio 1.1.7. Imagina que el procesador está ejecutando el programa de usuario del ejercicio 1.1.5 y en este momento al terminar de ejecutar la instrucción actual, el procesador se da cuenta de que hay una interrupción pendiente. Escribe los pasos que se dan en el sistema y por quién (software o hardware) hasta que se resuelve el tratamiento de la interrupción y el programa finaliza, sabiendo que la rutina de tratamiento de la interrupción comienza en la dirección de memoria principal 56 y termina en la dirección de memoria principal 70.

En primer lugar, a nivel de hardware, se da:

1. El procesador indica el reconocimiento de la interrupción.

2. El procesador apila PSW y el PC en la pila de control. Es decir:

$$M[35] = PC = 12 \quad M[34 - n] = PSW \quad SP = 33 - n$$

3. El procesador carga un nuevo valor en el PC basado en la interrupción. En este caso, como la interrupción comienza en la dirección 56, $PC = 56$.

A nivel de software, se da:

4. Se salva el resto de la información de estado del proceso.
5. Se procesa la interrupción. En este caso, se ejecutan las instrucciones desde la posición 56 a 70. Es decir, al terminar, $PC = 71$.
6. Se restaura la información de estado del proceso.
7. Se restauran los valores de PSW y PC y se actualiza el puntero de pila. Es decir,

$$PSW = M[34 - n] \quad PC = M[35] = 12 \quad SP = 35$$

Ejercicio 1.1.8. Basándonos en el ejercicio 1.1.7, ¿hay diferencias si en vez de producirse una interrupción se ha producido una excepción? Indique cuales.

Sí, hay diferencias. En primer lugar, las excepciones son un evento síncrono, mientras que las interrupciones son asíncronas. El procesador aquí se ha dado cuenta al finalizar la ejecución de que había una interrupción pendiente, mientras que en el caso de una excepción se ejecuta directamente. Además, por norma general, una excepción conlleva un posible error en el código y cómo resolverlo, mientras que las interrupciones no suelen estar asociadas a errores.

Ejercicio 1.1.9. Sea un ordenador elemental con una arquitectura tal y como se muestra en la figura 1.2, es decir, tres registros de propósito general, registro contador de programa (PC), registro de instrucción (IR) y registro de pila (SP). La memoria principal dispone de 512 palabras donde cada palabra tiene la longitud necesaria para albergar la instrucción de mayor tamaño. Describa el estado final de ejecución del procesador a partir del estado actual de la CPU mostrado en la figura y tras la ejecución del programa (nótese que la instrucción de la dirección 10 ya se ha ejecutado).

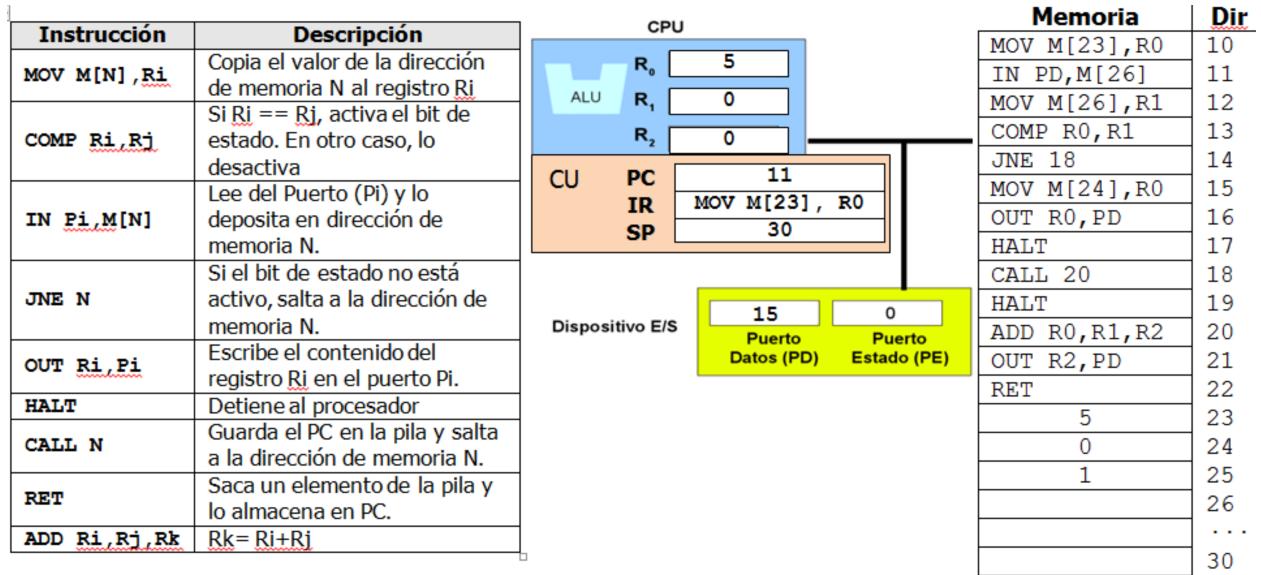


Figura 1.2: Computador del ejercicio 1.1.9

El estado del computador en cada ciclo es:

PC	IR	SP	R0	R1	R2	PE	PD
11	MOV M[23], R0	30	5	0	0	0	15
12	IN PD, M[26]	30	5	0	0	0	15
13	MOV M[26], R1	30	5	15	0	0	15
14	COMP R0, R1	30	5	15	0	0	15
15	JNE 18	30	5	15	0	0	15
16	CALL 20	29	5	15	0	0	15
21	ADD R0, R1, R2	29	5	15	20	0	15
22	OUT R2, PD	29	5	15	20	0	20
23	RET	30	5	15	20	0	20
20	HALT	30	5	15	20	0	20

Tabla 1.3: Ejecución del programa del ejercicio 1.1.9

Memoria	
	10
	:
	22
5	23
0	24
1	25
15	26
	:
19	30

Tabla 1.4: Memoria del ejercicio 1.1.9

1.2. Introducción a los Sistemas Operativos

Ejercicio 1.2.1. Sea un proceso en un SO con su información de contexto, de datos y de código según se muestra en la figura 1.5 y que ya ha sido atendido en un 50% y le resta la otra mitad para finalizar su ejecución. Con la idea de optimizar el espacio de memoria para que el SO pudiera disponer de un mayor número de procesos en ésta, ¿podría reducirse el espacio que ocupa en memoria en alguna de las siguientes instancias?

Código	12922 KB
Pila	3002 KB
Datos	434 KB

Tabla 1.5: Información del proceso del ejercicio 1.2.1

- (a) La lista de procesos.

El proceso aún no ha terminado, por lo que no es posible eliminarlo de la lista de procesos. Sí es cierto que los procesos que ya hayan terminado sí podrían eliminarse de la lista de procesos para liberar memoria; pero como este proceso no ha terminado no puede eliminarse de la lista de procesos.

- (b) Información del contexto del proceso.

El contexto del proceso contiene información importante acerca de este, como todos los registros de control o el PC, por ejemplo. Por tanto, no se puede liberar espacio aquí, ya que se perdería información necesaria para poder continuar con el proceso.

- (c) Tamaño de los datos.

El proceso no ha terminado, por lo que a priori, sin tener más información del proceso, no podemos eliminar datos que no sabemos si serán necesarios en un futuro. Por tanto, tampoco es posible liberar espacio aquí.

- (d) Tamaño del código.

El proceso no ha terminado, por lo que si redujésemos el código podríamos caer en el error fatal de que el programa no pudiese continuar, ya que podríamos haber eliminado instrucciones necesarias para el programa.

El único caso excepcional en el que se podría eliminar parte del código es que el programa no incluyese instrucciones del salto, y justo en este caso sí podríamos quitar parte del programa del principio. No obstante, no sabríamos cuántas instrucciones liminar a priori.

Ejercicio 1.2.2. ¿Por qué cuando un proceso está en modo “ejecutándose” y pretende acceder a una dirección de memoria fuera del área asignada, se informa de que se ha producido un error en la ejecución? ¿Quién informa de ello? Razone la respuesta.

