UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

RESUMEN

Organización del Computador
75.03 & 95.57

Autor
ANZU

July 11, 2019

Contents

1 Sistemas de numeración

1.1 Definición

Se conoce como un sistema de numeración un conjunto finito de símbolos que se emplea con algún método para asignar numerales , o símbolos numéricos, a los números. Hay diversos sistemas que han sido, o son actualmente empleados.

1.2 Clasificación

1.2.1 Sistemas de numeracion no posicionales

El valor de un simbolo es independiente de la posicion que ocupa.
Ejemplo: Numeros romanos

1.2.2 Sistemas de numeracion posicionales

El valor de un simbolo depende de la posicion que ocupa. El punto fraccionario es llamado punto decimal, en base diez, y punto binario, en base binaria. Ejemplo: Numeros decimales

1.3 Dígitos en distintos sistemas

- Binario: 0, 1
- Octal: 0, 1, 2, 3, 4, 5, 6, 7
- Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

1.4 Cambio de base

1.4.1 Numeros enteros

1.4.1.1 De base b a base 10

1.4.1.2 De base 10 a base n

1.4.1.3 De base n a base m

1.4.2 Numeros fraccionarios

1.4.2.1 De base b a base 10

1.4.2.2 De base 10 a base b

1.4.2.3 De base n a base m

1.4.3 Casos especiales de cambio de base

- $$b^x = p \Rightarrow 2^3 = 8$$

  tomo de a 3 dígitos

- $$b^{1/x} = p \Rightarrow 16^{1/4} = 2$$

  se expanden en 4 dígitos

1.5 Números de precisión finita

En la mayoria de las computadoras la cantidad de digitos para representar un número, se fija en su diseño. Observemos un ejemplo con números enteros positivos de tres digitos.

- 600+600 = 1200 muy grande
- 003-005 = -002 negativo
- 007/002 = 3.5 no es un entero

Las exclusiones se dividen en dos grupos, los resultados que no pertenecen al conjunto y los resultados que son mayores o menos al mayor y mínimo número del conjunto (overflow)

1.6 Representación de números negativos

1.6.1 Signo y magnitud

Se usa un bit para representa el signo, a menudo es el bit más significativo y por convención se usa el cero para números positivos y 1 para los negativos. Los n-1 bits restantes se usan para representar el significando que es la magnitud del número en valor absoluto.

1.6.1.1 Ventajas

- Posee un rango simetrico

1.6.1.2 Desventajas

- No permite operar arimeticamente
- Posee doble representación del cero

1.6.2 Complemento a uno

Consiste en aplicarle un NOT bit a bit al número. Se usa un bit para representar el signo, a menudo es el bit más significativo, por convención se usa un cero para los positivos y un 1 para los negativos. Los n-1 bit restantes para representar el significando que es la magnitud del numero en valor absoluto para el caso de números positivos o bien, es el complemento a uno del valor absoluto del número en caso de ser negativo.

1.6.2.1 Ventajas

- Posee un rango simetrico
- Permite operar arimeticamente y para obtener el resultado correcto al operar se debe sumar el acarreo obtenido al final de la suma/resta realizadas en caso de haberlo obtenido, este acarreo es conocido con el nombre de end-around carry.

1.6.2.2 Desventajas

- Posee doble representación del cero

1.6.3 Complemento a la base

El complemento de un número dado en una base es aquel que sumado al número original da la base a la n, siendo n la cantidad de dígitos que componene a ese número. El concepto de complemento puede ser usado para transformar una operación de resta de dos números en la suma de uno de ellos más el complemento del otro.

1.6.3.1 Complemento a 10 de 13579_{10}

$100000 - 13579 = 86421_{10}$

1.6.3.2 A-B

$A = 10376 \qquad B = 234$
$A - B = C$
$A + B_{comp} = C + D \qquad B_{comp} = 100000 - 234 = 99766$
$10376 + 99766 = D + 100000$
$10142 = D$

1.6.4 Complemento a dos

Permite representar números negatvios en el sistema binario y la realización de restas mediante sumas. Estos números negativos están representados a través de su complmento. El complemento a 2 de un número se obtiene de sumar 1 al número negado bit a bit.

1.6.4.1 Complemento a dos de 01101_2

$10010 + 00001 = 10011_2$

1.6.4.2 Ventajas

- No posee doble representación del cero
- Permite operar arimeticamente

1.6.4.3 Desventajas

- Posee un rango asimetrico

1.6.5 Exceso a la base

Consiste en tomar el valor real del número a representar sumarle la base elevada según la cantidad de dígitos menos 1 que se tienen disponibles. Esto se lo conoce como representación en Exceso a base B^{n-1}, puesto que a cada número se le suma el mismo valor y está en exceso por dicho valor.

1.6.5.1 Guardar

El exceso es $2^{8-1} = 128$
$-97_{10} + 128 = 31_{10}$
$31_{10} = 00011111_2$

1.6.5.2 Obtener

El exceso es $2^{8-1} = 128$
$181 - 128 = 53_{10}$
$53_{10} = 00110101_2$

1.6.5.3 Ventajas

- No hay empaquetación del número
- Permite operar arimeticamente

1.6.5.4 Desventajas

- Requiere operaciónes arimeticas intermedias
- Posee un rango asimetrico

2 Formato y Configuracion

2.1 Definición

- Formato: Representación computacional
- Configuración: Representación en una determinada base de un número en un formato

2.2 Expansión y truncamiento

2.3 Definición

- Expandir formato: Significa completar la representación computacional sin alterar el numero representado en el mismo.
- Truncar formato: Descartar digitos de su representación sin alterar el número representado en el mismo.

2.4 Formatos

2.4.1 Binario punto fijo sin signo

- Base: 2
- Representa: números enteros positivos
- Máximo: 2_{10}^{n-1}
- Mínimo: 0

2.4.1.1 Almacenar

1. Pasar el numero a base 2
2. completar con ceros a izquierda la capacidad del formato

2.4.1.2 Recuperar

1. Pasamos el numero de base 2 a la base deseada

2.4.2 Binario de punto fijo con signo

- Base: 2
- Representa: Enteros positivos y negativos
- Primer bit: reservado para el signo
- Máximo: $2^{n-1} - 1$
- Mínimo: -2^{n-1}

2.4.2.1 Almacenar

1. Pasar el numero a base 2
2. completar con ceros a izquierda la capacidad del formato
3. Si es un numero negativo hacerle el complemento a 2

2.4.2.2 Recuperar

1. Si el bit de signo es cero, se pasa de base 2 a base 10
2. Si el bit es 1, el numero es negativo, por lo que debemos complementarlo
3. Quitamos los ceros a la izquierda
4. Numero de base 2 a base 10
5. Colocamos el signo

2.4.2.3 Expansión

1. Se completa con el bit de signo a la izquierda

2.4.2.4 Truncamiento

1. Se extraen bits a la izquierda siempre y cuando no se esté alterando el bit de signo del número.

2.4.3 Empaquetado

- Base: 16
- Representa: Enteros positivos y negativos
- Máximo: $10^{2n-1} - 1$
- Mínimo: $-10^{2n-1} + 1$

2.4.3.1 Almacenar

1. numero a base 10
2. Colocal cada digito decimal en un nibble dejando el ultimo nibble ya que en el mismo se almacena el signo.
3. Colocar en el ultimo nible el signo, CAFE = + , DF = -
4. Se rellena con 0 hasta alcanzar la cantidad de bytes usados

2.4.3.2 Recuperar

1. los pasos en orden inverso

2.4.4 Zoneado

- Base: 16
- Representa: Enteros positivos y negativos
- Máximo: $10^n - 1$
- Mínimo: $-10^n + 1$

2.4.4.1 Almacenar

1. numero a base 10
2. colocar cada uno de los digitos decimales en un nibble derecho
3. completar todos los nibbles de izquierda con F salvo el ultimo que se completa con el signo siguiendo las mismas reglas que para empaquetados. CAFE = + , DF = -
4. se rellena con F0 hasta alcanzar la cantidad de bytes usados

2.4.4.2 Recuperar

1. los pasos en orden inverso

2.4.5 Binario de punto Flotante

es la manera que tiene una arquitectura de representar a los numeros reales. su notación cientifica se expresa de la siguiente manera MxB^E M: Mantissa B: base E: Exponente

Un numero binario está normalizado si el digito de la izquierda del punto es igual a 1.

2.4.5.1 IEEE754

- Precision Simple: signo: 1 bit exponente: 8 bits fraccion: 23 bits Exponente: Exceso 127
- Precision Doble: signo: 1 bit exponente 11bits fraccion: 52 bits Exponente: Exceso 1023

2.4.5.2 Ancho de paso

Marca cual es la distancia entre un flotante y su siguiente numero representable en el formato

2.4.5.3 Overflow

el exponente excede el limite superior, tanto para mantisas positivas como para negatvias, dando lugar a +inf, - inf

2.4.5.4 Underflow

el exponente excede el minimo valor permitido y caga en el intervalo (-inf, -0) y (+0,+inf)

2.4.5.5 Desnormalizados - Subnormales

tienen como exponente al cero, y el bit implicito a la izquierda del punto binario, es ahora un cero implicito. la diferencia entre los desnormalizados y los normalizados es que, estos ultimos no permiten al cero como exponente. Los normalizados tienen 24 bits significativos, mientras que los normalizados poseen 23.

