Tema 1. Sistema de Cómputo

Contenidos

1.1 Componentes de un Sistema de Cómputo.
- 1.1.1 Definiciones Básicas.
- 1.1.2 Registros del Procesador.
- 1.1.3 Ejecución de Instrucciones. Tipos de Instrucciones.

1.2 Capa Hardware.
- 1.2.1 Estructura de un Ordenador.
- 1.2.2 Técnicas de Comunicación de E/S.

1.3 El Sistema Operativo.

1.4 Utilidades del Sistema.

Objetivos

- Conocer los elementos principales de un Sistema de Cómputo.
- Disponer los elementos de la parte hardware.
- Conocer el software más próximo a la capa hardware: el Sistema Operativo.
- Conocer las principales utilidades software que se utilizan en un sistema de cómputo.

Bibliografía básica

[Prie06]	A. Prieto, A. Lloris, J.C. Torres, **Introducción a la Informática**, McGraw-Hill, 2006
[Stal05]	W. Stallings, **Sistemas Operativos, Aspectos Internos y Principios de Diseño (5ª Edición)**. Pearson Education, 2005
[Carr07]	J. Carretero, F. García, P. de Miguel, F. Pérez, **Sistemas Operativos (2ª Edición)**, McGraw-Hill, 2007

Definiciones Básicas [Prie06] (pp.1-7)

•**Informática** es una palabra de origen francés formada por la contracción de los vocablos INFORmación y autoMÁTICA.

- La Real Academia Española define la Informática como el conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores.

• **Computador, computadora u ordenador** es una máquina capaz de aceptar unos datos de entrada, efectuar con ellos operaciones lógicas y aritméticas y proporcionar la información resultante a través de un medio de salida. Todo ello mediante el control de un programa de instrucciones previamente almacenado en el propio computador.

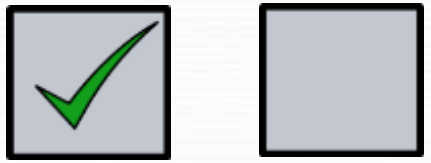
Considerando la definición de computador, se puede decir que la informática o ingeniería de los computadores (Computer Science) es el campo de conocimiento que abarca todos los aspectos del diseño y uso de los computadores.

Definición de Bit

- Bit: es el acrónimo de Binary digit. (dígito binario).
Un bit es un dígito del sistema de numeración binario.

- Unidad mínima de información

- Codifica información:
  - 1 bit: 0 ó 1
  - 2 bits: 00, 01, 10 ó 11
  - …



1 **0**

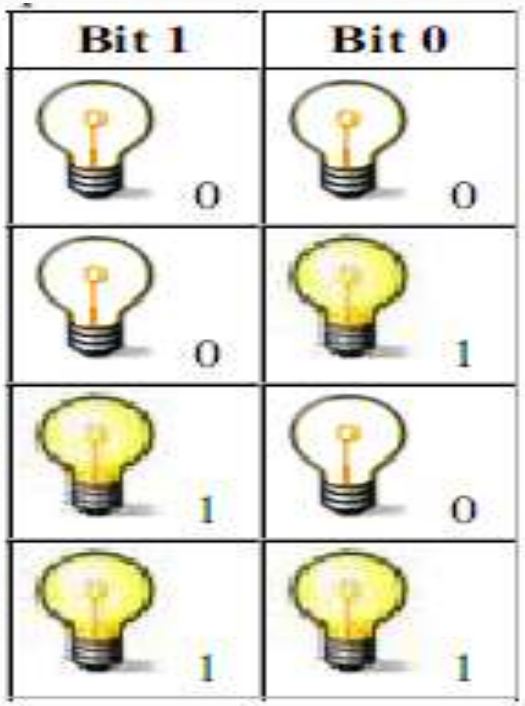
True **False**

Bit

- Con un bit podemos representar solamente dos valores, que suelen representarse como 0, 1.
- Para representar o codificar más información en un dispositivo digital, necesitamos una mayor cantidad de bits. Si usamos dos bits, tendremos cuatro combinaciones posibles:
  - 1 bit: 0 ó 1
  - 2 bits: 00, 01, 10 ó 11

Ejemplo

- 0 0 - Los dos están "apagados"
- 0 1 - El primero (de derecha a izquierda) está "encendido" y el segundo"apagado"
- 1 0 - El primero (de derecha a izquierda) está "apagado" y el segundo "encendido"
- 1 1 - Los dos están "encendidos"



Con estas cuatro combinaciones podemos representar hasta cuatro valores diferentes, como por ejemplo, los colores rojo, verde, azul y negro.

Bit

- Cuatro bits forman un nibble, y pueden representar hasta
  - $2^4 = 16$ valores diferentes;
- Ocho bits forman un octeto, y se pueden representar hasta
  - $2^8 = 256$ valores diferentes.

En general:

- Con un número n de bits pueden representarse hasta
  - **2^n valores diferentes.**

Múltiplos del bit y Unidades

- 1 Byte (B) = 8 bits (b)

- 1 Kilobyte (KB) = 2^{10} B
- 1 Megabyte (KB) = 2^{10} KB
- 1 Gigabyte (GB) = 2^{10} MB
- 1 Terabyte (TB) = 2^{10} GB
- 1 Petabyte (PB) = 2^{10} TB)

Sistemas de numeración

- Un sistema de numeración es un conjunto de símbolos y reglas que permiten representar datos numéricos. Los sistemas de numeración actuales son sistemas posicionales, que se caracterizan porque ***un símbolo tiene distinto valor según la posición que ocupa en la cifra***.

Cambio de base: binario, octal, hexadecimal [Prie06] (Apéndice A. pp.767)

- **Binario**: 000, 001, 010, 011, 100, 101, 110, 111
- **Octal**: 00, 01, 02, 03, 04, 05, 06, 07, 10, 11, 12, 13, 14, ...
- **Decimal**: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, …
- **Hexadecimal**: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, …

Sistema de numeracíon decimal

- El sistema de numeración que utilizamos habitualmente es el **decimal,** que se compone de diez símbolos o dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) a los que otorga un valor **dependiendo de la posición** que ocupen en la cifra: unidades, decenas, centenas, millares, etc.
- El valor de cada dígito está asociado al de una potencia de base 10, número que coincide con la cantidad de símbolos o dígitos del sistema decimal, y un exponente igual a la posición que ocupa el dígito menos uno, contando desde la derecha.

En el sistema decimal el número **528**, por ejemplo, significa:

5 centenas + 2 decenas + 8 unidades, es decir:

$$\mathbf{5*\ 10^2 + 2*10^1 + 8*\ 10^0}$$ o, lo que es lo mismo:

$$\mathbf{500 + 20 + 8 = 528}$$

Sistema de numeración binario

- El sistema de numeración binario utiliza sólo dos dígitos, el cero (0) y el uno (1).
- En una cifra binaria, cada dígito tiene distinto valor dependiendo de la posición que ocupe. El valor de cada posición es el de una potencia de base 2, elevada a un exponente igual a la posición del dígito menos uno.
- De acuerdo con estas reglas, el número binario 1011 tiene un valor que se calcula así:

$$1* \mathbf{2}^3 + 0* \mathbf{2}^2 + 1* \mathbf{2}^n + 1* \mathbf{2}^0 \text{ , es decir:}$$

$$8 + 0 + 2 + 1 = 11$$

- y para expresar que ambas cifras describen la misma cantidad lo escribimos así:

$$\mathbf{1011_2 = 11_{10}}$$

Conversión entre números decimales y binarios

Convertir un número decimal al sistema binario es muy sencillo: basta con realizar **divisiones sucesivas por 2** y escribir los restos obtenidos en cada división **en orden inverso** al que han sido obtenidos.