El SO se encarga de asignar a cada proceso solo una parte de la memoria a la que puede acceder, ya que en caso contrario se correría el riesgo de que un proceso

modificarse aspectos importantes en la memoria. Por tanto, no está permitido hacer lo indicado.

Hay un dispositivo hardware (MMU, *Memory Management Unit*) encargado de lanzar la excepción correspondiente a este error. La rutina de tratamiento de dicha excepción es la que informará al usuario después de abortar el proceso.

Ejercicio 1.2.3. ¿Tiene sentido un modelo de 5 estados de los procesos en un SO monousuario? Razone la respuesta.

Si, ya que aunque haya solo un usuario, este puede querer ejecutar simultáneamente más de un proceso.

Ejercicio 1.2.4. Dado un proceso que está en modo “ejecutándose” y pretende acceder a una dirección de memoria fuera del área asignada, lo cual sería un error en la ejecución, ¿a qué modo pasaría dicho proceso? Razone la respuesta.

- (a) Bloqueado.
- (b) No cambia de modo.
- (c) Finalizado.**
- (d) Preparado.

Pasaría a modo finalizado, ya que es un error no recuperable. El programa no podrá continuar con su ejecución, y el SO decidirá abortarlo tras el mensaje de error aportado por el MMU.

Ejercicio 1.2.5. Un planificador de procesos tiene una tarea concreta dentro de un SO multiprogramado. ¿Tiene sentido disponer de un planificador de procesos en un SO monoprogramado? Razone la respuesta.

No, ya que solo puede ejecutar un programa a la vez. La utilidad del planificador de procesos es cambiar de uno a otro para que la CPU no esté sin realizar cálculos. Sin embargo, si solo hay un programa pierde su utilidad.

Ejercicio 1.2.6. Dado un SO multiprogramado, ¿bajo qué circunstancias se podría prescindir del planificador de procesos? Razone la respuesta.

A pesar de ser multiprogramado, cabe la posibilidad de que solo se estén ejecutando, como máximo, tantos procesos como procesadores hay. En ese caso, no sería necesario el planificador de procesos.

No obstante, esto es una situación excepcional y, por norma general, es necesario.

Ejercicio 1.2.7. Diga cuales de las siguientes operaciones pueden realizarse únicamente en modo supervisor, o modo kernel:

- (a) Consultar la hora del sistema.

No es necesario, ya que no afecta al resto de procesos.

(b) Cambiar la fecha del sistema.

Sí es necesario el modo kernel; ya que otros procesos, como la alarma, dependen de la hora.

(c) Leer una pista/sector de un disco magnético.

Sí es necesario, ya que es una operación de muy bajo nivel y actúa directamente el SO.

(d) Generar una interrupción software.

Las interrupciones software se pueden generar desde ambos modos, tanto modo kernel como modo usuario.

(e) Generar una interrupción hardware.

En este caso, al estar referidas a dispositivos hardware externos como podrían ser los de Entrada/Salida, tenemos que estos pueden lanzar la excepción independientemente del modo en el que se esté.

(f) Modificar la dirección de un vector de la tabla de vectores de interrupción.

En este caso es claramente necesario el modo kernel, ya que si se modificase de forma inadecuada no se podría llamar correctamente a dicha interrupción.

(g) Deshabilitar las interrupciones.

En este caso también es evidente la necesidad de hacerlo desde el modo kernel.

Ejercicio 1.2.8. En el caso de un ordenador que se vaya a usar únicamente para un único usuario, ¿qué interés puede tener la existencia de los modos de funcionamiento supervisor/usuario?

El principal interés de esta separación del funcionamiento del sistema en dos modos reside en que el usuario puede, de cierta manera, estar tranquilo, en tanto que sabe que no es capaz de hacer nada que resulte crítico para el correcto funcionamiento del sistema (como borrar archivos que contengan información sobre el arranque, por ejemplo).

Además, de ser el ordenador multiprogramado, es necesario que los procesos no interfieran entre ellos y que no se perjudiquen mutuamente. El modo kernel en este caso se encarga de que, por ejemplo, un proceso no pueda modificar la dirección de un vector en la tabla de vectores de interrupciones y así no perjudique al resto de procesos.

Así, con esta separación ganamos en seguridad y control para nuestro sistema.

Ejercicio 1.2.9. Cuestiones sobre procesos, y asignación de CPU:

1. ¿Es necesario que lo último que haga todo proceso antes de finalizar sea una llamada al sistema para finalizar? ¿Sigue siendo esto cierto en sistemas monoprogramados?

Sí, es necesario ya que, en primer lugar, se debe realizar un cambio de contexto para que el proceso finalizado pase a dicho estado, mientras que un estado que se encontrase en “Preparándose” pase ahora al modo de ejecución.

Además, también es necesario para que se libere la memoria que ocupaba el programa finalizado, para así poder cargar en memoria un número mayor de programas.

En un sistema monoprogramado, si no se avisa al terminar un programa mediante una llamada al SO; entonces el procesador no va a pasar a otro proceso en ningún momento al no saber si se ha liberado la memoria correspondiente.

2. Cuando el controlador de un dispositivo produce una interrupción ¿se produce necesariamente un cambio de contexto?, ¿y cuando se produce una llamada al sistema?

En ambos casos, se realizará un cambio de modo a superusuario, ya que el sistema operativo se tendrá que encargar de gestionar tanto de la interrupción como de la llamada al sistema.

No obstante, no necesariamente ocurrirá un cambio de contexto; de hecho, esta ventaja de poder ejecutar rutinas del SO sin un cambio de contexto en el proceso ejecutándose es muy importante a la hora del diseño de ciertos tipos de Sistemas Operativos (aquellos que no están totalmente implementados con procesos). Sin embargo, cabe la posibilidad de que la rutina de tratamiento de la interrupción o llamada al sistema explícitamente ejecute un cambio de contexto.

3. Cuando un proceso se bloquea, ¿deberá encargarse él directamente de cambiar el valor de su estado en el descriptor de proceso o PCB?

No. El bloqueo se deberá a alguna interrupción o llamada al sistema, por lo que el SO se encargará de realizar el cambio de contexto. En este cambio, se actualizarán los estados de los procesos implicados (el que pasa a “Bloqueado” y el que pasa a “Ejecutándose”) y se modificará por tanto el PCB.

4. Sea un proceso que cambia de Ejecutándose a Bloqueado, ¿puede este cambio provocar un cambio de estado en otros procesos? Si es así, ¿en qué casos?

Sí, ya que se llevará a cabo un cambio de contexto y uno de los procesos que estaba en estado de “Preparado” pasará a ejecutarse, siempre y cuando los recursos estén disponibles (por ejemplo, si todos los procesos preparados requieren de la impresora, y esta está ocupada, es necesario esperar).

5. Idem para el cambio de estado Bloqueado a Ejecutable.

Al realizarse el cambio de estado de un proceso de “Bloqueado” a “Preparado”, es posible que simplemente entre en la cola de los procesos preparados para su ejecución y que, por tanto, no provoque cambios en los demás. No obstante, si el planificador de procesos decide que el proceso que ha entrado tiene una prioridad más alta que el que se estaba ejecutando, es posible que pase este de “Preparado” a “Ejecutándose”, mientras que el que se estaba ejecutando y era menos importante pase a estado “Bloqueado”.

Ejercicio 1.2.10. En los primeros ordenadores, cada byte de datos leído o escrito, era manejado directamente por la CPU (es decir, no existía DMA Acceso Directo

a Memoria). ¿Qué implicaciones tenía esta organización para la multiprogramación?

Al no disponer de un módulo independiente a la CPU que gestionase las operaciones en Entrada/Salida, el procesador estaría ocupado durante todo el desarrollo de éstas, que por norma general suelen ser muy lentas.

Por tanto, en el caso de un sistema monoprogramado, durante el tiempo de ejecución de esta entrada y salida de datos el procesador estaría completamente ocioso.