Normalizado	+/-	0<exp<max	cualquier patron de bits
Desnormalizado	+/-	0	cualquier patron de bits != 0
Cero	+/-	0	0
Infinito	+/-	11...11	0
NAN	+/-	11...11	Cualquier patron de bits != 0

- Infinito dividido Infinito = NAN
- Cero + = 0 00000000 00000000000000000000000
- Cero - = 1 00000000 00000000000000000000000
- Infinito + = 0 11111111 00000000000000000000000
- Infinito - = 1 11111111 00000000000000000000000
- No normalizados/Subnormales (no se asume que haya que añadir un 1 al significando para obtener su valor).

2.4.5.6 Valores no numericos

NaN (Not a number). 2 tipos, QNaN (quiet nan) y SNaN (signalling nan) Qnan = indeterminado Snan = operacion no valida

- Infinito dividido Infinito = NAN
- qnan = 0 11111111 10000100000000000000000
- snan = 1 11111111 00100010001001010101010

Operacion	Resultado
n $\pm$ infinito	0
$\pm$ infinito $\cdot$ $\pm$infinito	$\pm$infinito
$\pm$n $\div$ 0	$\pm$infinito
Infinito + Infinito	Infinito
Cualquier operación contra un NaN	NaN
$\pm 0 \div \pm 0$	NaN
Infinito - Infinito	NaN
$\pm$Infinito $\div$ $\pm$Infinito	NaN
$\pm$Infinito $\cdot$ ± 0	NaN

2.4.5.7 Almacenar -6,12510

1. El bit 31 tomará el valor del signo del número.
2. Pasar a binario la mantisa decimal.
   - $6 = 110_2$
   - $0,125 = 0,001_2$
   - $6,125 = 110,001_2$
3. Normalizar
   - Desplazamiento a la derecha -> Exponente negativo
   - Desplazamiento a la izquierda -> Exponente positivo
   - 1,10001 , exponente = 2
   - 2 expresado en Exceso 127 es 129 0000001_2
4. Mantisa representada con bit implícito: 1,10001 -> 10001
5. El número final es 1 10000001 10001000000000000000002 (Se agregan a la derecha los "0" necesarios para completar los 23 bits de la mantisa)

2.4.5.8 Recuperar

1. Se realizan los pasos en orden inverso

2.4.5.9 Binario de Punto Flotante (IBM mainframe)

- Base: 16
- Representa: enteros con coma decimal positivos y negativos
- Precision : imple 4 bytes, doble, 8 bytes, extendida 16 bytes.
- estructura: S nnnnnnn dddddd
  - s = signo 1- 0+
  - n = digitos de la caracteristica, en total son 7 bits se usan para calcular el exponente
  - C = E + 4016 donde E corresponde al exponente
  - d = mantisa normalizada -> 0,dddddd x 10^{e}_{16}

2.4.5.10 Almacenar - 321,54 10 -> Binario de punto flotante precisión simple

1. $321,54_{10} = 141,8A3_{16}$
2. $0,1418A3x10^{3}_{16}$
3. $C = E + 40_{16} = 3 + 40_{16} = 43_{16} en base 2 sera 100011_2$
4. Agregamos el bit de signo: 1100011_2 que en base 16 es $C3_{16}$
5. Resultado final : $C31418A3_{16}$

2.4.5.11 Ancho de paso

distancia entre un flotante y su siguiente numero representable en el formato

2.4.5.12 Absorcion

se da en las operaciones de suma y resta entre flotantes

- $A = 0,15A4x105_{16}$
- $B = 0,54Fx10^{-}2_{16}$

para poder operar entre flotantes debemos igualar los exponentes llevandolos al mayor de todos

- $A + B = 0.15A400 * 10^5 + 0,00...00 * 10^5 = 0,15A4000 * 10^5_{16}$

lo minimo que se puede sumar es el ancho de paso del numero de mayor exponente

3 Cadenas de caracteres

En una cadena de caracteres, cada caracter ocupa 1 byte y se representa según el codigo de caracteres que se esté utilizando.

3.1 ASCII

abbreviated from American Standard Code for Information Interchange, is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

3.2 EBCDIC

is an eight-bit character encoding used mainly on IBM mainframe and IBM midrange computer operating systems. It descended from the code used with punched cards and the corresponding six-bit binary-coded decimal code used with most of IBM's computer peripherals of the late 1950s and early 1960s.

3.3 UNICODE

Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems. The Unicode Standard consists of a set of code charts for visual reference, an encoding method and set of standard character encodings, a set of reference data files, and a number of related items, such as character properties, rules for normalization, decomposition, collation, rendering, and bidirectional display order (for the correct display of text containing both right-to-left scripts, such as Arabic and Hebrew, and left-to-right scripts).

3.3.1 Codificación

codificación organizada en planos. Cada plano tiene un espacio de 16 bits y la cantidad de planos definida es 17. plano 0 al plano 16. por lo que la cantidad de codigos posibles es $17 * 2^{16} = 1.114.112$

U-XXXXYYYY XXXX el valor hexadecimal del plano YYYY el valor hexadecimal correspondiente al caracter dentro del plano

U-00000040 significa plano 0 codigo 40_{16} y corresponde especificamente a la representación de la letra M.

Para el caso particular del plano 0 suele usarse una notación simplificaa la cual omite expresar el numero del plano y remplazar el - posterior a la U por el signo +. U+YYYY. Para el caso de ejemplo quedaría U+0040.

El rango total entonces es: 00000000 al 0010FFFF

3.3.2 UTF - Unicode Transformation Format

Consiste en un mapeo algoritmico de un codigo unicode a una secuencia de bytes.

3.3.2.1 Tipos

1. UTF-8
2. UTF-16
3. UTF-32

3.3.3 UTF - 8

Segun el rango al cual corresponde dentro de la tabla Unicode, un caracter puede representarse como 1, 2, 3 o 4 tiras de 8 bits.

```
Rango [00...7F]
Se necesitan 7 bits y se representan en 1 byte.
Primer bit en 0 y el resto igual que el codigo ASCII

Rango [80...7FF]
Se necesitan 11 bits y se representan en 2 bytes.
byte 1:
Los primeros 3 bits son fijos 110. Los 2 primeros en 1 previous a un 0 dan la señal de que s
la representación.
byte 2:
los primeros 2 bits son fijos 10
los 6 bits siguientes contienen los ultimos 6 del caracter segun la tabla unicode
```

```
rango [800...FFFF]
byte 1: los primeros 4 bits son fijos 1110
los 4 bits siguienten contienen los 4 primeros del caracter segun la tabla unicode
byte 2: los primeros 2 bits son fijos 10
los siguientes 6 bits contienen los siguientes 6 del caracter segun la tabla unicode
byte 3: idem segundo byte

rango [10000... 10FFFF]
Se necesitan 21 bits y se representan en 3 bytes
byte1:
los primeros 5 bits son fijos 11110
los 3 siguientes contienen los 3 primeros del caracter segun la tabla unicode
byte2:
los primeros 2 bits son fijos 10
los siguientes 6 bits contienen los siguientes 6 del caracter segun la tabla unicode
byte3
idem segundo byte
byte4
idem segundo byte
```

3.3.3.1 Ejemplo

El código Unicode de la ñ es el 00F116 = 0000 0000 1111 0001
Se encuentra en el rango entre el "80 y 7FF", por lo tanto se necesitan 11 bits (00011110001)
y se usarán 2 bytes para la representación:
Byte 1: 11000011
Byte 2: 10110001
Representación Unicode 11000011101100012 = C3B1

3.3.4 UTF - 16

Según el rango al cual corresponde dentro de la tabla Unicode, un caracter puede representarse como 1 o 2 tiras de 16 bits cada una.

```
Rango [0...FFFF]
Se necesitan 16 bits y se representan en 2 bytes tal cual se define en la tabla Unicode
Rango [10000... 10FFFF]
Siendo U el código unicode del caracter a representar, y dado que el máximo
valor es 10FFFF se calcula:
C = U-10000 donde C siempre tendrá 20 bits
```

```
Se necesitaran 20 bits y se representan en 4 bytes
byte 1 y 2
Los primeros 6 bits son fijos 110110
Los restantes 10 bits son los primeros 10 de C
byte 3 y 4
primeros 6 bits son fijos 110111
Los restantes 10 bits son los últimos 10 de C
```

3.3.4.1 Ejemplo

El código Unicode es el 1D11E C = 10000 - 1D11E = 0D11E16 = 0000 1101 0001 0001 1110 Byte 1 y 2: 1101100000 1101 00 Byte 3 y 4: 11011101 0001 1110 Representación Unicode: 11011000001101 00110111010001 11102 = D834 DD1E

3.3.5 UTF - 32

Es la mas simple de las 3 formas ya que todos los caracteres se representan en 32 bits en forma idéntica a como se codifican en la tabla Unicode.