Por ejemplo, para convertir al sistema binario el número $\mathbf{77_{10}}$ haremos una serie de divisiones que arrojarán los restos siguientes:

- **77 : 2 = 38** Resto: **1**
- **38 : 2 = 19** Resto: **0**
- **19 : 2 = 9** Resto: **1**
- **9 : 2 = 4** Resto: **1**
- **4 : 2 = 2** Resto: **0**
- **2 : 2 = 1** Resto: **0**
- **1 : 2 = 0** Resto: **1**

y, tomando los restos en orden inverso obtenemos la cifra binaria:

$$\mathbf{77_{10} = 1001101_2}$$

Ejercicio

- Expresa, en código binario, los números decimales siguientes: 191, 25, 67, 99, 135, 276

El tamaño de las cifras binarias

- La cantidad de dígitos necesarios para representar un número en el sistema binario es mayor que en el sistema decimal.
  - En el ejemplo del párrafo anterior, para representar el número **77**, que en el sistema decimal está compuesto tan sólo por dos dígitos, han hecho falta siete dígitos en binario.

- Para representar números grandes harán falta muchos más dígitos.
  - Por ejemplo, para representar números mayores de 255 se necesitarán más de ocho dígitos, porque $\mathbf{2^8}$ = 256 y podemos afirmar, por tanto, que 255 es el número más grande que puede representarse con ocho dígitos.

- Como regla general, con ***n*** dígitos binarios pueden representarse un máximo de $\mathbf{2^n}$, números. El número más grande que puede escribirse con ***n*** dígitos es una unidad menos, es decir, $\mathbf{2^n - 1}$.
  - Con cuatro bits, por ejemplo, pueden representarse un total de **16** números, porque $\mathbf{2^4 = 16}$ y el mayor de dichos números es el **15**, porque $\mathbf{2^4 - 1 = 15}$.

Ejercicios

Ejercicio 2:

- Averigua cuántos números pueden representarse con 8, 10, 16 y 32 bits y cuál es el número más grande que puede escribirse en cada caso.

Ejercicio 3:

- Dados dos números binarios: **01001000** y **01000100** ¿Cuál de ellos es el mayor? ¿Podrías compararlos sin necesidad de convertirlos al sistema decimal?

Sistema de numeración octal

- El inconveniente de la codificación binaria es que la representación de algunos números resulta muy larga. Por este motivo se utilizan otros sistemas de numeración que resulten más cómodos de escribir: el sistema octal y el sistema hexadecimal. Afortunadamente, resulta muy fácil convertir un número binario a octal o a hexadecimal.
- En el sistema de numeración octal, los números se representan mediante **ocho** dígitos diferentes: 0, 1, 2, 3, 4, 5, 6 y 7. Cada dígito tiene, naturalmente, un valor distinto dependiendo del lugar que ocupen. El valor de cada una de las posiciones viene determinado por las potencias de base 8.

Ejemplo

- Por ejemplo, el número octal 273_8 tiene un valor que se calcula así:

$$2* 8^3 + 7* 8^2 + 3* 8^1 = 2*512 + 7*64 + 3*8 = 1496_{10}$$

$$273_8 = 1496_{10}$$

Conversión de un número decimal a octal

- La conversión de un número decimal a octal se hace con la misma técnica que ya hemos utilizado en la conversión a binario, mediante divisiones sucesivas **por 8** y colocando los restos obtenidos **en orden inverso**. Por ejemplo, para escribir en octal el número decimal **12210** tendremos que hacer las siguientes divisiones:

122 : 8 = 15	Resto: **2**
15 : 8 = 1	Resto: **7**
1 : 8 = 0	Resto: **1**

- Tomando los restos obtenidos en orden inverso tendremos la cifra octal:

$$\mathbf{122_{10} = 172_8}$$

Ejercicio

Ejercicio 5:

- Convierte los siguientes números decimales en octales: $\mathbf{63_{10}}$, $\mathbf{513_{10}}$, $\mathbf{119_{10}}$

Conversión octal a decimal

- La conversión de un número octal a decimal es igualmente sencilla, conociendo el peso de cada posición en una cifra octal.

Por ejemplo, para convertir el número $\mathbf{237_8}$ a decimal basta con desarrollar el valor de cada dígito:

- $\mathbf{2*8^2 + 3*8^1 + 7*8^0 = 128 + 24 + 7 = 159_{10}}$

$$\mathbf{237_8 = 159_{10}}$$

Ejercicio

Ejercicio 6:

- Convierte al sistema decimal los siguientes números octales: $\mathbf{45_8}$, $\mathbf{125_8}$, $\mathbf{625_8}$

Sistema de numeración hexadecimal

- Sistema de numeración hexadecimal

En el sistema **hexadecimal** los números se representan con dieciséis símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Se utilizan los caracteres A, B, C, D, E y F representando las cantidades decimales 10, 11, 12, 13, 14 y 15 respectivamente, porque no hay dígitos mayores que 9 en el sistema decimal. El valor de cada uno de estos símbolos depende, como es lógico, de su posición, que se calcula mediante potencias de base 16.

Ejemplo

Calculemos, a modo de ejemplo, el valor del número hexadecimal $1A3F_{16}$:

$$1A3F_{16} = 1*\mathbf{16^3} + A*\mathbf{16^2} + 3*\mathbf{16^1} + F*\mathbf{16^0}$$

$$1*4096 + 10*256 + 3*16 + 15*1 = 6719$$

$$1A3F_{16} = 6719_{10}$$

Ejercicio

Ejercicio 7:

- Expresa en el sistema decimal las siguientes cifras hexadecimales: $\mathbf{2BC5_{16}}$, $\mathbf{100_{16}}$, $\mathbf{1FF_{16}}$

Convertir números Decimales a hexadecimal

Por ejemplo, para convertir a hexadecimal del número $\mathbf{1735_{10}}$ será necesario hacer las siguientes divisiones:

1735 : 16 = 108 Resto: **7**
108 : 16 = 6 Resto: **C** es decir, $\mathbf{12_{10}}$
6 : 16 = 0 Resto: **6**

De ahí que, tomando los restos en orden inverso, resolvemos el número en hexadecimal:

$$\mathbf{1735_{10} = 6C7_{16}}$$

Ejercicio

- Convierte al sistema hexadecimal los siguientes números decimales: $\mathbf{3519_{10}}$, $\mathbf{1024_{10}}$, $\mathbf{4095_{10}}$

Conversión de números binarios a octales y viceversa

- Cada dígito de un número octal se representa con tres dígitos en el sistema binario. Por tanto, el modo de convertir un número entre estos sistemas de numeración equivale a "expandir" cada dígito octal a tres dígitos binarios, o en "contraer" grupos de tres caracteres binarios a su correspondiente dígito octal

Decimal	Binario	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Ejemplo

$$101001011_2 = 513_8$$

Ejercicio

Ejercicio 9:

- Convierte los siguientes números binarios en octales: $\mathbf{1101101_2}$, $\mathbf{101110_2}$, $\mathbf{11011011_2}$, $\mathbf{101101011_2}$

Conversión de números binarios a hexadecimales y viceversa

Podemos establecer una equivalencia directa entre cada dígito hexadecimal y cuatro dígitos binarios

Decimal	Binario	Hex
0	0000	0
15	1111	F

Ejemplo

Por ejemplo, para expresar en hexadecimal el número binario 101001110011_2 bastará con tomar grupos de cuatro bits, empezando por la derecha, y reemplazarlos por su equivalente hexadecimal:

- $\mathbf{1010_2 = A_{16}}$
- $\mathbf{0111_2 = 7_{16}}$
- $\mathbf{0011_2 = 3_{16}}$
- y, por tanto: $\mathbf{101001110011_2 = A73_{16}}$
-

En caso de que los dígitos binarios no formen grupos completos de cuatro dígitos, se deben añadir ceros a la izquierda hasta completar el último grupo. Por ejemplo:

$$\mathbf{101110_2 = 00101110_2 = 2E_{16}}$$

- **Ejercicio 11:**

Convierte a hexadecimales los siguientes números binarios:

1010100101011101010_2, 1110000111110000_2, 1010000111010111_2

Instrucciones vs. Datos

- **Instrucción**: conjunto de símbolos insertados en una secuencia estructurada o específica que el procesador interpreta y ejecuta.
- **Datos**: Símbolos que representan hechos, condiciones, situaciones o valores. Elementos de información.