En el caso de la multiprogramación, aunque se podrían ejecutar otros procesos durante la lectura, en ese tiempo se dispondría de menos capacidad de la CPU; por lo que la cantidad de programas que se podrían ejecutar haciendo uso de multiprogramación se vería bastante limitada. Por tanto, la principal ventaja de la multiprogramación se vería muy limitada.

Ejercicio 1.2.11. ¿Por qué no es el intérprete de órdenes (shell) parte del propio sistema operativo? ¿Qué ventajas aporta el no serlo?

La independencia del intérprete de órdenes respecto al SO supone diferentes ventajas, ya que siempre se busca minimizar el SO a lo exclusivamente esencial por diferentes motivos. Algunos de los beneficios de reducir el SO al mínimo son:

- Un fallo en la ejecución del shell no supondrá un fallo a nivel global en el SO, por lo que este se podrá encargar de gestionar dicho fallo. Los errores del sistema operativo suelen ser fatales, ya que no son de fácil gestión por él mismo.
- Al cargar lo mínimo posible en el sistema operativo, tenemos que al arrancar el ordenador este proceso de ejecuta de una manera más rápida y simple. Por tanto, esto aporta velocidad, ligereza y simpleza.
- Aporta versatilidad, ya que se puede actualizar el shell sin necesidad de actualizar simultáneamente el SO.
- El shell no puede realizar operaciones en modo kernel; algo que sí puede realizar el SO y podría provocar errores si no se gestiona adecuadamente.

Ejercicio 1.2.12. Para cada una de las llamadas al sistema siguientes, especificar y explicar si su procesamiento por el sistema operativo implica un cambio de contexto:

1. Crear un proceso.

Se produce un cambio de contexto cuando un proceso que estaba ejecutándose pasa a otro estado, mientras que uno que estaba preparado pasa a ejecutarse.

En este caso, si solo se crea el proceso no es necesario el cambio de contexto, ya que pasará a estado preparado pero no empezará a ejecutarse.

No obstante, en el caso puntual en el que al crearlo tenga más prioridad que algún proceso que se esté ejecutando, ese proceso pasará a estar preparado; mientras que el que se acaba de crear pasaría a ejecutarse. En este caso, por tanto, sí se produciría el cambio de contexto.

2. Abortar un proceso, es decir, terminarlo forzosamente.

Sí, en este caso se produce un cambio de contexto, ya que al terminarlo forzosamente se liberan los recursos que estaban siendo ocupados por él. Un proceso que estuviese en estado de “Preparado” pasaría a ejecutarse gracias a esa liberación de recursos. Por tanto, se produciría un cambio de contexto.

3. Suspender o bloquear un proceso.

Sí, ya que el proceso pasaría de estado “Ejecutándose” a otro estado, por lo que se liberarían recursos que serían empleados para ejecutar un proceso que estaba preparado en espera. Por tanto, se produce un cambio de contexto.

4. Reanudar un proceso (inverso al caso anterior).

También se produce un cambio de contexto, ya que un proceso pasa de estar “Preparado” a ejecutarse.

5. Modificar la prioridad de un proceso.

No necesariamente implica un cambio de contexto.

Si el cambio en la prioridad de un proceso que se esté ejecutando o de uno que esté en espera implica que la prioridad del que está en espera sea mayor que la del que se está ejecutando, entonces la CPU desasignará los recursos del proceso que se estaba ejecutando y los asignará al que estaba preparado. Es decir, un proceso que estaba en ejecución a pasado a estar preparado, mientras que uno preparado ha pasado a ejecutarse, por lo que se ha producido un cambio de contexto.

No obstante, si el cambio en la prioridad no implica que la prioridad de un proceso en espera sea mayor que la de los que se están ejecutando, entonces no se producen cambios de contexto.

Ejercicio 1.2.13. ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo?

Depende del motivo por el cual estén bloqueados. En el caso de que estén bloqueados por motivos distintos, no tiene sentido ya que volverán a ejecutarse en cuanto termine el motivo por el cual bloqueados.

En el caso de que estén bloqueados por el mismo motivo, sí tiene sentido; ya que cabe la posibilidad de que un proceso sea más importante que otro y, por tanto, sea necesario que se ejecute en cuanto se termine el motivo por el cual ha sido detenido. Por ejemplo, si varios procesos están bloqueados en espera de la impresora, tiene sentido ordenarnos para determinar cuál va a poder usarla antes.

Ejercicio 1.2.14. ¿Por qué se utilizan potencias de dos para los tamaños de página, número de páginas en el espacio lógico de un proceso, y números de marcos de página?

Se utilizan potencias de 2 porque un ordenador trabaja internamente con números en sistema binario. Además, las potencias de dos son siempre divisibles en potencias de 2 de menor orden, por lo que se puede dividir una página en páginas de menor tamaño sin que quede espacio sin utilizar o fragmentado, evitando así la fragmentación externa.

Ejercicio 1.2.15. Sitúese en un sistema paginado, en donde la memoria real tiene un tamaño de 16 Mbytes, una dirección lógica ocupa 32 bits, de los cuales los 22 de la izquierda constituyen el número de página, y los 10 de la derecha el desplazamiento dentro de la página. Según lo anterior,

1. ¿Qué tamaño tiene cada página?

El tamaño de página es $2^{10} B = 1024 B = 1 KB$. Es en bytes porque la unidad mínima en gestión de memoria es el byte.

2. ¿En cuántos marcos de página se divide la memoria física?

Como el tamaño del marco de página es el mismo que el tamaño de la página, tengo que el tamaño del marco de página es $1024 B$.

Por tanto, el número de marcos de página es:

$$\frac{16 MB}{1024 B} = \frac{16 \cdot 2^{20} B}{1 \cdot 2^{10} B} = 16 \cdot 2^{10} = 2^{14} \text{ marcos de página.}$$

3. ¿Qué tamaño deberá tener el campo **Número de Marco** de la Tabla de Páginas?

Como tenemos 2^{14} marcos de página, necesitamos 14 bits para representar todos. Por tanto, el tamaño ha de ser 14 bits.

4. Además de dicho campo, suponga que la Tabla de Páginas tiene los siguientes campos con los siguientes valores:

Protección: 1 bit (1=Sólo se permite leer; 0=Cualquier tipo de acceso).

¿Cuál es el tamaño de la Tabla de Páginas para un proceso cuyo espacio de memoria lógico es de $103 KB$?

Como el tamaño de cada página es de $1 KB$, necesitamos 103 páginas en total. Como el tamaño de cada página es $14 + 1$ bits, tenemos que el tamaño de la tabla de páginas es:

$$103 \times (14 + 1) = 1545 \text{ bits}$$

Ejercicio 1.2.16. Suponga que la tabla de páginas para el proceso actual se parece a la de la figura. Todos los números son decimales, la numeración comienza en todos los casos desde cero, y todas las direcciones de memoria son direcciones en bytes. El tamaño de página es de 1024 bytes.

N. Pág. Virtual	N. Marco de Página
0	4
1	7
2	1
3	2
4	10
5	0

Tabla 1.6: Tabla de Páginas para el proceso del ejercicio 1.2.16.

¿Qué direcciones físicas corresponderán con cada una de las siguientes direcciones lógicas del proceso?

1. 999

Calculamos en primer lugar el número de página, el marco y el desplazamiento. Tenemos que:

$$999 = 1024 \cdot 0 + 999$$

Como el cociente de $\frac{999}{1024}$ es 0, tenemos que el número de página es el 0. Equivalentemente, el marco de página es el 4.

Como el resto de $\frac{999}{1024}$ es 999, tenemos que el desplazamiento es 999.

Por tanto, la dirección física es:

$$\begin{aligned} \text{Dir. Física} &= \text{Num. Marco} \times \text{Tam. Marco} + \text{Desplazamiento} \\ &= 4 \cdot 1024 + 999 = 5095 \end{aligned}$$

Además, la dirección lógica sería (0, 999).

2. 2121

Calculamos en primer lugar el número de página, el marco y el desplazamiento. Tenemos que:

$$2121 = 1024 \cdot 2 + 73$$

Como el cociente de $\frac{2121}{1024}$ es 2, tenemos que el número de página es el 2. Equivalentemente, el marco de página es el 1.