3.3.6 Decodificación y detección de errores

3.3.6.1 UTF-8

Al decodificar una tira de bits codificados en UTF-8 se mira mirar el primer bit Si es 0, entonces el caracter a decodificar está en los siguientes 7 bits. Si es 1: Si el siguiente 0, es un error. Sino hay que contar cuantos unos (1) en total hay antes del primer 0. Si son más de 4, es un error. Sino, los siguientes N bytes (siendo N la cantidad de unos encontrados) deben comenzar con los bits 10, sino, es error. El código Unicode final se arma con los últimos 7-N bits del primer byte seguido de los últimos 6 bits de los bytes siguientes.

3.3.6.2 UTF-16

Para decodificar una tira de bits codificados en UTF-16 se comienza evaluando los primeros 6 bits del primer byte. Si son distintos de 110110, es un dato coficado en 2 bytes y se lo busca en la tabla Unicode Sino, se verifican los primeros 6 bits del tercer byte: Si son distintos de 110111 es un error. Sino se arma el código Unicode con los últimos 2 bits del primer byte, seguidos por los 8 bits del segundo byte. A esto se le suman los últimos 2 bits del siguiente byte, mas los 8 bits del último byte.

4 Maquina Elemental

4.1 Definiciones

- Data: Data is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized.
- Information: When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information
- Bit: Unidad de información
- Digito binario: unidad de información más pequeña con la que pueden trabajar los dispositivos electrónicos que constituyen una computadora digital.
- Biestable: Un biestable, es un multivibrador capaz de permanecer en uno de dos estados posibles durante un tiempo indefinido en ausencia de perturbaciones.
- Byte: Unidad de información de base utilizada en computación y en telecomunicaciones, y que resulta equivalente a un conjunto ordenado de ocho bits
- Proceso: Serie de operaciones llevadas acabo por una computadora.
- Computadora: Es una máquina que consta de elementos mecánicos, eléctricos y electrónicos capaz de procesar gran cantidad de información a alta velocidad.

4.2 Clasificación de computadoras según su generación

4.2.1 Generación 0 - Computadoras Mecánicas (1642 - 1945)

- Pascal
  - Máquina mecánica de cálculo 1642
  - Babbage
- Máquina de diferencias 1790
  - Máquina analítica (propósito general) Primera programadora (ADA) 1834
- Aiken
  - Máquina analítica Mark I (Harvard 1944)

4.2.2 Generación 1 - Tubos de vacío (1945-1955)

- Turing
  - Colossus - Computadora electrónica (1943)
- Mauchley/Eckert
  - ENIAC - Programable por switches (1946)
- Von Neumann
  - IAS Machine - Programa almacenado (1952)

4.2.3 Generación 2 - Transistores (1955-1965)

- DEC
  - PDP 1 - Primera minicomputadora (1961)
  - PDP 8 - Primera mini masiva / Omnibus (1965)
- IBM
  - 7094 - Dominio en uso cientifico (1962)
- CDC
  - 6600 - Primera supercomputadora (1964)
- Burroughs
  - B5000 - Primera computadora diseñada para un lenguale de alto nivel (1963)

4.2.4 Generación 3 - Circuitos Integrados (1965 - 1980)

- DEC
  - PDP 11 - Minicomputadora dominante (1970)
- IBM
  - SYSTEM/360 - Familia de computadoras (1964)
    - Uso comercial y cientifico
    - Distintos modelos compatibles entre sí con distintas capacidades
    - multiprogramación
    - simulaba otras arquitecturas con microarquitectura programada (7094 y 1401)

4.2.5 Generación 4 - Integración a muy gran escala VLSI (1980 - ?)

- IBM
  - PC - Comienzo era computadores personales (1981)
    - Intel 8088
    - MS-DOS
- Apple
  - Apple Lisa - Primera computadora con GUI (1983)
- DEC
  - Alpha - Primera computadora RISC de 64 bits (1992)
- COMMODORE/ATARI
  - Computadoras hogareñas sin estandar

4.2.6 Generación 5 - Computadoras "invisibles"

- Apple
  - Newton - Primera palmtop (1993)
- Computadores embebidos
  - Relojes inteligentes
  - Celulares
  - Electrodomesticos

5 Arquitectura de Von Neumann

5.1 Principios básicos de Von Neumann

INSERTAR GRAFICO

5.1.1 Programa almacenado

Tanto las instrucciones como los datos que en ellas se usan residen en una misma memoria

5.1.2 Ruptura de secuencia

Debe existir una instrucción que permita a la máquina no seguir con la secuencia de ejecución

5.2 Definiciones

- Compuerta: Son circuitos electrónicos biestables unidireccionales, es decir, permiten el pasaje de información en un sólo sentido.
- Registro: memoria muy rápida " que permite almacenar una cierta cantidad de bits
- Celda de memoria: Contenido/Direccion
- Bus de Datos: mueve la información por los componentes de hardware internos y externos del sistema tanto de entrada como de salida (teclado, Mouse etc.)
- Bus de Direcciones: ubica los datos en la memoria teniendo relación directa con los procesos de la CPU.
- Bus de Control: marca el estado de una instrucción que fue dada a la PC.

/*Proceso de transferencia de información*/

6 Maquina Abacus

/*Grafico Maquina Abacus*/

6.1 Elementos

- Dispositivo de Entrada/Salida
- Unidad Arimetico Logica : Acumulador
- Unidad de Control: Secuenciador de Instrucciones, Registro de próxima instrucción, Registro de Instrucción.
- Memoria: Registro de Dirección de Memoria, Registro de Memoria, Celda.
- Buses: Bus de Direcciones, Bus de Datos
- Compuertas

6.1.1 Relaciones

Tamaño RPI = Tamaño RDM = Tamaño Op = Cantidad de Celdas Direccionables Tamaño AC = Tamaño RI = Tamaño RM = Longitud de Instrucción = Longitud de Celda

6.2 Uso de elementos

- Memoria: Operaciones
- Unidad Arimetica y Logica: Operaciones
- Unidad de Control - Instrucciones
  - Fase de búsqueda
  - Fase de ejecución

6.2.1 Fase de busqueda

La unidad de control ordena la transferencia del contenido del RPI al RDM y envía a la memoria una orden de lectura. El contenido de la celda de memoria queda almacenado en el RM, luego la Unidad de Control ordena la transferencia del contenido del RM al RI, pudiendo entonces los circuitos especializados analizar el código de operación de la instrucción. Finalmente se prepara el RPI para ejecutar la próxima instrucción. Para realizar la búsqueda de un operando, una vez que la Unidad de control analiza el contenido del código de operación, ordena la transferencia del contenido del campo operando del RI al RDM y luego envía una orden de operación de lectura. El operando buscado queda disponible en el RM.

6.2.2 Fase de ejecución

la ejecución de cada instrucción implica el movimiento de los datos y como estos pasos se realizan en forma secuencial y ordenada la UCP sigue las señales dadas por el reloj del sistema. Se ha indicado más arriba que esta fase depende exclusivamente del tipo de tarea que se deba realizar, por ende se analizarán casos particulares. Suma: se debe sumar el contenido del RM al contenido del acumulador Carga: se debe almacenar en el acumulador un dato contenido en memoria Almacenamiento: se debe "guardar" en memoria el contenido del acumulador Bifurcación: se debe "saltar" a la dirección indicada en la instrucción. La dirección de bifurcación debe ser transferida al RPI (para buscar la próxima instrucción a ejecutar).

7 Maquina Super Abacus

/*Grafico Maquina Super Abacus*/

7.1 Características Principales

- Registros Generales (Datos o direcciones)
- No posee RPI, R0 contiene la dirección de la próxima instrucción. Incremento vía sumador.
- Máquina de dos direcciones (dos operandos)
- UAL permite hacer cálculos con direcciones y datos

7.2 Formato de Instrucciones

/*Gráfico de formato de instrucciones*/

7.2.1 Fase de busqueda

A escribir

7.2.2 Fase de ejecucion

- Sumar Registro
- Sumar Inmediato
- Sumar palabra en memoria (Registro Indirecto)
- Sumar palabra en memoria (desplazamiento)

7.2.3 Adicional

¿Como se da cuenta la maquina Superabacus en el llenado de los operandos? Y la respuesta es la siguiente, cada instrucción Sumar tiene un código de operación diferente en cada caso.

8 Arquitectura del conjunto de Instrucciones

8.1 Definiciones

- Arquitectura de computadoras: Son las características computacionales visibles al programador, es decir, los atributos que tienen impacto directo en la ejecución lógica de un programa.

- Organización de computadoras: Implementación de la arquitectura (microarquitectura). Define las unidades operativas y sus interconexiones (señales de control, interfaces entre el CPU y los periféricos, tecnología de memoria, trayecto de datos, etc.)