Instrucciones vs. Datos (cont)

- Lenguaje natural:

```
Suma lo que hay en A con lo que tiene la posición
17 de una secuencia de valores.
```

- Lenguaje de programación de alto nivel:

```
A= A + M[17]
```

- Ensamblador y lenguaje máquina:

```
ADD A, M(17)      →      11000   001   0001   001111
```

Hardware (Soporte Físico)

Firmware

- El firmware es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, flash, etc), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo es en parte hardware, pero también es software, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación.
- Funcionalmente, el firmware es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica, ya que es el encargado de controlar a ésta última para ejecutar correctamente dichas órdenes externas.

Firmware


LINKSYS


Menu
Music Library
Now Playing
Play Mode
Photos
FM Radio
Extras
CREATIVE

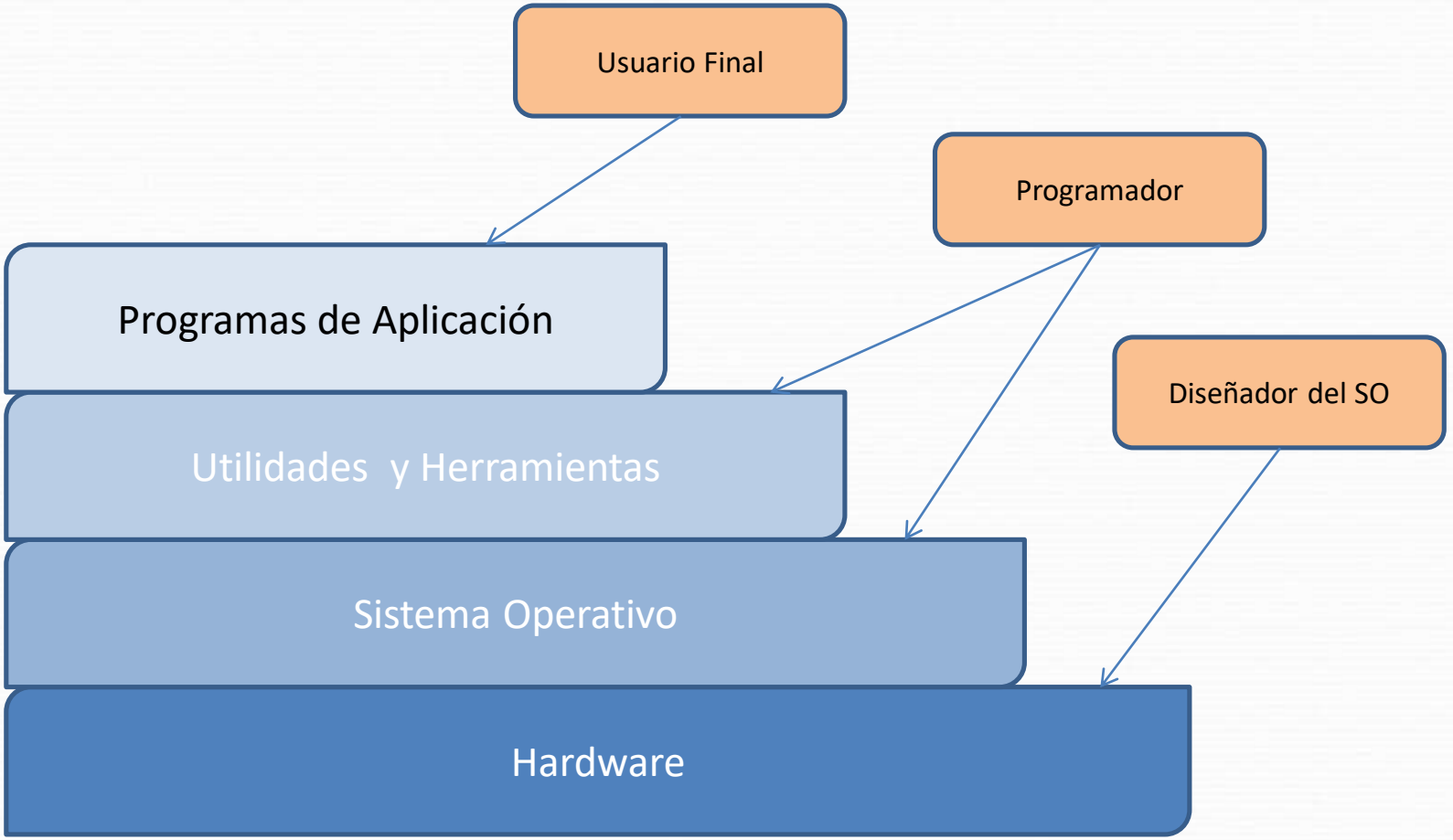

HD DVD
LG

Software (Soporte Lógico)

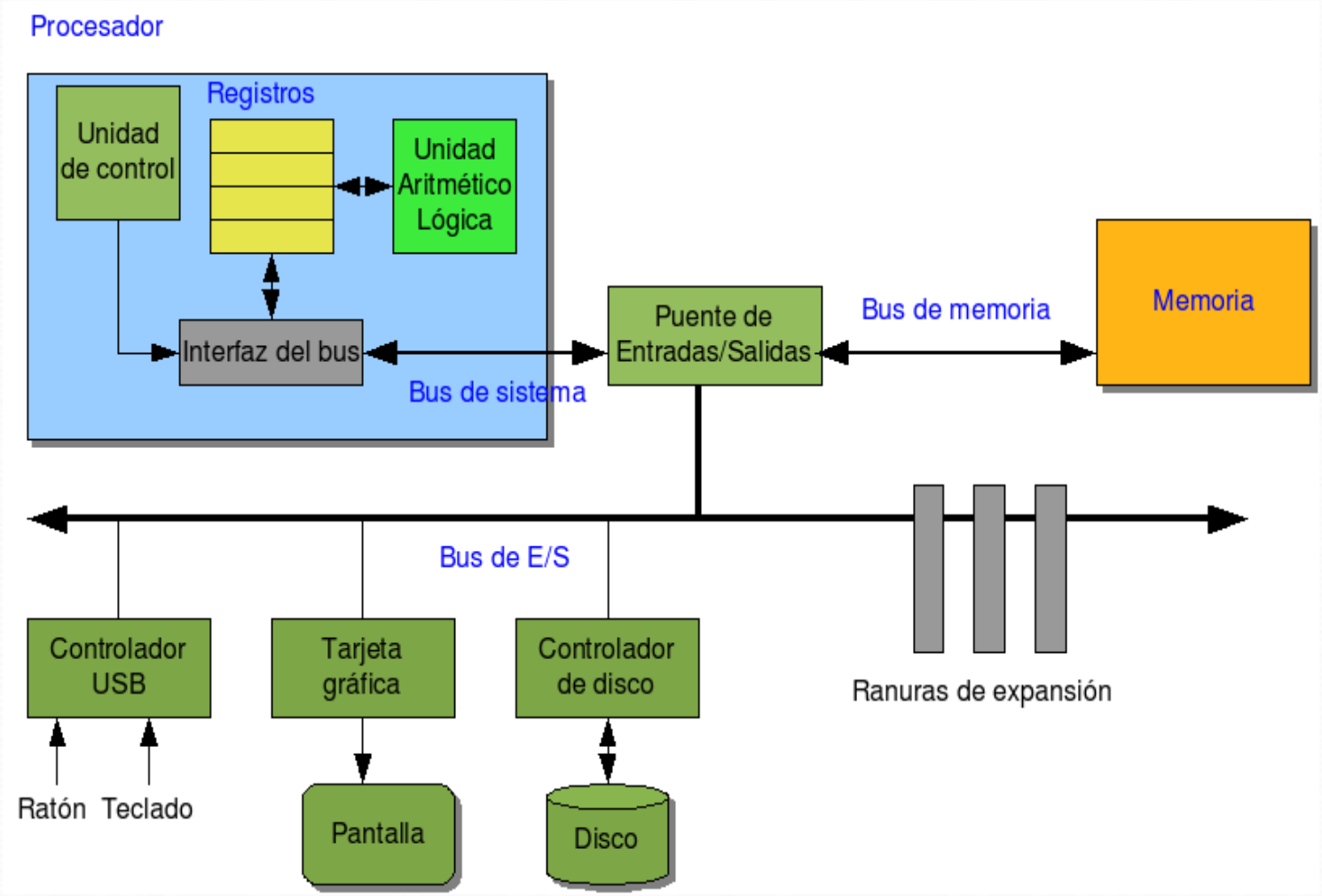
Definiciones Básicas [Stal05] (pp.55)