Como el resto de $\frac{2121}{1024}$ es 73, tenemos que el desplazamiento es 73.

Por tanto, la dirección física es:

$$\begin{aligned} \text{Dir. Física} &= \text{Num. Marco} \times \text{Tam. Marco} + \text{Desplazamiento} \\ &= 1 \cdot 1024 + 73 = 1097 \end{aligned}$$

Además, la dirección lógica sería (2, 73).

3. 5400

Calculamos en primer lugar el número de página, el marco y el desplazamiento. Tenemos que:

$$5400 = 1024 \cdot 5 + 280$$

Como el cociente de $\frac{5400}{1024}$ es 5, tenemos que el número de página es el 5. Equivalentemente, el marco de página es el 0.

Como el resto de $\frac{5400}{1024}$ es 280, tenemos que el desplazamiento es 280.

Por tanto, la dirección física es:

$$\begin{aligned} \text{Dir. Física} &= \text{Num. Marco} \times \text{Tam. Marco} + \text{Desplazamiento} \\ &= 0 \cdot 1024 + 280 = 280 \end{aligned}$$

Además, la dirección lógica sería (5, 280).

Ejercicio 1.2.17. ¿Qué tipo de fragmentación se produce en un sistema de gestión de memoria paginado? ¿Qué decisiones de diseño se pueden tomar para minimizar dicho problema, y cómo afectan estas decisiones al comportamiento del sistema?

La paginación se realiza en páginas de la misma longitud. Al ejecutar cada proceso, se emplean tantas páginas como sean necesarias según el tamaño del proceso. El problema en este caso es que si el tamaño del proceso no es múltiplo del tamaño de página, se desperdicia memoria, ya que la última página asociada a dicho proceso no estará completa. Esto se denomina **fragmentación interna**.

Para minimizar dicho problema, se podría emplear un tamaño de página menor, para así conseguir que la memoria desperdiciada en cada proceso fuese menor. No obstante, esto provocaría que la tabla de páginas aumentase de tamaño, disminuyendo así la eficiencia del programa.

Ejercicio 1.2.18. Suponga que un proceso emite una dirección lógica igual a 2453 y que se utiliza la técnica de paginación, con páginas de 1024 palabras:

1. Indique el par de valores (número de página, desplazamiento) que corresponde a dicha dirección.

Tenemos que:

$$2453 = 1024 \cdot 2 + 405$$

Como el cociente de $\frac{2453}{1024}$ es 2, y el resto (desplazamiento) es 405, tenemos que la dirección lógica sería (2, 405).

2. ¿Es posible que dicha dirección lógica se traduzca en la dirección física 9322? Razónelo.

Supongamos que sí. Como el desplazamiento de la memoria lógica es 405, significa que ese marco de página empieza en la dirección física $9322 - 405 = 8917$. En paginación, tenemos que todos los marcos tienen el mismo tamaño. Como 8917 no es múltiplo del tamaño del marco (1024), tenemos que no es posible un marco de página que empiece en ese valor.

Ejercicio 1.2.19. Suponga que tenemos 3 procesos ejecutándose concurrentemente en un determinado instante. El sistema operativo utiliza un sistema de memoria con paginación. Se dispone de una memoria física de 131072 bytes (128K). Sabemos que nuestros procesos al ser ejecutados tienen los parámetros que se muestran en la tabla.

Proceso	Código	Pila	Datos
A	20480	14288	10240
B	16384	8200	8192
C	18432	13288	9216

Tabla 1.7: Parámetros de los procesos del ejercicio 1.2.19.

Los datos indican el tamaño en bytes de cada uno de los segmentos que forman parte de la imagen del proceso. Sabiendo que una página no puede contener partes de dos segmentos diferentes (pila, código o datos), hemos de determinar el tamaño

de página que debería utilizar nuestro sistema y se barajan dos opciones: páginas de 4096 bytes (4K) o páginas de 512 bytes (1/2K). Se pide:

1. ¿Cuál sería la opción más apropiada, $4096 \text{ } B = 4 \text{ KB}$ o $512 \text{ } B = 0,5 \text{ KB}$? Justifica totalmente la respuesta mostrando todos los cálculos que has necesitado para llegar a dicha conclusión.

Calculamos el número de marcos de páginas que serían necesarios para cada opción:

$$\frac{131072 \text{ } B}{4096 \text{ } B} = 32 \text{ marcos de página} \quad \frac{131072 \text{ } B}{512 \text{ } B} = 256 \text{ marcos de página}$$

Suponiendo la opción de $4096 \text{ } B$, el número de páginas de cada parte y cada proceso sería:

Proceso	Código	Pila	Datos
A	5	4	3
B	4	3	2
C	5	4	3

En total serían necesarias 33 páginas, pero solo disponemos de 32 de ellas.

Suponiendo la opción de 512, el número de páginas de cada parte y cada proceso sería:

Proceso	Código	Pila	Datos
A	40	28	20
B	32	17	16
C	36	26	18

En total serían necesarias 233 páginas, por lo que sí nos caben los 3 procesos en memoria física.

2. ¿Cuál es el formato de cada entrada de la Tabla de Páginas con el tamaño de página elegido? Justifica el tamaño de los campos con direcciones. Puedes añadir los bits que consideres necesarios para el buen funcionamiento del sistema indicando para qué van a ser utilizados.

Como hay 256 marcos de página, necesitamos 8 bits para codificar el marco de página.

Respecto a los bits adicionales, sería conveniente incluir 1 bit de protección² y 2 bits para codificar el tipo de página (Código, Pila o Datos).

Por tanto, cada entrada de la Tabla contendría el marco de página (8 bits) y los bits adicionales (3 bits).

²Para codificar si es de lectura o escritura.

3. ¿Cuántas Tablas de Páginas habrá en este sistema? ¿Cuántas entradas hay en cada tabla de páginas (filas)?

Habrá una tabla de página por proceso. Por tanto, habrá 3 Tablas de Páginas.

Para el proceso A , serían necesarias $40 + 28 + 20 = 88$ páginas, por lo que se necesitarían 88 entradas.

Para el proceso B , serían necesarias $32 + 17 + 16 = 65$ páginas, por lo que se necesitarían 65 entradas.

Para el proceso C , serían necesarias $36 + 26 + 18 = 80$ páginas, por lo que se necesitarían 80 entradas.

Ejercicio 1.2.20. En la gestión de memoria en un sistema paginado, ¿qué estructura/s de datos necesitará mantener el Sistema Operativo para administrar el espacio libre?

La estructura de datos que emplea el SO en el caso de un sistema paginado es la Tabla de Páginas.

Ejercicio 1.2.21. Estamos trabajando con un sistema operativo que emplea una gestión de memoria paginada. Cada página tiene un tamaño de 2.048 bytes. La memoria física disponible para los procesos es de 8 MBytes. Suponga que primero llega un proceso que necesita 31.566 posiciones de memoria (o bytes) y, después, llega otro proceso que consume 18.432 posiciones cuando se carga en memoria. Se pide calcular la fragmentación interna provocada en cada proceso.

Respecto al proceso A , tenemos que el cociente de $\frac{31566}{2048}$ es 15, por lo que se emplearían 15 páginas completas. Además, como el resto es 846, se emplearían 846 B de la página 16, y de esta página se desperdiciarían $2048 - 846 = 1202$ B .

Respecto al proceso B , como la división de $\frac{18432}{2048} = 9$ es una división entera, no se produce fragmentación interna, ya que se ocupan 9 páginas completamente.

Ejercicio 1.2.22. Considere la siguiente tabla de segmentos:

Segmento	Dirección base	Longitud
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Tabla 1.8: Tabla de segmentos del ejercicio 1.2.22.

¿Qué direcciones físicas corresponden a las direcciones lógicas (*nº de segmento, desplazamiento*) siguientes? Si no puede traducir alguna dirección lógica a física, explique el por qué.