8.2 Modelo de capas

Software	
Level 6	Problem Oriented Language Level (Translation - Compiler)
Level 5	Assembly Language Level (Translation - Assembler)
Level 4	Operating System Machine Level (Partial Interpretation)
Level 3	Instruction Set Architecture Level (Interpretation/Direct Execution)
Hardware	
Level 2	Micro-Architecture Level (registers)
Level 1	Digital logic level (gates)
Level 0	Device Level (individual transistors)

8.3 ISA - Instruction Set Architecture

8.3.1 Arquitectura de computadoras

- Repertorio de instrucciones
- Especificacion de su operacion
- Registros
- Tipos de datos
- Modos de direccionamiento
- Formato de instrucciones
- Memoria
  - Word size
  - Big/Little ENdian
  - Direccionamiento
  - Espacio de direcciones (address space)

8.3.2 Organizacion de computadoras

- Diferentes implementaciones de una misma arquitectura
  - Costos
  - Velocidad de procesamiento
  - Consumo de energía
- Microarquitectura
  - Cableada (hardware - latches, contadores, decodificadores, etc.)
  - Microprogramada (software - microprograma)

8.3.3 Familia de computadoras

- Misma arquitectura base, distintas organizaciones (implementaciones)
- Modelos con prestaciones y precios diferentes pero compatibles entre si

8.3.3.1 Ejemplos

- Intel 80x86
- IBM Mainframe (360/370/390/zArch)
- PowerPC
- Sparc
- Arm

8.3.4 Clasificación de computadoras según su poder de cálculo

8.3.4.1 Supercomputadoras

- extremadamente rápidas
- manejan volúmenes de datos enormes
- poseen miles de cpus
- usos específicos (aplicaciones científicas, simulaciones, campo militar)

8.3.4.1.1 Ejemplos

- Summit USA
- Sierra USA
- Sunway TaihuLight China
- DeepBlue IBM USA

8.3.4.2 Macrocomputadoras o Mainframes

- Muy rápidas
- Manejan volúmenesde datos muy grandes
- Poseen cientos de CPU
- Muy alta disponibilidad
- Usos comerciales y científicos
  - Sistemas de gestión bancarios
  - Telecomunicaciones
  - Instituciones gubernamentales

8.3.4.2.1 Ejemplos

- IBM Mainframe

8.3.4.3 Minicomputadoras o servidores middle range

- Rápidas
- Manejan volúmenesde datos muy grandes
- Poseen decenas de CPUS

- Usos comerciales
  - Empresas medianas y grandes
  - Varios equipos en una misma empresa

8.3.4.3.1 Ejemplos

- IBM RS/6000
- SUn UltraSparc
- HP-NonStop (Itanium)

8.3.4.4 Minicomputadoras / PC

- Uso individual o redes pequeñas a medianas
- Manejan volúmenes de datos no muy grandes
- Poseen uno o varios CPU
- Uso hogareño, educativo, comercial, recreativo
  - estaciones de trabajo en empresas
  - Computadora en el hogar
  - Negocios/Colegios
  - Consolas de videojuego

8.3.4.4.1 Ejemplos

- IBM PC compatible
- Apple Macintosh
- Video consolas

8.3.4.5 Computadoras portátiles / notebooks / netbooks

- Uso individual portátil
- Manejan volúmenes de datos no muy grandes
- Poseen uno o varios CPU
- Uso hogareño, educativo, comercial, recreativo
  - Estaciones de trabajo
  - Computadora en el hogar
  - Negocios/Colegios
  - Consolas de videojuego

8.3.4.5.1 Ejemplos

- Sony
- Toshiba
- HP
- Dell

8.3.4.6 Computadoras de mano

- Uso individual portátil acotado
- Manejan volúmenes de datos pequeños
- Poseen uno o varios CPU
- Uso hogareño, educativo, comercial, recreativo
  - Acopio de datos en vía pública
  - Información personal
  - Visualización de contenidos
  - Consolas de videojuego

8.3.4.6.1 Ejemplos

- Smartphones
- Tablets
- Dispositivos Usables (Samsun gear 2 - reloj inteligente)

9 Arquitectura Harvard

/*Foto arquitecutra */

9.1 Highlights

- Las instrucciones y los datos se almacenan en memorias diferentes
- Hay dos conexiones entre la uniad de control de la CPU y cada sistema de memoria.
- Las instrucciones se pueden cargar al mismo tiempo que los datos (instruction fetch y data access en paralelo por distintos buses)
- Se manejan distintos espacios de direcciones para instrucciones y datos lo que idifculta la programación
- Implementado en algunos microcontroladores PIC y en procesadores de señales digitles (DSP)
- Usado en los DSP para streaming de datos
  - Mayor ancho de banda de memoria
  - Ancho de banda más predecible

9.2 Instruction types

9.2.1 Data handling and memory operations

load, store , move

- Set a register to a fixed constant value.
- Copy data from a memory location to a register, or vice versa (a machine instruction is often called move; however, the term is misleading). Used to store the contents of a register, the result of a computation, or to retrieve stored data to perform a computation on it later. Often called load and store operations.

- Read and write data from hardware devices.

9.2.2 Arithmetic and logic operations

Aritméticas y lógicas add , subtract , multiply , divide (BPF c/s, Decimal, BPFlot and, or , xor

- Add, subtract, multiply, or divide the values of two registers, placing the result in a register, possibly setting one or more condition codes in a status register.
- Increment, decrement in some ISAs, saving operand fetch in trivial cases.
- Perform bitwise operations, e.g., taking the conjunction and disjunction of corresponding bits in a pair of registers, taking the negation of each bit in a register.
- Compare two values in registers (for example, to see if one is less, or if they are equal).
- Floating-point instructions for arithmetic on floating-point numbers.

9.2.3 Control flow operations

branch , jump , compare, call , return

- Branch to another location in the program and execute instructions there.
- Conditionally branch to another location if a certain condition holds.
- Indirectly branch to another location.
- Call another block of code, while saving the location of the next instruction as a point to return to.

9.2.4 Coprocessor instructions

- Load/store data to and from a coprocessor, or exchanging with CPU registers.
- Perform coprocessor operations.

9.2.5 Complex instructions

- Transferring multiple registers to or from memory (especially the stack) at once
- Moving large blocks of memory (e.g. string copy or DMA transfer)

- complicated integer and floating-point arithmetic (e.g. square root, or transcendental functions such as logarithm, sine, cosine, etc.)
- SIMD instructions, a single instruction performing an operation on many homogeneous values in parallel, possibly in dedicated SIMD registers
- performing an atomic test-and-set instruction or other read-modify-write atomic instruction
- instructions that perform ALU operations with an operand from memory rather than a register

9.3 ISA - Instruction Set Architecture

An instruction set architecture (ISA) is an abstract model of a computer. An instruction set architecture is distinguished from a microarchitecture, which is the set of processor design techniques used, in a particular processor, to implement the instruction set. Processors with different microarchitectures can share a common instruction set. For example, the Intel Pentium and the Advanced Micro Devices Athlon implement nearly identical versions of the x86 instruction set, but have radically different internal designs.

9.3.1 Machine instructions characteristics

The operation of the processor is determined by the instructions it executes, referred to as machine instructions or computer instructions. The collection of different instructions that the processor can execute is referred to as the processor's instruction set.

9.3.2 Repertorio de instrucciones

How many and which operations to provide, and how complex operations should be.

9.3.3 Especificación de su operación

- Operation code: Specifies the operation to be performed (e.g., ADD, I/O). The operation is specified by a binary code, known as the operation code, or opcode.
- Source operand reference: The operation may involve one or more source operands, that is, operands that are inputs for the operation.
- Result operand reference: The operation may produce a result.
- Next instruction reference: This tells the processor where to fetch the next instruction after the execution of this instruction is complete.

Source and result operands can be in one of four areas:

9.3.4 Especificación de su operación

- Main or virtual memory: As with next instruction references, the main or virtual memory address must be supplied.
- Processor register: With rare exceptions, a processor contains one or more registers that may be referenced by machine instructions. If only one register exists, reference to it may be implicit. If more than one register exists, then each register is assigned a unique name or number, and the instruction must contain the number of the desired register.
- Immediate: The value of the operand is contained in a field in the instruction being executed.
- I/O device: The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address.

9.3.5 Clasificación según la ubicación de los operandos

- Stack (‘60s a ‘70s
- Acumulador (antes de ‘60s)
- Registro Memoria (‘70s hasta ahora)
- Registro Registro (Load/Store) (‘60s hasta ahora)
- Memoria Memoria (‘70s a ‘80s)

9.3.6 Registros

Number of processor registers that can be referenced by instructions, and their use

9.3.7 Tipos de datos

9.3.7.1 Numéricos

- BPF s/s
- BPF c/s
- BPFlotante (IEEE 754 o
- BCD (decimales)

9.3.7.2 Caracteres

- ASCII
- EBCDIC
- Unicode

9.3.7.3 Datos lógicos

9.3.7.4 Direcciones

9.3.8 Instruction Sets: Addressing Modes

/*458 William Stallings 10th edition*/

9.3.8.1 Immediate

The simplest form of addressing is immediate addressing, in which the operand value is present in the instruction. The advantage of immediate addressing is that no memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle. The disadvantage is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length. Operand = A

9.3.8.2 Direct

A very simple form of addressing is direct addressing, in which the address field contains the effective address of the operand. The technique was common in earlier generations of computers but is not common on contemporary architectures. It requires only one memory reference and no special calculation. The obvious limitation is that it provides only a limited address space. EA = A

9.3.8.3 Indirect

With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range. One solution is to have the address field refer to the address of a word in memory, which in turn contains a full- length address of the operand. This is known as indirect addressing: EA = (A) As defined earlier, the parentheses are to be interpreted as meaning contents of. The obvious advantage of this approach is that for a word length of N, an address space of 2N is now available. The disadvantage is that instruction execution requires two memory references to fetch the operand: one to get its address and a second to get its value. Although the number of words that can be addressed is now equal to 2N, the number of different effective addresses that may be referenced at any one time is limited to 2K, where K is the length of the address field. Typically, this is not a burdensome restriction, and it can be an asset.