- El hardware y el SW utilizados para proporcionar aplicaciones a los usuarios se pueden ver de forma jerárquica o en capas

Definiciones Básicas [Stal05] (pp.55)

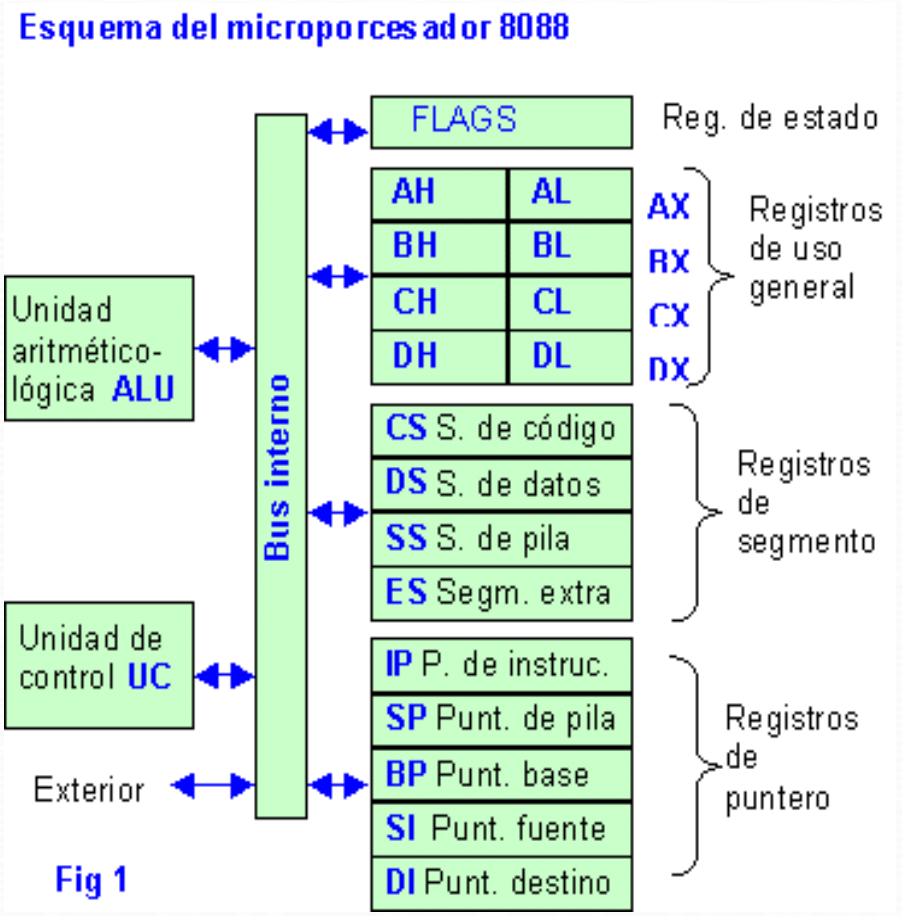


Arquitectura de un Sistema



Registros del Procesador [Stal05] (pp.11-13)

- Registros visibles para el usuario.
- Registros de control y estado
  - Contador de programa (PC).
  - Puntero de pila (SP).
  - Registro de instrucción (IR).
  - Registro de estado (bits informativos)



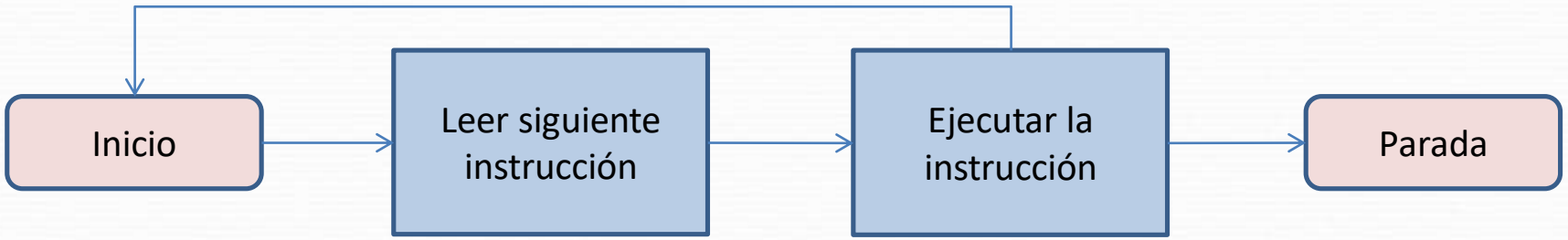
Ejecución de Instrucciones [Stal05] (pp.14-17)

- Procesar una instrucción consta de dos pasos:

  1. El Procesador **lee** (busca) instrucciones de la memoria, una cada vez.
  2. El Procesador **ejecuta** cada instrucción.

- La ejecución de un programa consiste en repetir el proceso de búsqueda y ejecución de instrucciones.

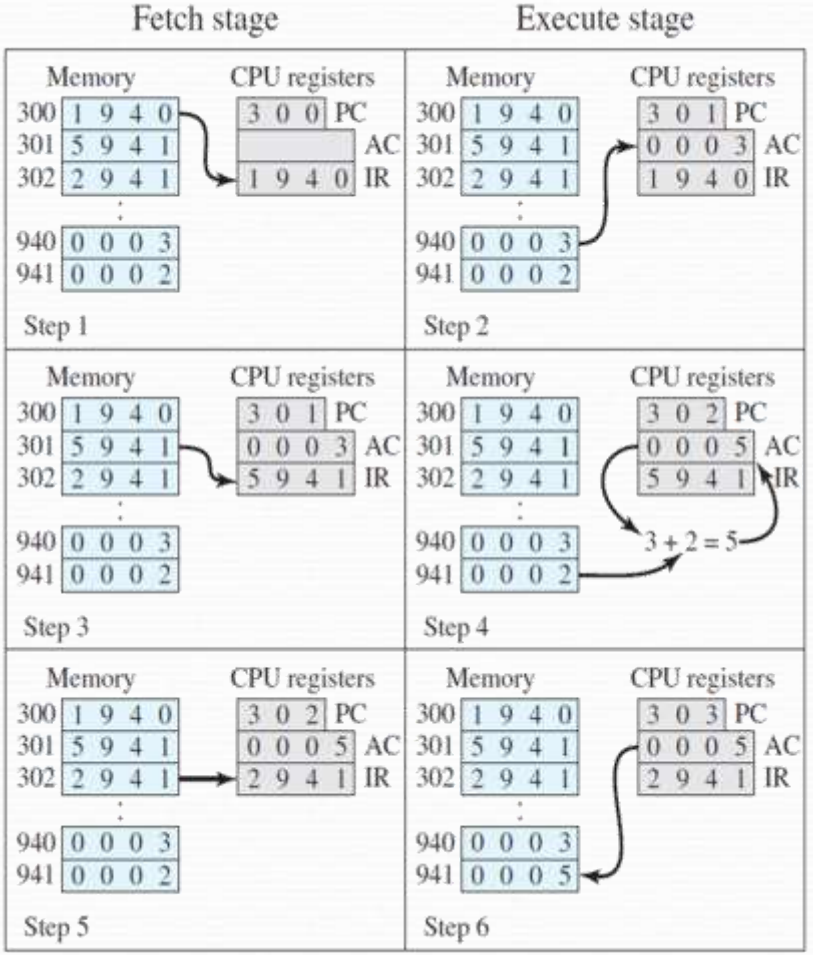
- Se denomina **ciclo de instrucción** al procesamiento requerido por una única instrucción.