1. 0, 430

$$\begin{aligned}\text{Dir. Física} &= \text{Dir. Base} + \text{Desplazamiento} \\ &= 219 + 430 = 649\end{aligned}$$

2. 1, 10

$$\text{Dir. Física} = 2300 + 10 = 2310$$

3. 3, 400

$$\text{Dir. Física} = 1327 + 400 = 1727$$

4. 4, 112

$$\text{Dir. Física} = 1952 + 112 = \text{ERROR}$$

En este caso no es posible, ya que el desplazamiento (112) es mayor o igual que la longitud del segmento correspondiente (96).

Ejercicio 1.2.23. ¿Qué cambio de contexto tardará menos y por qué?

(a) El producido entre dos hebras del mismo proceso.

(b) El producido entre dos hebras de distintos procesos.

Tarda menos en producirse el cambio de contexto entre hebras del mismo proceso, ya que estas comparten el PCB y las direcciones de memorias asociadas, ya que estas no dependen de la hebra sino del proceso. Por tanto, al tener que cambiar menos información, el cambio de contexto es más rápido al ser del mismo proceso.

1.3. Compilación y Enlazado de Programas

Ejercicio 1.3.1. Un procesador (CPU) puede interpretar y ejecutar directamente las instrucciones de un programa en:

- (a) Lenguaje de alto nivel de tipo intérprete.
- (b) Lenguaje ensamblador o en lenguaje máquina, cualquiera de los dos.
- (c) Sólo lenguaje máquina.**
Ya que la CPU no entiende ninguna otra cosa; de hecho, la existencia del propio lenguaje máquina no tendría sentido si este no fuese el caso.
- (d) En pseudocódigo o en lenguaje ensamblador.

Ejercicio 1.3.2. ¿Es lo mismo un token que un lexema? Muestre algún ejemplo.

No, no es lo mismo. Un token relaciona a un conjunto de lexemas que tienen la misma función sintáctica.

Por ejemplo, el token IDENT que contiene a todos los identificadores posibles de variables puede estar formado por las combinaciones de letras y números. En este caso, los lexemas asociados a ese token podrían ser `i`, `var` o `lado1`, por ejemplo.

No obstante, para el caso de las palabras reservadas se tiene que hay un token por cada palabra reservada.

Ejercicio 1.3.3. ¿El compilador es la única utilidad necesaria para generar un programa ejecutable en una computadora?

No, también son necesarias otras. De hecho, con el compilador tan solo se pasa del código fuente a código ensamblador.

Otras utilidades necesarias para generar un programa ejecutable son el ensamblador para pasar a código máquina, el enlazador, o el preprocesador.

Ejercicio 1.3.4. El análisis léxico es una etapa de la compilación cuyo objetivo es:

- (a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.**

La segunda parte sí sería correcta, pero el análisis léxico no se encarga de extraer la estructura; ese es el objetivo de la fase de análisis.

- (b) Descomponer el programa fuente en sus componentes léxicos (tokens).**

- (c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.**

Esta fase es el análisis semántico.

- (d) Sintetizar el programa objeto.**

Esto pertenece a la fase de síntesis.

Ejercicio 1.3.5. El análisis sintáctico es una etapa de la compilación cuyo objetivo es:

- (a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.
 Es esta la única función del analizador sintáctico, el comprobar la correcta estructura de cada una de las sentencias del programa.
- (b) Descomponer el programa fuente en sus componentes léxicos (tokens).
- (c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.
- (d) Sintetizar el programa objeto.

Ejercicio 1.3.6. Para el siguiente código que aparece a la izquierda en lenguaje C++ (fichero `test.cpp`, código fuente 1), indique el nombre de la fase en la que el compilador produce el mensaje de error que aparece a la derecha y explique la naturaleza del mismo:

```

1 int main (void)
2 {
3     int i;
4     char* j;
5
6     j = i;
7
8     if (i == 0)
9         i += ;
10
11    ;
12
13     return 0;
14 }
```

Código fuente 1: Fichero `test.cpp`

1. `test.cpp:9: error: expected primary-expression before ‘;’ token`

Tenemos que se trata de un error sintáctico; ya que aunque todos los lexemas de la línea 9 son válidos (tienen un token asociado) no se puede llegar a esa línea con las reglas de producción. Esto se debe a que falta el *r-value*, es decir, a la derecha del operador binario `+=` falta un valor (dada la naturaleza del mismo precisada en su nombre).

2. `test.cpp:6: error: invalid conversion from ‘int’ to ‘char*’`

Es un error semántico. Todas las secuencias de la línea tienen asociadas un token preciso y además la estructura de la sentencia (sintaxis) es correcta; sin embargo, intentar hacer una conversión de tipo entero a tipo puntero a

carácter no tiene sentido en el lenguaje de programación: esta sentencia es irrealizable y se produce el error mencionado.

3. `test.cpp:11: error: stray '\302' in program`

Se trata de un error léxico, ya que el carácter `\302` no tiene ningún token asociado (no es un identificador, operador, expresión aritmética, palabra clave, etc.) pues no pertenece al alfabeto de la gramática de C++.

Ejercicio 1.3.7. Muestre un ejemplo a partir de una sentencia en lenguaje C++ en la que un error léxico origine un error sintáctico derivado y otro error léxico que no derive en error sintáctico.

Consideramos en primer lugar la línea `int mai?n(){` donde suponemos que en la siguiente línea continua el código. Tenemos que se trata de un error léxico, ya que no reconoce `mai?n` como un identificador válido. Además, dará error sintáctico por la falta de la función `main`.

Consideramos ahora la línea `int ñum;.` Tenemos que se trata de un error léxico ya que la letra `ñ` no pertenece al alfabeto de C++, por lo que no encontrará un token asociado. No obstante, no genera un error sintáctico ya que la sintaxis es correcta, al especificar el tipo, el nombre de la variable y terminar con el `;`.

Observación. Estos casos no obstante no son normales, ya que al detectar un error se detiene la compilación. No se ha corregido por tanto en clase.

Ejercicio 1.3.8. Muestre un ejemplo a partir una sentencia de en lenguaje C++ en la que un error léxico origine un error sintáctico y semántico derivados y otro error léxico que no los derive.

Un ejemplo sería declarar una variable de la forma `int 1variable = 2; int var = 1variable;` El identificador no es válido y provocaría un error léxico , que derivará en uno sintáctico y en uno semántico, pues se utiliza abajo una variable no declarada.

Observación. Estos casos no obstante no son normales, ya que al detectar un error se detiene la compilación. No se ha corregido por tanto en clase.

Ejercicio 1.3.9. ¿Sería siempre posible realizar la depuración de un archivo objeto? Razona la respuesta.

No, ya el fichero objeto no es ejecutable. Además, al no haber sido enlazado no contiene la cabecera en la que, por ejemplo, figura dónde empieza el programa, por lo que el depurador no sabría por donde empezar.

Como caso excepcional, se podría depurar una parte concreta del programa, pero no es útil y en la práctica no se considera este caso. En la mayoría de depuradores no es ni siquiera posible.

Ejercicio 1.3.10. Dado un programa escrito en lenguaje ensamblador de una arquitectura concreta, ¿sería directamente interpretable ese código por esa computadora? En caso contrario ¿qué habría que hacer?

No, para que fuese interpretable por parte de la computadora necesitaría traducirse el código a código máquina. De esta traducción se encargaría el ensamblador.

Ejercicio 1.3.11. ¿Sería necesario usar siempre el enlazador para obtener un programa ejecutable?

Sí, ya que si no se enlaza la cabecera del archivo no se actualizará, y por tanto no se tendrá la instrucción con la que se desea empezar.

Ejercicio 1.3.12. Dado un único archivo objeto, ¿podría ser siempre un programa ejecutable y correcto simplemente añadiendo la información de cabecera necesaria?

En general no, ya que se necesitaría el uso del enlazador porque, aunque haya un solo archivo objeto, puede que haya bibliotecas externas que tengan que ser cargadas.