9.3.8.4 Register

Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address: EA = R To clarify, if the contents of a register address field in an instruction is 5, then register R5 is the intended address, and the operand value is contained in R5. Typically, an address field that references registers will have from 3 to 5 bits, so that a total of from 8 to 32 general- purpose registers can be referenced. The advantages of register addressing are that (1) only a small address field is needed in the instruction, and (2) no time- consuming memory references are required. As was discussed in Chapter 4, the memory access time for a register internal to the processor is much less than that for a main memory address. The disadvantage of register addressing is that the address space is very limited.

9.3.8.5 Register indirect

Just as register addressing is analogous to direct addressing, register indirect addressing is analogous to indirect addressing. In both cases, the only difference is whether the address field refers to a memory location or a register. Thus, for register indirect address, EA = (R) The advantages and limitations of register indirect addressing are basically the same as for indirect addressing. In both cases, the address space limitation (limited range of addresses) of the address field is overcome by having that field refer to a word- length location containing an address. In addition, register indirect addressing uses one less memory reference than indirect addressing.

9.3.8.6 Displacement

A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing. It is known by a variety of names depending on the context of its use, but the basic mechanism is the same. We will refer to this as displacement addressing: EA = A + (R) Displacement addressing requires that the instruction have two address fields, at least one of which is explicit. The value contained in one address field (value = A) is used directly. The other address field, or an implicit reference based on opcode, refers to a register whose contents are added to A to produce the effective address. We will describe three of the most common uses of displacement addressing:

9.3.8.6.1 Relative addressing

For relative addressing, also called PC-relative addressing, the implicitly referenced register is the program counter (PC). That is, the next instruction address is added to the address field to produce the EA. Typically, the address field is treated as a twos complement number for this operation. Thus, the effective address is a displacement relative to the address of the instruction. Relative addressing exploits the concept of locality. If most memory references are relatively near to the instruction being executed, then the use of relative addressing saves address bits in the instruction.

9.3.8.6.2 Base-register addressing

For base-register addressing, the interpretation is the following: The referenced register contains a main memory address, and the address field contains a displacement (usually an unsigned integer representation) from that address. The register reference may be explicit or implicit. Base-register addressing also exploits the locality of memory references.

9.3.8.6.3 Indexing

For indexing, the interpretation is typically the following: The address field references a main memory address, and the referenced register contains a positive displacement from that address. Note that this usage is just the opposite of the interpretation for base-register addressing. Of course, it is more than just a matter of user interpretation. Because the address field is considered to be a memory address in indexing, it generally contains more bits than an address field in a comparable base-register instruction. Also, we will see that there are some refinements to indexing that would not be as useful in the base- register context. Nevertheless, the method of calculating the EA is the same for both base- register addressing and indexing, and in both cases the register reference is sometimes explicit and sometimes implicit (for different processor types). An important use of indexing is to provide an efficient mechanism for performing iterative operations. Consider, for example, a list of numbers stored starting at location A. Suppose that we would like to add 1

to each element on the list. We need to fetch each value, add 1 to it, and store it back. The sequence of effective addresses that we need is A, A + 1, A + 2, . . . , up to the last location on the list. With indexing, this is easily done. The value A is stored in the instruction's address field, and the chosen register, called an index register, is initialized to 0. After each operation, the index register is incremented by 1

9.3.8.7 Stack

The final addressing mode that we consider is stack addressing. As defined in Appendix I, a stack is a linear array of locations. It is sometimes referred to as a pushdown list or last-in-first-out queue. The stack is a reserved block of locations. Items are appended to the top of the stack so that, at any given time, the block is partially filled. Associated with the stack is a pointer whose value is the address of the top of the stack. Alternatively, the top two elements of the stack may be in processor registers, in which case the stack pointer references the third element of the stack. The stack pointer is maintained in a register. Thus, references to stack locations in memory are in fact register indirect addresses. The stack mode of addressing is a form of implied addressing. The machine instructions need not include a memory reference but implicitly operate on the top of the stack.

9.3.9 Instruction Sets: Addressing Modes

Definición:"Define el despliegue de los bits que componen la instrucción"

9.3.9.1 Components:

- Opcode
- 0 a n operandos
- Modo de direccionamiento de cada operando
- Flags

/*Formato ARM*/ U3 Pág 11 /*Formato x86*/ U3 Pág 12

9.3.10 Clasificación de la ISA según el número de direcciones.

9.3.10.1 3 addresses

- Operand 1, Operand 2, Result
- e.g. a=b+c

9.3.10.2 2 address

- One address doubles as operand and result
- eg a . . = a+c

9.3.10.3 1 address

- Implicit second address (accumulator)

9.3.10.4 0 address

- All addresses are implicitly defined
- Stack based computer

9.4 Memoria

Word size: The "natural" unit of organization of memory. The size of a word is typically equal to the number of bits used to represent an integer and to the instruction length

9.4.1 Big / Little Endian

/* Libro página 452 10th edition*/

9.4.2 Direccionamiento

Addressing: The mode or modes by which the address of an operand is specified.

9.4.3 Espacio de direcciones (address space)

- Memory: The memory space includes system main memory. It also includes PCIe I/O devices. Certain ranges of memory addresses map into I/O devices.
- I/O: This address space is used for legacy PCI devices, with reserved memory address ranges used to address legacy I/O devices.
- Configuration: This address space enables the TL to read/write configuration registers associated with I/O devices.
- Message: This address space is for control

10 Lenguaje ensamblador

10.1 Lenguaje de máquina / código máquina

10.1.1 Definición

Representación binaria de un programa de computadora el cual es leído e interpretado por el computador. Consiste en una secuencia de instrucciones de máquina

10.2 Lenguaje ensamblador

10.2.1 Definición

Representación simbólica del lenguaje de máquina de un procesador específico."

10.3 Transición entre lenguaje de máquina y ensamblador.

/*U4 p1 pag 3*/

10.4 ¿Por qué usarlo aún hoy?

- Debugging y verificación
- Desarrollar compiladores
- Sistemas embebidos
- Drivers de hardware y código de sistema
- Acceder a instrucciones no disponibles en un lenguaje de alto nivel
- Código automodificable

- Optimizar código en tamaño
- Optimizar código en velocidad
- Biblioteca de funciones

10.5 Elementos que lo componen

- Etiquetas
- Mnemónicos
- Operandos
- Comentarios

10.6 Tipos de sentencias

- Instrucciones
- Directivas (pseudoinstrucciones)
  - Macroinstrucciones

10.7 Traducción versus Interpretación

10.7.1 Traductor

- Programa que convierte un programa de usuario escrito en un lenguaje (fuente) en otro lenguaje (destino)
- Clasificación
  - Compiladores
  - Ensambladores

10.7.2 Intérprete

Programa que ejecuta directamente un programa de usuario escrito en un lenguaje fuente

10.8 Ensambladores

- Definición Programa que traduce un programa escrito en lenguaje ensamblador y produce código objeto como salida
- Traducción 1 a 1 a lenguaje máquina
- Hay dos tipos
  - Dos pasadas
  - Una pasada

10.8.1 Primera pasada

- Definición de etiquetas -> Tabla de símbolos
- LC: location counter (empieza en 0 con el 1er byte del código objeto ensamblado)
- Se examina cada sentencia de lenguaje ensamblador
- Determina la longitud de la instrucción de máquina (reconoce opcode + modo de direccionamiento + operandos) para actualizar el LC
- Revisa directivas al ensamblador. -Ejemplo: definición de áreas de storage
- Por cada etiqueta encontrada se fija si está en la tabla de símbolos. Si no lo está la agrega (si es la definición, registra el LC como tal, sino lo registra como referenciando a la etiqueta)

10.8.2 Segunda (Traducción)

- Traduce el mnemónico en el opcode binario correspondiente
- Usa el opcode para determinar el formato de la instrucción y la posición y tamaño de cada uno de los campos de la instrucción
- Traduce cada nombre de operando en el registro o código de memoria apropiado
- Traduce cada valor inmediato en un string binario en la instrucción
- Traduce las referencias a etiquetas en el valor apropiado de LC usando la tabla de símbolos
- Setear otros bits necesarios en la codificación de la instrucción. -Ejemplo: indicadores de modo de direccionamiento, bits de código de condición, etc.