Ejecución de Instrucciones [Stal05] (pp.14-17)

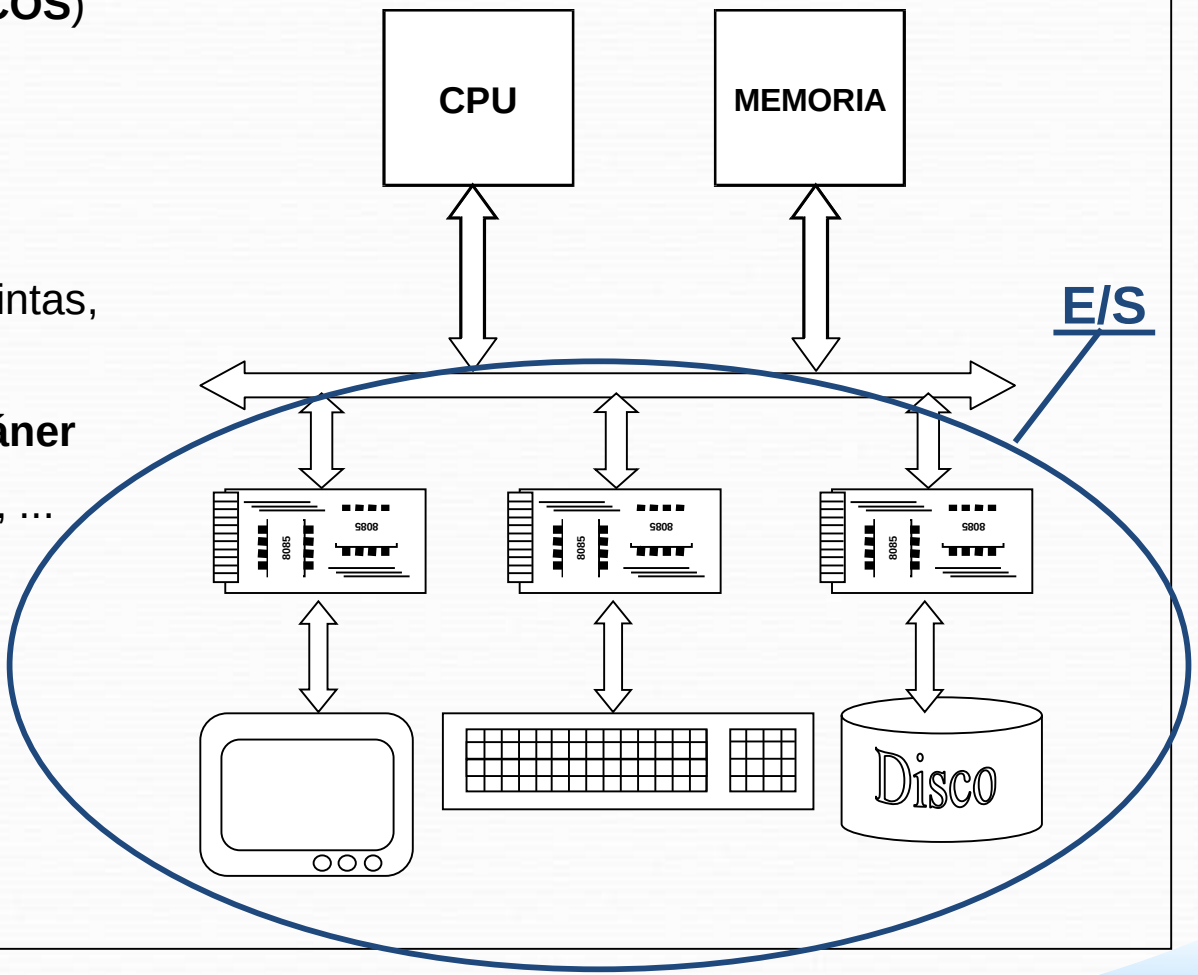
- **Ejemplo 1.**

1. El contador del programa (PC) tiene 300, la dirección de la primera instrucción.
2. Los primeros 4 bits (dígitos en hexadecimal) en el registro de instrucción (IR) indica que el acumulador (AC) será cargado desde memoria. Los siguientes 12 bits (tres dígitos en hexadecimal) indican la dirección, 940.
3. La siguiente instrucción (5941) será captada desde la dirección 301. El PC se incrementa.
4. El anterior contenido del AC y el contenido de la dirección 941 se suman y el resultado se almacena en el AC.
5. La siguiente instrucción (2941) será captada desde la dirección 302. El PC se incrementa.
6. El contenido del AC se aloja en la dirección 941.



Subsistema de E/S

- La E/S permite al computador interactuar con el “mundo exterior”
- Dispositivos típicos de E/S (**PERIFÉRICOS**)
  - **Dispositivos de E/S básica**
    - teclado, ratón, pantalla
  - **Dispositivos de almacenamiento**
    - discos, disquetes, CD-ROM, cintas, discos magneto-ópticos, ...
  - **Dispositivos de impresión y escáner**
    - impresoras, plotters, scanners, ...
  - **Dispositivos de comunicación**
    - redes, módems, ...
  - **Dispositivos multimedia**
    - audio, video, ...
  - …



Técnicas de Comunicación de E/S [Stal05] (pp.34)

¿Qué necesita la CPU para hacer su trabajo leer/escribir en memoria o en un dispositivo de E/S:

Hay tres técnicas para llevar a cabo las operaciones de E/S:

- **E/S Programada.**
- **E/S dirigida de interrupciones.**
- **Acceso directo de memoria**

Técnicas de Comunicación de E/S [Stal05] (pp.34)

- **E/S Programada.** El procesador encuentra una instrucción con la E/S. Se genera un mandato al módulo de E/S apropiado.

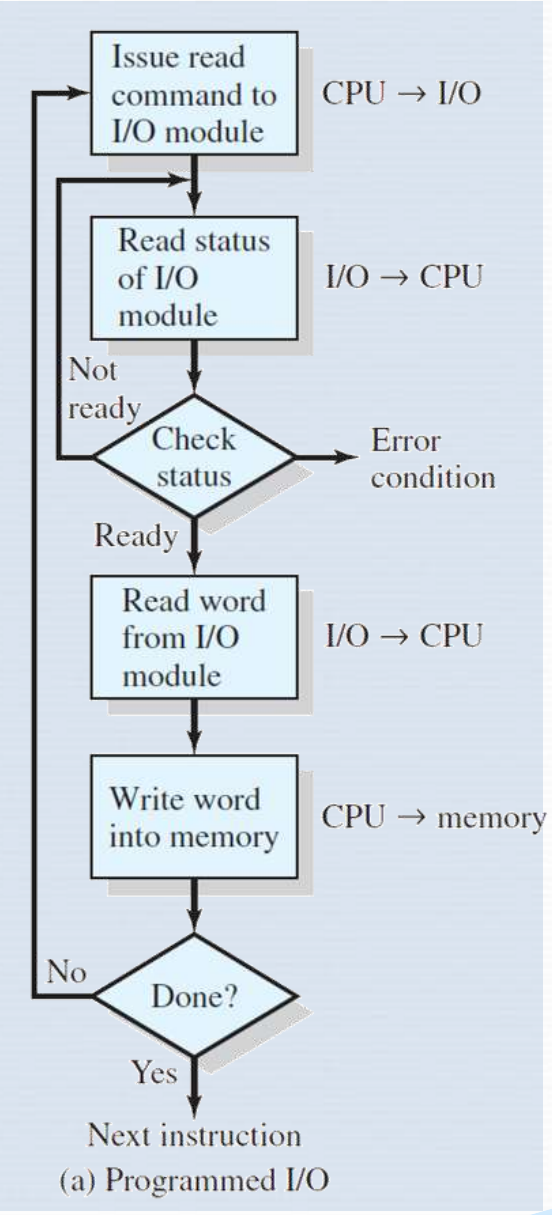
- Con esta técnica, el procesador es el responsable de extraer los datos de la memoria principal en una operación de salida y almacenarlos en ella en una operación de entrada

- El software de E/S se escribe de manera que el procesador ejecuta instrucciones que le dan control directo de la operación de E/S incluyendo:
  - Comprobar el estado del dispositivo
  - Enviar un mandato de lectura o de escritura
  - Transferir los datos

Técnicas de Comunicación de E/S [Stal05] (pp.34)

La figura muestra un ejemplo del uso de E/S programada para leer un bloque de datos de un dispositivo externo (p. ej. Un registro de cinta) y almacenarlo en memoria.