Ejercicio 1.3.13. Dado un programa ejecutable que requiere de una biblioteca dinámica, ¿por qué no es necesario recompilar el código fuente de dicho programa si se modifica la biblioteca?

Las bibliotecas dinámicas se integran en el programa en tiempo de ejecución. Por tanto, como el programa compilado no contiene a la biblioteca, no es necesario recompilar para guardar los cambios. En el caso de hacer uso de bibliotecas estáticas, el código obsoleto sí que estaría incluido en el código máquina del programa ejecutable, y por tanto debería recompilarse para actualizarse.

En único caso en el que la actualización de la biblioteca dinámica haría necesario volver a compilar el programa es que se cambiase las cabeceras de las funciones.

Ejercicio 1.3.14. Indique en qué fase del proceso de traducción y ejecución de un programa se realizará cada una de las siguientes tareas:

1. Enlazar una biblioteca estática: Fase de enlazado.
2. Eliminar los comentarios del código fuente: En la compilación, en la fase de análisis léxico.

También es válido indicar que ocurre en el preprocessado, dependiendo de la fuente consultada.

3. Mensaje de error de que una variable no ha sido declarada: Fase de análisis semántico.

En particular, seguramente ocurrirá porque no la encontrase en la tabla de símbolos (cualquier cuestión relacionada con el ámbito será asunto de esta fase).

4. Enlazar una biblioteca dinámica: Fase de ejecución.

Ejercicio 1.3.15. Indique en qué fase o fases del proceso de compilación de un lenguaje de programación de alto nivel se detectarían los siguientes errores:

1. Una variable no está definida: Fase de análisis semántico.
De nuevo, cuestión de ámbito dentro del programa.
2. Aparece un carácter o símbolo no esperado: Fase de análisis léxico.
3. Aparecen dos identificadores consecutivos: Fase de análisis sintáctico.
4. Aparecen dos funciones denominadas bajo el mismo nombre: Fase de análisis semántico.

No obstante, dependiendo del lenguaje puede ser que no diese error, ya que la función puede estar sobrecargada.

5. Aparece el final de un bloque de sentencias pero no el inicio del mismo: Fase de análisis sintáctico.
6. Aparece un paréntesis cerrado y no se ha podido emparejar con su correspondiente paréntesis abierto: Fase de análisis sintáctico.
7. Una llamada a una función que no ha sido definida: Fase de análisis semántico.
8. En la palabra reservada `main` aparece un carácter extraño no esperado, por ejemplo `mai;ñ`: Fase de análisis léxico que derivará en error sintáctico.

Ejercicio 1.3.16. ¿Todo error sintáctico origina un error semántico? En caso contrario, demuéstrelo usando algún contrajejemplo.

No, y como contrajejemplo, sea el siguiente programa:

```
int num=7;
num += ;
```

Tenemos que hay un error sintáctico en la segunda línea al faltar el *r-value*, pero esto no genera ningún error semántico.

Observación. Estos casos no obstante no son normales, ya que al detectar un error se detiene la compilación. No se ha corregido por tanto en clase.

Ejercicio 1.3.17. Consideramos la siguiente gramática $G = \{V_T, V_N, S, P\}$ donde el axioma $S = \text{programa}$ y P contiene las siguientes reglas de producción:

- $\text{programa} \longrightarrow \{\text{lista_sentencias}\}$
- $\text{lista_sentencias} \longrightarrow \text{sentencia} \mid \text{sentencia} ; \text{lista_sentencias}$
- $\text{sentencia} \longrightarrow \text{identificador} = \text{expresión}$
- $\text{identificador} \longrightarrow \text{letra} \mid \text{identificador dígito} \mid \text{identificador letra}$
- $\text{letra} \longrightarrow a|b|c|\dots|z$
- $\text{dígito} \longrightarrow 0|1|2|\dots|9$
- $\text{expresión} \longrightarrow \text{identificador operador identificador} \mid \text{identificador}$
- $\text{operador} \longrightarrow +|-|*|/$

1. Obtener la lista de Tokens

En primer lugar, obtengo la lista de Tokens:

TOKEN	PATRÓN
LLAVE_AB	{
LLAVE_CE	}
PUNTO_Y_COMA	;
ASSIGN	=
IDENTIF	letra(letra dígito)*
OPERADOR	+ - * /

Tabla 1.9: Lista de Tokens de la gramática del ejercicio 1.3.17

2. Ver si las siguientes sentencias se pueden generar mediante esta gramática:

a) $\{8a = b + c\}$

El proceso de derivación es:

$$\begin{aligned} \text{programa} &\longrightarrow \{\text{lista_sentencias}\} \longrightarrow \{\text{sentencia}\} \longrightarrow \\ &\longrightarrow \{\text{identificador} = \text{expresión}\} \end{aligned}$$

Por tanto, tenemos que no es posible generar dicha sentencia, ya que un identificador ha de empezar por letra.

b) $\{\text{suma} + b = a * c\}$

El proceso de derivación es:

$$\begin{aligned} \text{programa} &\longrightarrow \{\text{lista_sentencias}\} \longrightarrow \{\text{sentencia}\} \longrightarrow \\ &\longrightarrow \{\text{identificador} = \text{expresión}\} \end{aligned}$$

Por tanto, tenemos que no es posible generar dicha sentencia, ya que un identificador no puede contener el operador +.

c) $\{\text{suma} = b * z\}$

El proceso de derivación es:

$$\begin{aligned} \text{programa} &\longrightarrow \{\text{lista_sentencias}\} \longrightarrow \{\text{sentencia}\} \longrightarrow \\ &\longrightarrow \{\text{identificador} = \text{expresión}\} \longrightarrow \\ &\longrightarrow \{\text{identificador} = \text{identificador operador identificador}\} \longrightarrow \\ &\longrightarrow \{\text{suma} = b * z\} \end{aligned}$$

1.4. Introducción a las Bases de Datos

Ejercicio 1.4.1. Se desea diseñar un esquema relacional de una base de datos para un centro de enseñanza que contenga información sobre los alumnos (dni, nombre, dirección . . .), las asignaturas (código de asignatura y nombre de esta se considera la información mínima) y las calificaciones que se obtienen en cada una de las mismas. Desarrollar un modelo ER del mismo y posteriormente reducirlo a tablas.

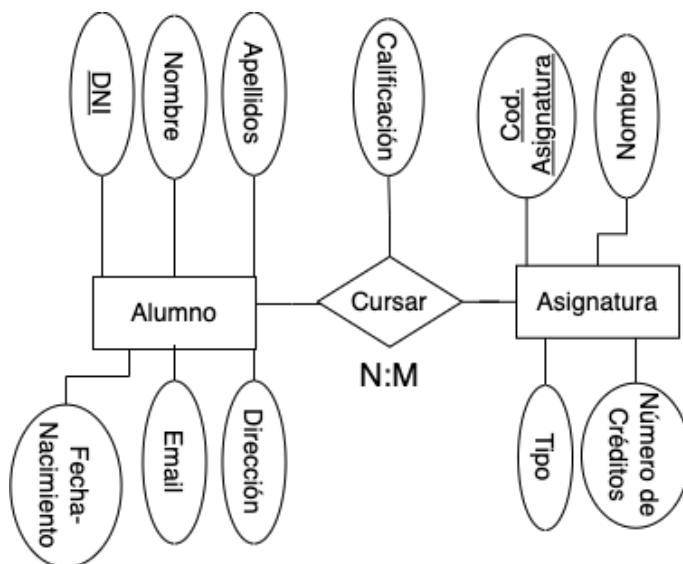


Figura 1.3: Modelo Entidad-Relación del ejercicio 1.4.1

Tendríamos tres tablas:

Alumno (DNI, Nombre, Apellidos, Dirección, Email, Fecha de Nacimiento)

Asignatura (Cód-Asignatura, Nombre, Tipo, Número de Créditos)

Cursar (DNI, Cód. Asignatura, Calificación)

Ejercicio 1.4.2. Se desea diseñar la base de datos de una biblioteca particular, de modo que para cada libro se deberá almacenar: su título, número de páginas, ISBN, materia, año de edición, editorial y autor o autores del mismo, para los que, además de su nombre, se recogerán los siguientes datos: dirección de correo electrónico, nacionalidad y fecha de nacimiento. Además, para cada editorial se deberá guardar su dirección, localidad y país. Teniendo en cuenta que se pueden añadir los campos que se consideren oportunos para poder relacionar convenientemente las distintas entidades del problema, realizar lo que se pide en cada uno de los siguientes apartados:

- Dibujar el esquema conceptual, utilizando el modelo entidad-relación.