10.8.3 Código objeto

Es la representación en lenguaje de máquina del código fuente programado. Es creado por un compilador o ensamblador y es luego transformado en código ejecutable por el linkeditor"

10.9 Más allá del ensamblado

10.9.1 Linker

Programa utilitario que combina uno o más archivos con código objeto en un único archivo que contiene código ejecutable o cargable

10.9.2 Loader

Rutina de programa que copia un ejecutable a memoria principal para ser ejecutado
/*pag 3 part2*/

10.9.3 Dos problemas a resolver

- Direcciones externas
  - Existen direcciones en el código objeto que no se pueden resolver en tiempo de ensamblado
- Reubicabilidad
  - ¿Por qué es necesaria?
    - No se sabe que otro programa habrá en memoria a la vez
    - Swap a disco en un entorno de multiprogramación

10.10 Código objeto

- Estructura interna
  - Identificación: nombre del módulo, longitudes de las partes del módulo
  - Tabla de punto de entrada: lista de símbolos que pueden ser referenciados desde otros módulos
  - Tabla de referencias externas: lista de símbolos usados en el módulo pero definidos fuera de él y sus referencias en el código
  - Código ensamblado y constantes
  - Diccionario de reubicabilidad: lista de direcciones a ser reubicadas
  - Fin de módulo

10.11 Linking

- Estático (linkage editor)
- Dinámico
  - Load time dynamic linking
  - Run time dynamic linking

10.11.1 Estático (linkage editor)

- Cada módulo objeto compilado o ensamblado es creado con referencias relativas al inicio del módulo
- Se combinan todos los módulos objeto en un único load module reubicable con todas las referencias relativas al load module

Generación del load module

1. Construye tabla de todos los módulos objeto y sus longitudes
2. Asigna dirección base a cada módulo en base a esa tabla
3. Busca todas las instrucciones que referencian a memoria y les suma una constante de reubicación igual a la dirección de inicio de su módulo objeto
4. Busca todas las instrucciones que referencian a otros procedimientos e inserta su dirección

/*pag 10 part 2*/

10.11.2 Linking dinámico

Se difiere la linkedición de algún módulo hasta luego de la creación del load module
Dos tipos:

- Load time dynamic linking
- Run time dynamic linking

10.11.3 Load time dynamic linking

1. Se levanta a memoria el load module
2. Cualquier referencia a un módulo externo hace que el loader busque ese módulo, lo cargue y cambie la referencia a una dirección relativa desde el inicio del load module

Ventajas

- Facilita la actualización de versión del módulo externo porque no hay que recompilar
- El sistema operativo puede cargar y compartir una única versión del módulo externo
- Facilita la creación de módulos de linkeo dinámico a los programadores (ej. Bibliotecas .so en Unix)

10.11.4 Run time dynamic linking

- Se pospone el linkeo hasta el tiempo de ejecución
- Se mantienen las referencias a módulos externos en el programa cargado
- Cuando efectivamente se invoca al módulo externo, el sistema operativo lo busca, lo carga y linkea al módulo llamador.

Ventajas

- No ocupo memoria hasta que la necesito (ej. Bibliotecas DLL de Windows)

10.11.5 Loading

- Loading absoluto
  - El compilador/ensamblador genera direcciones absolutas
  - Solo se puede cargar en un único espacio de memoria

  Loading reubicable

  - El compilador/ensamblador genera direcciones relativas al LC=0
  - El loader debe sumar un valor X a cada referencia a memoria cuando carga el módulo en memoria
  - El load module tiene que tener información para saber cuales son las referencia a memoria a modificar (diccionario de reubicación)

Loading por registro base

- Arquitecturas que usan registros base para el direccionamiento
- Se asigna un valor para el registro base asociado a la ubicación en la que se cargó el programa en memoria

Loading dinámico en tiempo de ejecución

- Se difiere el cálculo de las direcciones absolutas hasta que realmente se vaya a ejecutar
- El load module se carga a memoria con las direcciones relativas
- La dirección se calcula solo al momento de ejecutar realmente la instrucción (con soporte de hardware especial)

/*Linking resumen momentos linkeo pag 17 part 2*/ /*Loading resumen de momentos de binding pag 18 part 2*/

10.12 Intel x86

10.13 Arm

11 Componentes de un computador

11.1 Procesador

11.1.1 CISC: Complex Instruction Set Computer

- Pocos registros de procesador (especializados)
- Set de Instrucciones amplio
- Muchas instrucciones para trabaja con memoria
- Microarquitectura en sofware /hardware compleja
- Instrucciones complejas (más de un ciclo de reloj)
- Varios modos de direccionamiento
- Muchos tipos de datos
- Muchos formatos de instrucción (variables ohíbridos)
- Orientado al hardware, compiladores relativamente
- simples (tamaño de código pequeño)

11.1.2 RISC (Reduced Instruction Set Computer)

- Muchos registros de procesador de uso general
- Set de Instrucciones pequeño
- Solo acceso a memoria a través de LOAD/STORE
- Microarquitectura en hardware simple
- Instrucciones simples (un ciclo de reloj)
- Pocos modos de direccionamiento
- Pocos tipos de datos
- Pocos formatos de instrucción (fijos)
- Orientado al software, compiladores relativamente
- complejos (tamaño de código largo)

11.1.3 Paralelismo

El paralelismo es una función que realiza el procesador para ejecutar varias tareas al mismos tiempo, es decir puede realizar varias cálculos simultáneamente, basados en principio de dividir los problemas grandes para obtener varios problemas pequeños.

Limitación de la velocidad del reloj ???

11.1.3.1 Técnicas

11.1.3.1.1 A nivel instrucción

Un programa de ordenador es, en esencia, una corriente de instrucciones ejecutadas por un procesador. Estas instrucciones pueden ser reordenadas y se combinan en grupos que luego se ejecutan en paralelo sin cambiar el resultado del programa. Esto se conoce como paralelismo a nivel de instrucción. Los procesadores modernos tienen tuberías de instrucciones de múltiples etapas. Cada etapa en la tubería corresponde a una acción diferente que el procesador lleva a cabo en el que la instrucción en esa etapa; un procesador con una tubería N-etapa puede tener hasta N diferentes instrucciones en diferentes fases de ejecución. El ejemplo canónico de un procesador segmentado es un procesador RISC, con cinco

etapas: extracción de instrucción, decodificar, ejecutar, acceso a la memoria, y escribir de nuevo. El procesador Pentium 4 con una cartera de 35 etapas.

Este mecanismo consiste en romper el flujo secuencial de instrucciones para simultanear la ejecución de varias en el mismo procesador. Existen diferentes estrategias para lograrlo.

- Pipelining: The execution of an instruction involves multiple stages of operation, including fetching the instruction, decoding the opcode, fetching operands, performing a calculation, and so on. Pipelining enables a processor to work simultaneously on multiple instructions by performing a different phase for each of the multiple instructions at the same time. The processor overlaps operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously. For example, while one instruction is being executed, the computer is decoding the next instruction. This is the same principle as seen in an assembly line.

- Hazards: The model of sequential execution assumes that each instruction completes before the next one begins; this assumption is not true on a pipelined processor. A situation where the expected result is problematic is known as a hazard. Processors that can compute the presence of a hazard may stall[definition needed], delaying processing of the second instruction (and subsequent instructions) until the values it requires as input are ready. This creates a bubble[definition needed] in the pipeline, also partly negating the advantages of pipelining. Some processors can not only compute the presence of a hazard but can compensate by having additional data paths that provide needed inputs to a computation step before a subsequent instruction would otherwise compute them, an attribute called operand forwarding.[5][6] Some processors can determine that instructions other than the next sequential one are not dependent on the current ones and can be executed without hazards. Such processors may perform out-of-order execution.

- Intel 80486 Pipelining An instructive example of an instruction pipeline is that of the Intel 80486. The 80486 implements a five-stage pipeline:

  - Instruction fetch unit
  - Instruction decode unit
  - Calculate the effective address
  - Fetch the operands from memory
  - Instruction execution unit
  - Write back unit

/*6 U5 CPU*/

- Dual pipelining: was introduced in the Intel Pentium processor. This technology allows the processor to break down a command into two shorter commands and execute them simultaneously when it receives a long command. If there are separate tasks that must be completed for a result that are independent of one another, they can be executed simultaneously to save time /*8 U5 CPU*/

- Superescalar: A superscalar implementation of a processor architecture is one in which common instructions—integer and floating-point arithmetic, loads, stores, and conditional branches—can be initiated simultaneously and executed independently. Such implementations raise a number of complex design issues related to the instruction pipeline. Superscalar design arrived on the scene hard on the heels of RISC architecture. Although the simplified instruction set architecture of a RISC machine lends itself readily to superscalar techniques, the superscalar approach can be used on either a RISC or CISC architecture. Whereas the gestation period for the arrival of commercial RISC machines from the beginning of true RISC research with the IBM 801 and the Berkeley RISC I was seven or eight years, the first superscalar machines became commercially available within just a year or two of the coining of the term superscalar. The superscalar approach has now become the standard method for implementing high-performance microprocessors. /*10 U5 CPU*/

11.1.3.1.2 A nivel procesador

- Procesadores paralelos de datos
  - Una sola unidad de control
  - Múltiples procesadores
  - Métodos
    - - SIMD - Single Instruction Multiple data: Múltiples procesadores ejecutan la misma secuencia de pasos sobre un conjunto diferente de datos.
    - Vector processors: la ruta de datos se multiplexa en tiempo entre los elementos del vector de operandos. No ahorra tiempo de proceso, solo permite disminuir el tamaño del código por el uso de instrucciones vectoriales
- Multiprocesadores
  - Múltiples CPUs que comparten memoria común
  - CPUs fuertemente acoplados
  - Diferentes implementaciones

    - Single bus y memoria compartida (centralizada) (UMA Uniform memory access)
    - CPUs con memoria local y memoria compartida (NUMA non uniform memory access)

- Multicomputadores
  - Computadores interconectados con memoria local (memoria distribuida)
  - No hay memoria compartida
  - CPUs ligeramente acoplados
  - Intercambio de mensajes
  - Topologías de grillas, árboles o anillos

11.2 Memoria

11.2.1 Definicion

Computer memory is any physical device capable of storing information temporarily like RAM (random access memory), or permanently, like ROM (read-only memory). Memory devices utilize integrated circuits and are used by operating systems, software, and hardware.