- Los datos se leen palabra a palabra (por ejemplo 16 bits).
- Por cada palabra que se lee, el procesador debe permanecer en un bucle de comprobación del estado hasta que determina que la palabra está disponible en el registro de datos del módulo de E/S.



(a) Programmed I/O

E/S programada

Problema: El problema de la técnica de la E/S programada es que el procesador tiene que esperar mucho tiempo hasta que el módulo de E/S correspondiente esté listo para la recepción o la transmisión de más datos. El procesador mientras está esperando, debe comprobar repetidamente el estado del módulo de E/S.

- Cómo resultado, el nivel de rendimiento de todo el sistema se degrada gravemente.

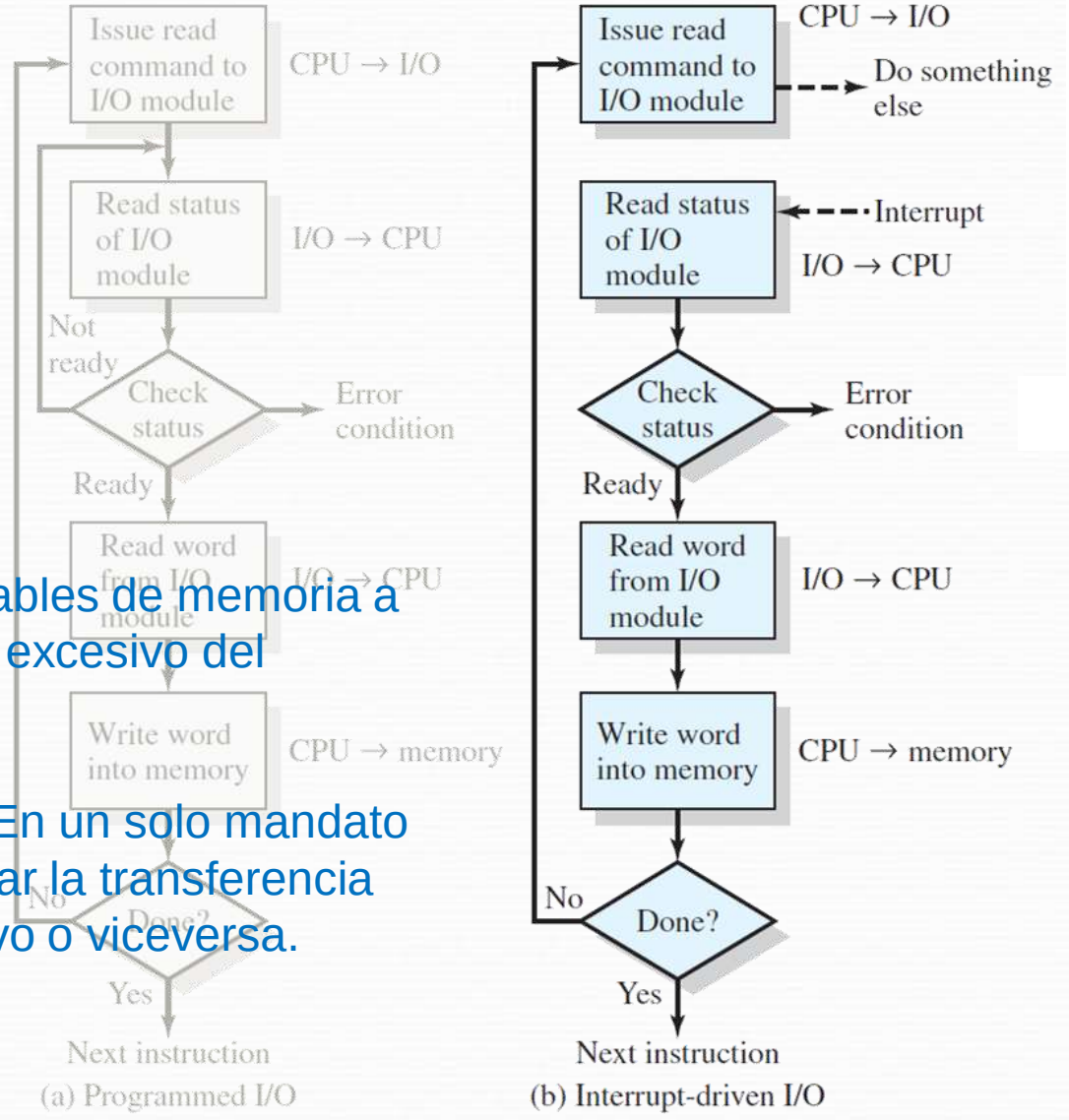
Solución: Mientras se atiende al módulo de E/S, el procesador pueda continuar con trabajo útil.

Técnicas de Comunicación de E/S

- **E/S Dirigida por Interrupciones.** Evento que interrumpe el flujo normal de ejecución producido por un elemento externo al procesador. Es un evento asíncrono.

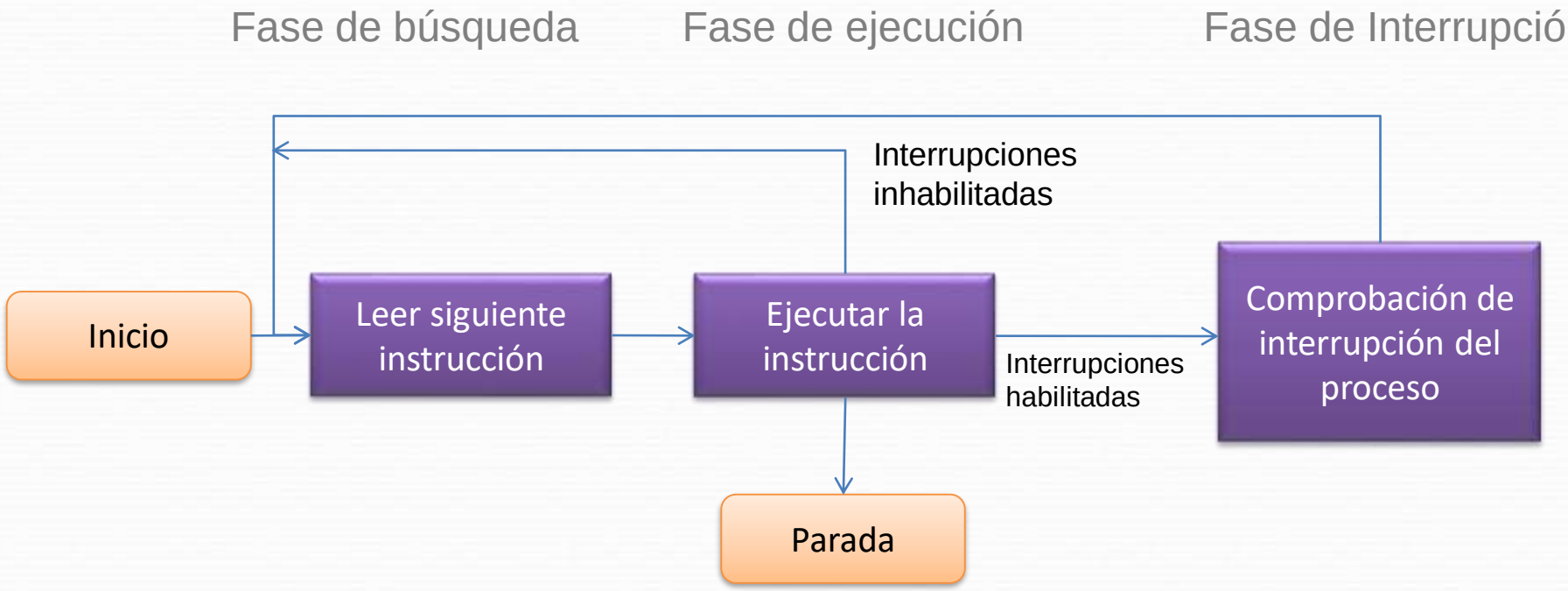
Problema: En transferencias considerables de memoria a dispositivo o viceversa conlleva un uso excesivo del procesador.