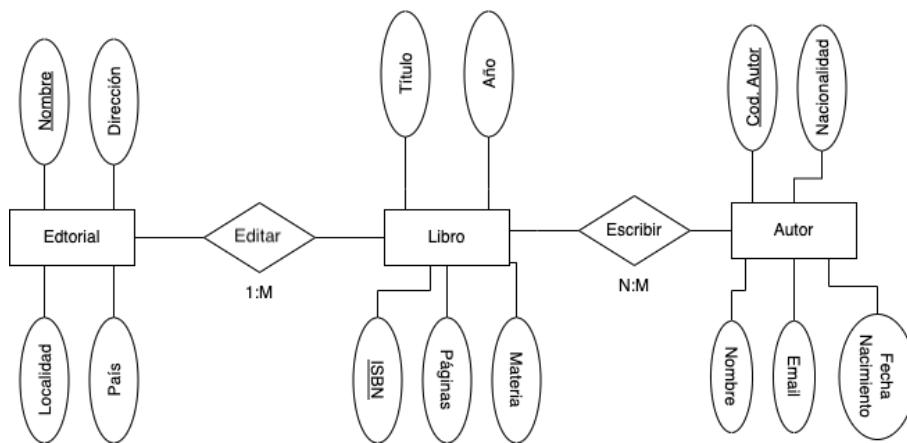


Figura 1.4: Modelo Entidad-Relación del ejercicio 1.4.2

- Obtener, a partir de lo realizado en el apartado anterior, las tablas que se tendrían que crear en un SGBD relacional, indicando qué campos compondrían cada tabla y cuál sería la clave primaria de cada una de ellas.

Tendríamos cuatro tablas:

Autor (Cod. Autor, Nombre, Email, FechaNacimiento, Nacionalidad)

Libro (ISBN, Título, Páginas, Materia, Año, NombreEd.)

Editorial (NombreEd., Direcciones, Localidad, País)

Escribir (Cod. Autor, ISBN)

Ejercicio 1.4.3. Suponga que la base de datos para una Universidad del ejercicio (1) considera además de la información sobre los alumnos y las asignaturas, las carreras que se pueden estudiar. Construir un modelo ER y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:

- Un alumno puede estar matriculado en muchas asignaturas.
- Una asignatura sólo puede pertenecer a una carrera.
- Una carrera puede tener muchas asignaturas.

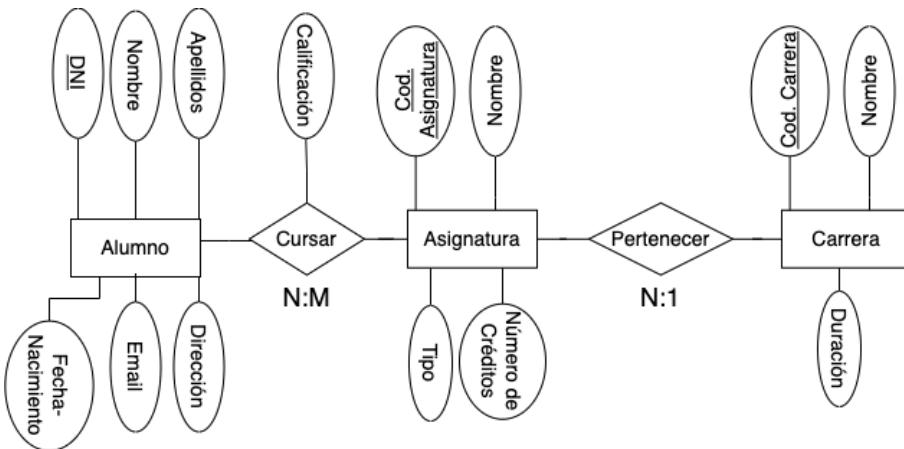


Figura 1.5: Modelo Entidad-Relación del ejercicio 1.4.3

Tendríamos tres tablas:

Alumno (DNI, Nombre, Apellidos, Dirección, Email, Fecha de Nacimiento)

Asignatura (Cód. Asignatura, Nombre, Tipo, Número de Créditos, Cód. Carrera)

Carrera (Cód. Carrera, Nombre, Duración)

Cursar (DNI, Cód. Asignatura, Calificación)

Observación. Se podría considerar que el nombre de titulación fuese el campo llave.

Ejercicio 1.4.4. Se desea diseñar una base de datos para un centro comercial organizado por secciones que contenga información sobre los clientes que han comprado algo, los trabajadores, el género que se oferta y las ventas realizadas. Construir un modelo ER y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:

- Existen tres tipos de trabajadores: gerentes, jefes y vendedores.
- Cada sección está gestionado por un gerente.
- Un determinado producto sólo se encuentra en una sección.
- Los jefes y vendedores sólo pueden pertenecer a una única sección.
- Un gerente tiene a su cargo a un cierto número de jefes y éstos a su vez a un cierto número de vendedores.
- Una venta la realiza un vendedor a un cliente y debe quedar constancia del artículo vendido. Sólo una unidad de cada artículo por apunte de venta.

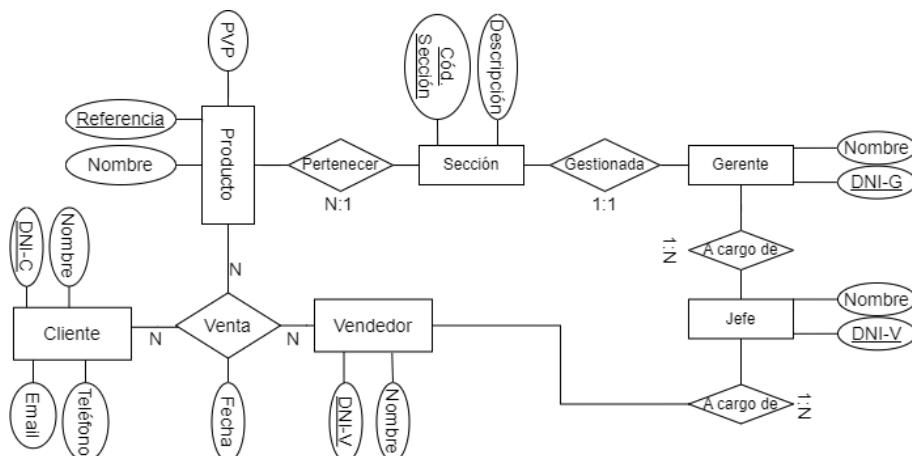


Figura 1.6: Modelo Entidad-Relación del ejercicio 1.4.4

Tendríamos las siguientes tablas:

Producto (Referencia, Nombre, PVP)

Sección (Cód. Sección, Descripción)

Gerente (DNI-G, Nombre, Cód-Sección)

Jefe (DNI-J, Nombre, DNI-G)

Vendedor (DNI-V, Nombre, DNI-J)

Comprador (DNI-C, Nombre, Teléfono, Email)

Venta (Fecha, DNI-C, DNI-V, Referencia)

Observación. En la tabla **Venta**, el campo Fecha sería clave o no si consideramos que la misma venta se puede repetir en fechas distintas.

Ejercicio 1.4.5. (EXAMEN) Se desea implementar un sistema de gestión de cupones de la ONCE en el que se registre cada venta de cupones. En el caso de que un cupón resulte premiado, el comprador ha de acudir a una Administración a cobrarlo. Tenga en cuenta las siguientes restricciones:

- Cada cupón es único.
- Cada vendedor tiene asociado su código de la seguridad social.
- Cada comprador puede comprar más de un cupón y a distintos vendedores.
- Existe una única administración por provincia.