11.2.2 Formado por elementos con distintas cualidades

- Tecnología
- Organización
- Performance
- Costo

11.2.3 Jerarquía de subsistemas de memoria

- Internos al sistema (accedidos directamente por el procesador)
- Registros
- Memoria interna para unidad de control
- Memoria Cache
- Externo al sistema (accedidos por el procesador a través de un módulo de E/S)
- Dispositivos de almacenamiento perifericos, discos, cintas, etc.

11.2.4 Puntos importantes

- Capacidad
- Tiempo de acceso
- Costo

/*Grafico pág 4*/

11.2.5 A medida que se baja la piramide

- Costo por bit decreciente
- Capacidad decreciente
- Tiempo de acceso creciente
- Frecuencia de acceso de la memoria por parte del procesador decreciente

11.2.6 Caracteristicas

- Capacidad
  - bytes/palabras (memoria interna)
  - bytes (memoria externa)
- Unidad de transferencia
- número de líneas eléctricas del módulo de memoria, típicamente el tamaño de palabra o 64, 128, o 256 bits (memoria interna)
- bloques (memoria externa)

11.2.7 Metodos de acceso de unidades de datos

- Acceso secuencial
  - Unidades de datos: registros
  - Acceso lineal en secuencia
  - Se deben pasar y descartar todos los registros intermedios antes de acceder al registro deseado
  - Tiempo de acceso variable
  - ej cintas magnéticas

- Acceso directo
  - Dirección única para bloques o registros basada en su posición física
  - Tiempo de acceso variable
  - ej discos magnéticos
- Acceso aleatorio
  - cada posición direccionable de memoria tiene un mecanismo de direccionamiento cableado físicamente
  - tiempo de acceso constante, independiente de la secuencia de accesos anteriores
  - ej memoria principal y algunas memorias cache
- Acceso asociativo
  - Tipo de acceso aleatorio por comparación de patrón de bits
  - La palabra se busca por una porción de su contenido en vez de por su dirección
  - Cada posición de memoria tiene un mecanismo de direccionamiento propio
  - Tiempo de acceso constante, independiente de la secuencia de accesos anteriores o su ubicación
  - Ej. memorias cache

11.2.8 Parametros de performance

- Tiempo de acceso (latencia)
  - Memorias de acceso aleatorio: tiempo necesario para hacer una operación de lectura o escritura
  - memorias sin acceso aleatorio: tiempo necesario para posicionar el mecanismo de lectura/escritura en la posición deseada
- Tiempo de ciclo de memoria
  - memorias de acceso aleatorio: tiempo de acceso mas el tiempo adicional necesario para que una nueva operacion pueda comenzar
- Tasa de transferencia
  - tasa con la cual los datos son transferidos dentro o fuera de la unidad de memoria
  - memorias de acceso aleatorio: 1/tiempo de ciclo de memoria

- memorias sin acceso aleatorio: $t_n + T_a + n/R$
- donde
  - t_n = tiempo promedio para leer escribir n bits
  - T_a = tiempo promedio de acceso
  - n = numero de bits
  - R = tasa de transferencia, en bits por segundo

11.2.9 Tipos fisicos de memoria

- Memorias semiconductoras (memoria principal y cache)
- Memorias de superficie magnética (discos y cintas)
- Memorias ópticas (medios ópticos)

11.2.10 Caracteristicas fisicas de memoria

- Memorias volátiles: se pierde su contenido ante la falta de energía eléctrica (Ej. algunas memorias semiconductoras)
- Memorias no volátiles: no se necesita de energía eléctrica para mantener su contenido (Ej. memorias de superficie magnéticas y algunas memorias semiconductoras)
- Memorias de solo lectura: (ROM –ReadOnlyMemory) no se puede borrar su contenido (Ej. algunas memorias semiconductoras)

11.2.11 Principio de localidad de referencia

Durante la ejecución de un programa, las referencias a memoria que hace el procesador tanto para instrucciones como datos tienden a estar agrupadas (Ej. loops, subrutinas, tablas, vectores)

11.2.12 Memoria cache

- Memoria semiconductora más rápida (y costosa) que la principal
- Se ubica entre el procesador y la memoria principal
- Permite mejorar la performance general de acceso a memoria principal
- Contiene una copia de porciones de memoria principal
- como funciona

- CPU trata de leer una palabra de la memoria principal
- Se chequea primero si existe en la memoria cache.
  * Si es así se la entrega al CPU
  * Sino se lee un bloque de memoria principal (número fijo de palabras), se incorpora a la cache y la palabra buscada se entrega al CPU

Por el principio de localidad de referencia es probable que próximas palabras buscadas estén dentro del bloque de memoria subido a la cache

11.2.13 Estructura Memoria principal

- 2^n palabras direccionables(direccion única de n-bits para cada una)
- Bloques fijos de Kpalabras cada uno (M bloques)

11.2.14 Estructura Memoria Cache

- mbloques llamados líneas
- Cada linea contiene:
  - Kpalabras
  - Tag(conjunto de bits para indicar qué bloque está almacenado, usualmente una porción de la dirección de memoria principal)
  - Bits de control (Ej. bit para indicar si la línea se modificó desde la última vez que se cargó en la cache)

11.3 Interrupciones

11.3.1 Definicion

Mecanismos por los cuales otros modulos (E/S y memoria) interrumpen el normal procesamiento del CPU

11.3.2 ¿Para que existen?

Para mejorar la eficiencia de procesamiento de un computador

11.3.3 Clases de interrupciones

- programa
- reloj
- e/s
- fallas de hardware

11.3.4 Ciclo de instruccion

- Fetch instruction
- Decode instruction
- Fetch operand
- Execute instruction
- Store result
- ————————> interrupt breakpoint
- process interrupt

11.3.5 Transferencia de control al S.O. (Handler)

/*4pdf*/

11.3.6 Procesamiento de interrupciones

/*5pdf*/
/*6pdf ejemplo*/

11.3.7 Multiples interrupciones

11.3.7.1 Deshabilitar interrupciones (secuencia)

/*7pdf*/

11.3.7.2 Priorizar interrupciones (anidadas)

/*8pdf*/

11.3.7.3 Múltiples interrupciones - ejemplo

Tres dispositivos de E/S

- Línea de comunicación (Prioridad 1)
- Disco (Prioridad 2)
- Impresora (Prioridad 3)

Eventos

- T=10 Interrupción de Impresora
- T=15 Interrupción de línea de comunicación
- T=20 Interrupción de disco

/*10pdf*/

11.4 MODULO DE E/S

11.4.1 Que hace

Conecta a los periféricos con la CPU y la memoria a través del bus del sistema o switch central y permite la comunicación entre ellos

11.4.2 Para que sirve

Oculta detalles de timing, formatos y electro mecánica de los dispositivos periféricos

11.4.3 Por que existe?

- Amplia variedad de periféricos con distintos métodos de operación
- La tasa de transferencia de los periféricos es generalmente mucho más lenta que la de la memoria y procesador
- Los periféricos usan distintos formatos de datos y tamaños de palabra

/*4pdf*/ 230 stalling

11.4.4 Interface interna - bus del sistema

- Datos
- Direcciones
- Control

11.4.5 Interface externa - perifericos

- Datos
- Estado
- Control

11.4.6 Funciones

11.4.6.1 Control and Timing

ontrola flujo de tráfico entre CPU/Memoria y periféricos

11.4.6.2 Comunicación con el procesador

Decodificación de comandos: The I/O module accepts commands from the processor, typically sent as signals on the control bus. For example, an I/O module for a disk drive might accept the following commands: READ SECTOR, WRITE SECTOR, SEEK track number, and SCAN record ID. The latter two commands each include a parameter that is sent on the data bus

- Datos: Data are exchanged between the processor and the I/O module over the data bus.
- Información de estado: Because peripherals are so slow, it is important to know the status of the I/O module. For example, if an I/O module is asked to send data to the processor (read), it may not be ready to do so because it is still working on the previous I/O command. This fact can be reported with a status signal. Common status signals are BUSY and READY. There may also be signals to report various error conditions.
- Reconocimiento de direcciones: Just as each word of memory has an address, so does each I/O device. Thus, an I/O module must recognize one unique address for each peripheral it controls.

11.4.6.3 Comunicación con el dispositivo

- Comandos
- Información de estado
- Datos

11.4.6.4 Buffering de datos

11.4.6.5 Detección de errores

11.4.7 The control of the transfer of data from an external device to the processor might involve the following sequence of steps

1. The processor interrogates the I/O module to check the status of the attached device.
2. The I/O module returns the device status.
3. If the device is operational and ready to transmit, the processor requests the transfer of data, by means of a command to the IO module.
4. The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.
5. The data are transferred from the I/O module to the processor.