Solución: Acceso Directo a Memoria. En un solo mandato se genera todo lo necesario para realizar la transferencia de información de memoria al dispositivo o viceversa.

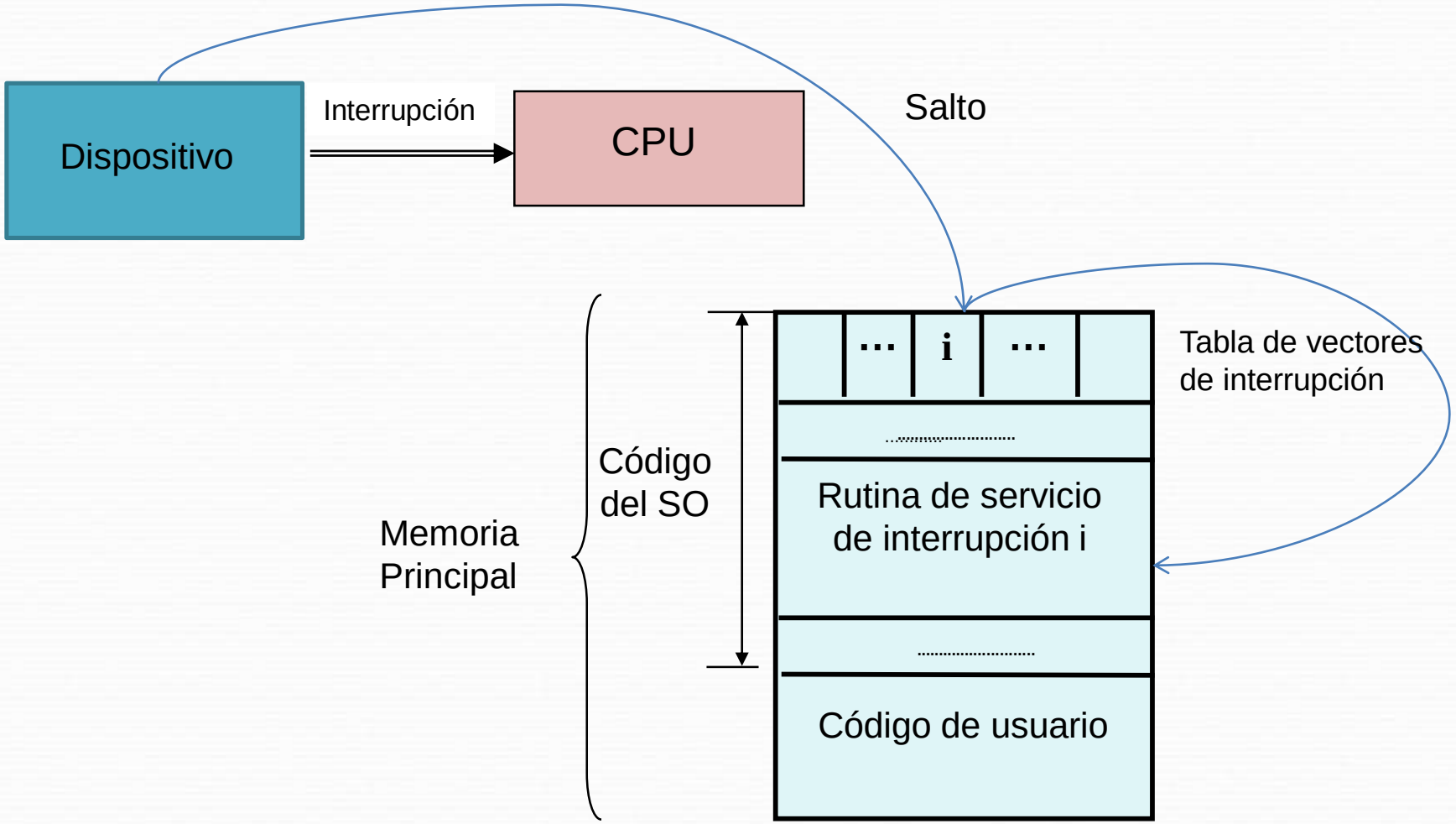


Técnicas de Comunicación de E/S

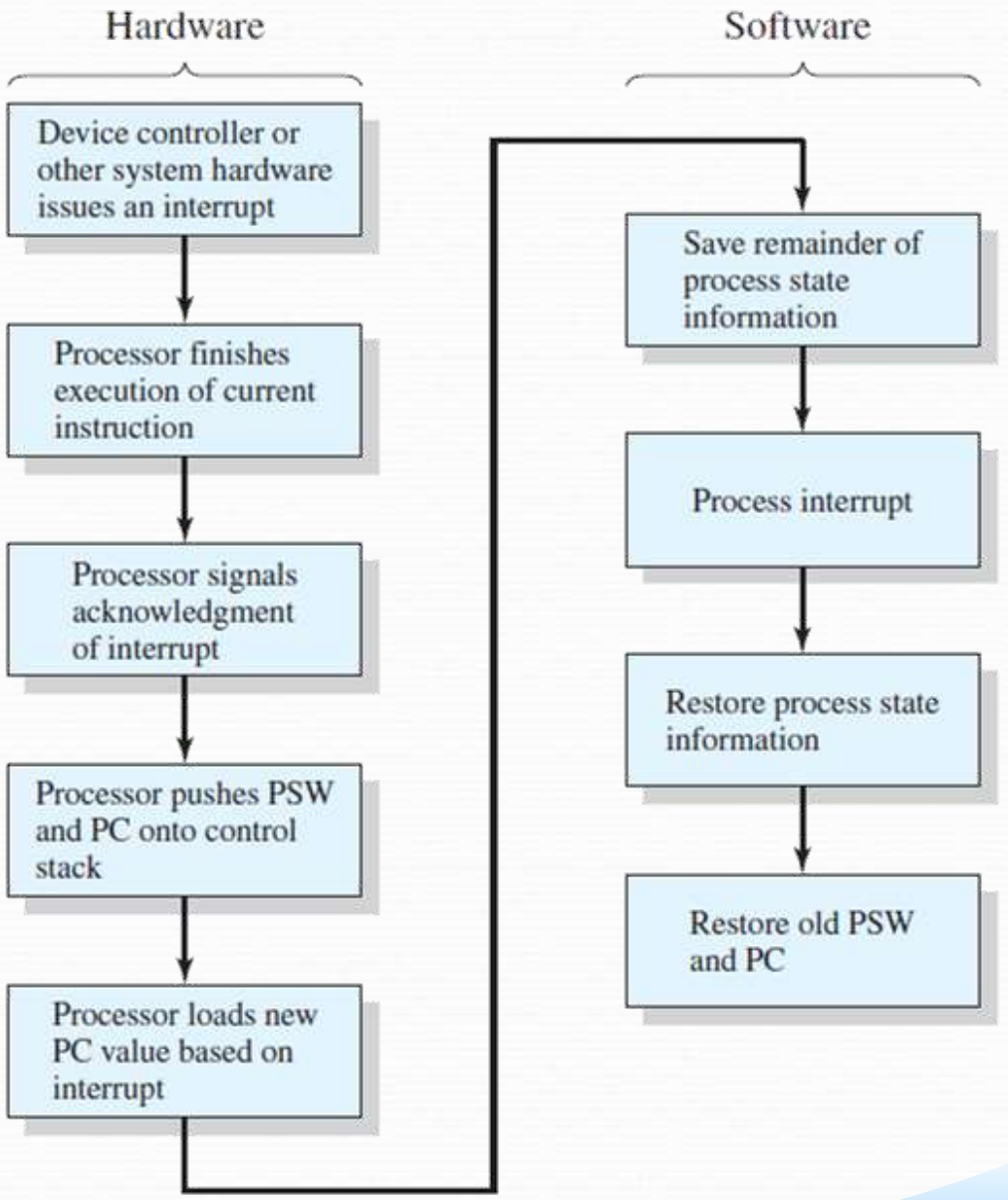
- **Ciclo de instrucción** con interrupciones.