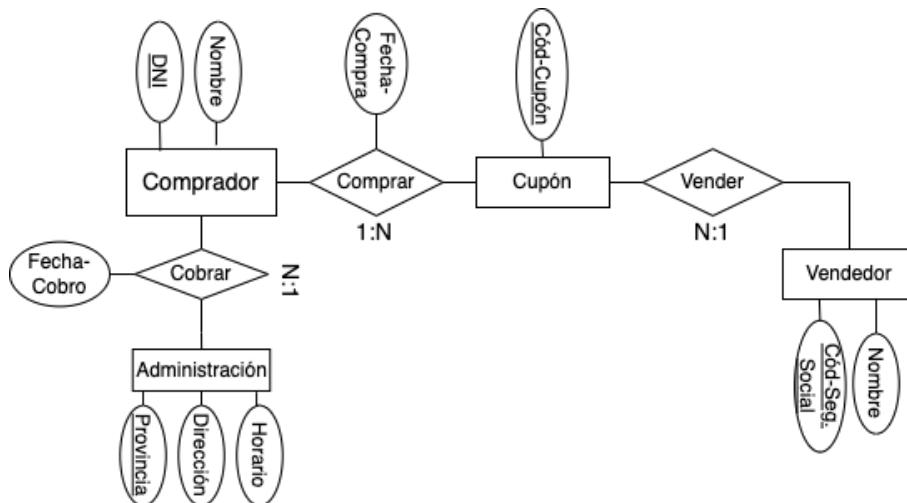


Figura 1.7: Modelo Entidad-Relación del ejercicio 1.4.5

Tendríamos las siguientes tablas:

Vendedor (Cód-Seguridad Social, Nombre)

Cupón (Cód-Cupón, Cód-Seguridad Social)

Comprador (DNI, Nombre)

Comprar (DNI, Cód-Cupón, Fecha-Compra)

Administración (Provincia, Dirección, Horario)

Cobrar (Provincia, DNI, Fecha-Cobro)

Ejercicio 1.4.6. Los profesores de la asignatura de Bases de Datos de una Escuela Universitaria deciden crear una base de datos que contenga la información de los resultados de las pruebas realizadas a los alumnos. Para realizar el diseño, se sabe que:

- Los alumnos están definidos por su número de matrícula, nombre y grupo al que asisten a clase.
- Dichos alumnos realizan dos tipos de pruebas a lo largo del curso académico:
 1. Exámenes escritos: cada alumno realiza varios a lo largo del curso, y se definen por el número de examen, el número de preguntas que consta y la fecha de realización (la misma para todos los que realizan el mismo examen). Evidentemente, es importante almacenar la nota de cada alumno por examen.
 2. Prácticas: se realiza un número indeterminado de ellas durante el curso académico, algunas serán en grupo y otras individuales. Se definen por un código de la práctica, título y el grado de dificultad. En este caso, los alumnos pueden examinarse de cualquier práctica cuando lo deseen, debiéndose almacenar la fecha y la nota obtenida.
- En cuanto a los profesores, únicamente interesa conocer (además de sus datos personales: DNI y nombre), quién es el que ha diseñado cada práctica, sabiendo que en el diseño de una práctica puede colaborar más de uno, y que un profesor puede diseñar más de una práctica. Interesa, además, la fecha en que ha sido diseñada cada práctica por el profesor correspondiente.

Para ello se pide:

1. Diseñar un modelo Entidad Relación para este caso.

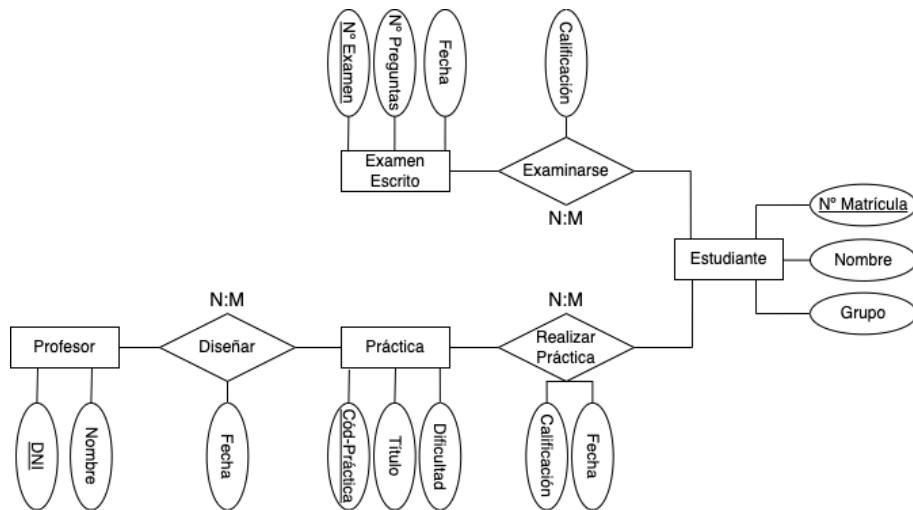


Figura 1.8: Modelo Entidad-Relación del ejercicio 1.4.6

2. Obtener, a partir de lo realizado en el apartado anterior, las tablas que se tendrían que crear en un SGBD relacional, indicando qué campos compondrían cada tabla y cuál sería la clave primaria para cada una de ellas.

Tendríamos las siguientes tablas:

Profesor (DNI, Nombre)

Diseñar (Cód-Práctica, DNI, Fecha)

Práctica (Cód-Práctica, Título, Dificultad)

Realizar Práctica (Cód-Práctica, Nº Matrícula, Fecha, Calificación)

Estudiante (Nº Matrícula, Nombre, Grupo)

Examinarse (Nº Matrícula, Nº Examen, Calificación)

Examen (Nº Examen, Nº de preguntas, Fecha)

Ejercicio 1.4.7. Se desea diseñar un esquema relacional de una base de datos para un colegio mayor que represente que los residentes (DNI, edad, nombre, apellidos) tienen un responsable asignado (DNI, nombre, apellidos, dirección, móvil, parentesco) desde una determinada fecha para que el director de la residencia pueda contactar con ellos en caso de incidencias. Un residente puede tener un único tutor asignado y un tutor solo puede ser responsable de un alumno.

Para ello se pide:

1. Diseñar un modelo Entidad Relación para este caso.

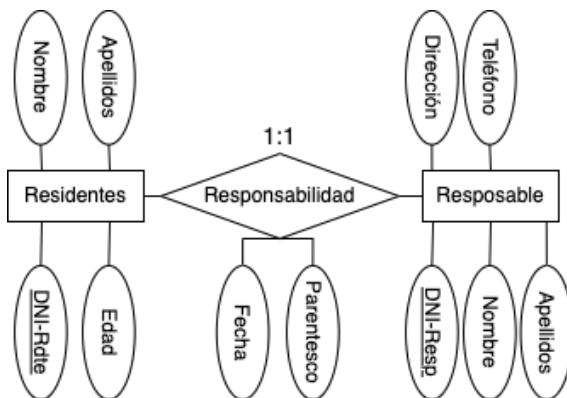


Figura 1.9: Modelo Entidad-Relación del ejercicio 1.4.7

2. Obtener, a partir de lo realizado en el apartado anterior, las tablas que se tendrían que crear en un SGBD relacional, indicando qué campos compondrían cada tabla y cuál sería la clave primaria para cada una de ellas.

Tendríamos las siguientes tablas:

Residente (DNI-Rdte, Nombre, Apellidos, Edad)

Responsable (DNI-Resp, Nombre, Apellidos, Dirección, Teléfono)

Responsabilidad (DNI-Rdte, DNI-Resp, Fecha, Parentesco)

Observación. Hemos supuesto que un residente tiene el mismo responsable durante toda su estancia, que no puede cambiarlo.

En el caso de que un residente pudiese cambiar de responsable, la cardinalidad sería 1:N, y el campo llave de la tabla responsabilidad sería DNI-Resp o ambos DNIs, a elección.