Tratamiento de Interrupciones Vectorizadas

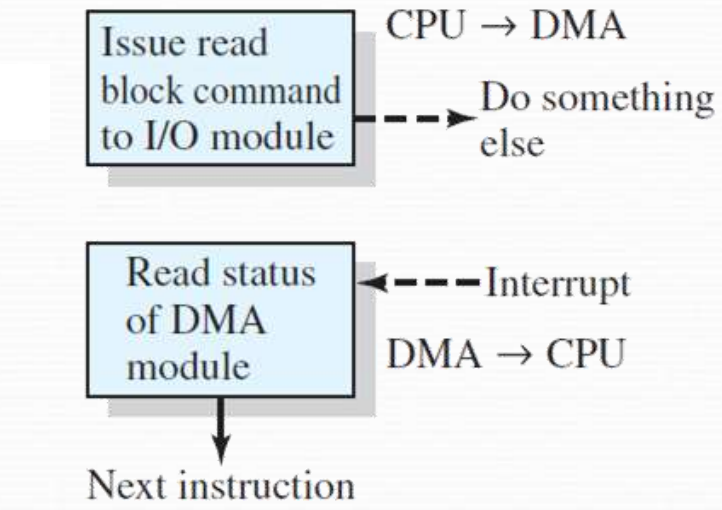


Tratamiento de Interrupciones Vectorizadas [Stal05] (pp. 23. Fig. 1.10)



Técnicas de Comunicación de E/S [Stal05] (pp.34-37)

- **Acceso Directo a Memoria (DMA, Direct Access Memory).** Realizada por un módulo separado conectado en el bus del sistema o incluida en un módulo de E/S. Útil cuando el procesador desea leer o escribir un bloque de datos.



(c) Direct memory access

Excepciones [Stal05] (pp.34-37)

- **Definición de excepción:** Evento inesperado generado por alguna condición que ocurre durante la ejecución de una instrucción (ejemplo, desbordamiento aritmético, dirección inválida, instrucción privilegiada, etc.). Es un evento síncrono.

Protección [Carr07] (pp.4)

- **Funcionamiento en Modo Dual.** ¿Qué ocurre si un programa accede a la memoria donde se alojan los vectores de interrupciones? ¿Qué pasa si las modifica?

Solución: El procesador dispone de diferentes modos de ejecución de instrucciones:

- **Instrucciones privilegiadas (modo supervisor/kernel):** Aquellas cuya ejecución puede interferir en la ejecución de un programa cualquiera o programa del SO (ejemplo, escribir en el puerto de un dispositivo).

- **Instrucciones no privilegiadas (modo usuario):** Aquellas cuya ejecución no presenta ningún problema de seguridad para el resto de programas (ejemplo, incrementar un contador).

Protección de los Dispositivos de E/S [Carr07] (pp.25-28)

- Los dispositivos de E/S son recursos que han de estar protegidos (ejemplo, los archivos, las impresoras, …)
- ¿Cómo se consigue? → Las instrucciones máquina para acceso a los dispositivos de E/S no pueden ejecutarse en modo usuario: son privilegiadas.
- Cualquier acceso a los dispositivos desde un programa de usuario se hará mediante peticiones al SO.

Protección de Memoria

- Cada programa en ejecución requiere de un espacio de memoria.
- **Objetivo:** Hay que proteger la zona de memoria asignada y la memoria en la que está el código del sistema operativo (tabla de vectores de interrupción, rutinas de tratamiento de cada interrupción).

El Sistema Operativo [Stal05] (cap.2, pp.53-104)

Un SO es un programa o conjunto de programas que controla la ejecución de los programas de aplicación y que actúa como interfaz entre el usuario de una computadora y el hardware de la misma.

El SO como interfaz Usuario/Computadora

Presenta al usuario una máquina abstracta más fácil de programar que el hardware subyacente:

- Oculta la complejidad del hardware.
- Da tratamiento homogéneo a diferentes objetos de bajo nivel (archivos, procesos, dispositivos, etc.).

Una aplicación se puede expresar en un lenguaje de programación y la desarrolla un programador de aplicaciones.

Es más fácil programar las aplicaciones en lenguajes de alto nivel que en el lenguaje máquina que entiende el hardware.

El SO como interfaz Usuario/Computadora

Un SO proporciona normalmente utilidades en las siguientes áreas:

- **Desarrollo de programas** (editores de texto, compiladores, depuradores de programas).
- **Ejecución de programas** (cargador de programas y ejecución de éstos).
- **Acceso a dispositivos de E/S** (cada dispositivo requiere su propio conjunto de instrucciones).

El SO como interfaz Usuario/Computadora

(cont.)

- **Acceso al sistema** (En sistemas compartidos o públicos, el SO controla el acceso y uso de los recursos del sistema: Shell, Interfaz gráfico).
- **Detección y respuesta a errores** (tratamiento de errores a nivel software y hardware).
- **Contabilidad** (estadísticas de uso de los recursos y medida del rendimiento del sistema).

El SO como Administrador de Recursos

Un computador es un conjunto de recursos y el SO debe gestionarlos y para ello posee un mecanismo de control en dos aspectos:

- Las funciones del SO actúan de la misma forma que el resto del software, es decir, son programas ejecutados por el procesador.
- El SO frecuentemente cede el control y depende del procesador para volver a retomarlo.

El SO como Administrador de Recursos

Por lo tanto:

- El SO Dirige al procesador en el uso de los recursos del sistema y en la temporización de la ejecución de otros programas.
- Una parte del código del SO se encuentra cargado en la memoria principal (kernel y, en ciertos momentos, otras partes del SO que se estén usando). El resto de la memoria está ocupada por programas y datos de usuario.

El SO como Administrador de Recursos

Por lo tanto: (cont.)

- La asignación de la memoria principal la realizan conjuntamente el SO y el hardware de gestión de memoria del procesador.
- El SO decide cuándo un programa en ejecución puede usar un dispositivo de E/S y también el acceso y uso de los ficheros. El procesador es también un recurso.

Características deseables en un Sistema Operativo

- **Comodidad en el uso.**
- **Eficiencia**: Existen más programas que recursos. Hay que repartir los recursos entre los programas
- **Facilidad de Evolución**: Un SO importante debe evolucionar en el tiempo por las siguientes razones:
  - Actualizaciones del hardware y nuevos tipos de hardware.
  - Mejorar y/o aportar nuevos servicios.
  - Resolución de fallos.

Programas de Servicio del SO [Prie06] (Cap.13, sección 13.1, pp.518-520)

Se trata de un conjunto de programas de servicio que, en cierta medida, pueden considerarse como una ampliación del SO:

- Compactación de discos.
- Compresión de datos.
- Gestión de comunicaciones.
- Navegadores de internet.
- Respaldo de seguridad.
- Recuperación de archivos eliminados.
- Antivirus.
- Salvapantallas.
- Interfaz gráfica.

Herramientas Generales

Su misión es facilitar la construcción de las aplicaciones de los usuarios, sea cual sea la naturaleza de éstas, tales como:

- Editores de texto.
- Compiladores.
- Intérpretes.
- Enlazadores.
- Cargadores/Montadores.
- …