/*Diagrama I/O pag 234 Will, 7pdf */

11.4.8 Tecnicas para operaciones de E/S

1. E/S programada
2. E/S manejada por interrupciones
3. Acceso directo a memoria (DMA)

/*pag 237*/

When large volumes of data are to be moved, a more efficient technique is required: direct memory access (DMA). DMA involves an additional module on the system bus. The DMA module (Figure 7.12) is capable of mimicking the processor and, indeed, of taking over control of the system from the processor. It needs to do this to transfer data to and from memory over the system bus. For this purpose, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily. The latter technique is more common and is referred to as cycle stealing, because the DMA module in effect steals a bus cycle. When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:

1. Whether a read or write is requested, using the read or write control line between the processor and the DMA module.
2. The address of the I/O device involved, communicated on the data lines.
3. The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register.
4. The number of words to be read or written, again communicated via the data lines and stored in the data count register.

/*pag 249*/ The processor then continues with other work. It has delegated this I/O operation to the DMA module. The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor. When the transfer is complete, the DMA module sends an interrupt signal to the processor. Thus, the processor is involved only at the beginning and end of the transfer

/* 250 Figure 7.13 DMA and Interrupt Breakpoints during an Instruction Cycle*/

/* 251 Figure 7.14 Alternative DMA Configurations*/

The DMA mechanism can be configured in a variety of ways. Some possibilities are shown in Figure 7.14. In the first example, all modules share the same system bus. The DMA module, acting as a surrogate processor, uses programmed I/O to exchange data between memory and an I/O module through the DMA module. This configuration, while it may be inexpensive, is clearly inefficient. As with processor- controlled programmed I/O, each transfer of a word consumes two bus cycles The number of required bus cycles can be cut substantially by integrating the DMA and I/O functions. As Figure 7.14b indicates, this means that there is a path between the DMA module and one or more I/O modules that does not include the system bus. The DMA logic may actually be a part of an I/O module, or it may be a separate module that controls one or more I/O modules. This concept can be taken one step further by connecting I/O modules to the DMA module using an I/O bus (Figure 7.14c). This reduces the number of I/O interfaces in the DMA module to one and provides for an easily expandable configuration. In both of these cases (Figures 7.14b and c), the system bus that the DMA module shares with the processor and memory is used by the DMA module only to exchange data with memory. The exchange of data between the DMA and I/O modules takes place off the system bus.

11.4.9 I /O Channels and Processors

The Evolution of the I/O Function As computer systems have evolved, there has been a pattern of increasing complexity and sophistication of individual components. Nowhere is this more evident than in the I/O function. We have already seen part of that evolution. The evolutionary steps can be summarized as follows:

1. The CPU directly controls a peripheral device. This is seen in simple microprocessor-controlled devices.

2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts. With this step, the CPU becomes somewhat divorced from the specific details of external device interfaces.

3. The same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, thus increasing efficiency.

4. The I/O module is given direct access to memory via DMA. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.

5. The I/O module is enhanced to become a processor in its own right, with a specialized instruction set tailored for I/O. The CPU directs the I/O processor to execute an I/O program in memory. The I/O processor fetches and executes these instructions without CPU intervention. This allows the CPU to specify a sequence of I/O activities and to be interrupted only when the entire sequence has been performed.

6. The I/O module has a local memory of its own and is, in fact, a computer in its own right. With this architecture, a large set of I/O devices can be controlled, with minimal CPU involvement. A common use for such an architecture has been to control communication with interactive terminals. The I/O processor takes care of most of the tasks involved in controlling the terminals.

11.4.10 Characteristics of I/O Channels

The I/O channel represents an extension of the DMA concept. An I/O channel has the ability to execute I/O instructions, which gives it complete control over I/O operations. In a computer system with such devices, the CPU does not execute I/O instructions. Such instructions are stored in main memory to be executed by a special-purpose processor in the I/O channel itself. Thus, the CPU initiates an I/O transfer by instructing the I/O channel to execute a program in memory. The program will specify the device or devices, the area or areas of memory for storage, priority, and actions to be taken for certain error conditions. The I/O channel follows these instructions and controls the data transfer

Two types of I/O channels are common, as illustrated in Figure 7.18. A selector channel controls multiple high-speed devices and, at any one time, is dedicated to the transfer of data with one of those devices. Thus, the I/O channel selects one device and effects the data transfer. Each device, or a small set of devices, is handled by a controller, or I/O module, that is much like the I/O modules we have been discussing. Thus, the I/O channel serves in place of the CPU in controlling these I/O controllers. A multiplexor channel can handle I/O with multiple devices at the same time. For low-speed devices, a byte multiplexor accepts or transmits characters as fast as possible to multiple devices. For example, the resultant character stream from three devices with different rates and individual

streams A1A2A3A4 c, B1B2B3B4 c, and C1C2C3C4 c might be A1B1C1A2C2A3B2C3A4, and so on.

/*263*/

11.5 ADMINISTRACION DE MEMORIA

11.5.1 Sistema Operativo

Software que administra los recursos del computador, provee servicios y controla la ejecución de otros programas

11.5.2 Algunos servicios que provee

- Schedule de procesos
- Administración de memoria
- Monitor
- Parte residente del Sistema Operativo

/*284*/ In a uniprogramming system, main memory is divided into two parts: one part for the OS (resident monitor) and one part for the program currently being executed. In a multiprogramming system, the "user" part of memory is subdivided to accommodate multiple processes. The task of subdivision is carried out dynamically by the OS and is known as memory management.

11.5.3 Administración de memoria simple

11.5.4 Sistema con uniprogramación

Se divide la memoria en dos partes Monitor del S.O. Programa en ejecución en ese momento

11.5.4.1 Ventajas:

Simplicidad

11.5.4.2 DESVentajas:

- Desperdicio de memoria

- Desaprovechamiento de los recursos del computador

Administración de memoria simple /*281*/

simple batch systems Early processors were very expensive, and therefore it was important to maximize processor utilization. The wasted time due to scheduling and setup time was unacceptable. To improve utilization, simple batch operating systems were developed. With such a system, also called a monitor, the user no longer has direct access to the processor. Rather, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor. To understand how this scheme works, let us look at it from two points of view: that of the monitor and that of the processor. From the point of view of the monitor, the monitor controls the sequence of events. For this to be so, much of the monitor must always be in main memory and available for execution (Figure 8.3). That portion is referred to as the resident monitor. The rest of the monitor consists of utilities and common functions that are loaded as subroutines to the user program at the beginning of any job that requires them. The monitor reads in jobs one at a time from the input device (typically a card reader or magnetic tape drive). As it is read in, the current job is placed in the user program area, and control is passed to this job. When the job is completed, it returns control to the monitor, which immediately reads in the next job. The results of each job are printed out for delivery to the user. Now consider this sequence from the point of view of the processor. At a certain point in time, the processor is executing instructions from the portion of main memory containing the monitor. These instructions cause the next job to be read in to another portion of main memory. Once a job has been read in, the processor will encounter in the monitor a branch instruction that instructs the processor to continue execution at the start of the user program. The processor will then execute the instruction in the user's program until it encounters an ending or error condition. Either event causes the processor to fetch its next instruction from the monitor program. Thus the phrase "control is passed to a job" simply means that the processor is now fetching and executing instructions in a user program, and "control is returned to the monitor" means that the processor is now fetching and executing instructions from the monitor program. It should be clear that the monitor handles the scheduling problem. A batch of jobs is queued up, and jobs are executed as rapidly as possible, with no intervening idle time. How about the job setup time? The monitor handles this as well. With each job, instructions are included in a job control language (JCL). This is a special type of programming language used to provide instructions to the monitor.

11.5.5 Multiprogramming

- Varios procesos de usuario en ejecución a lavez
- Se divide la memoria de usuario entre los procesos en ejecución

- Se comparte el tiempo de procesador entre los procesos en ejecución (timeslice
- Condiciones de finalización:
  - Termina el trabajo
  - Se detecta un error y se cancela
  - Requiere una operación de E/S (suspensión)
  - Termina el timeslice (suspención

The simplest scheme for partitioning available memory is to use fixed- sizepartitions, as shown in Figure 8.13. Note that, although the partitions are of fixed size, they need not be of equal size. When a process is brought into memory, it is placed in the smallest available partition that will hold it. Even with the use of unequal fixed-size partitions, there will be wasted memory. In most cases, a process will not require exactly as much memory as provided by the partition. A more efficient approach is to use variable-size partitions. When a process is brought into memory, it is allocated exactly as much memory as it requires and no more. As this example shows, this method starts out well, but eventually it leads to a situation in which there are a lot of small holes in memory. As time goes on, memory becomes more and more fragmented, and memory utilization declines. One technique for overcoming this problem is compaction: From time to time, the OS shifts the processes in memory to place all the free memory together in one block. This is a time-consuming procedure, wasteful of processor time.

11.5.6 Memory management: partitioning

- Sistema con multiprogramación
- La memoria de usuario se divide en particiones detamaño fijo:
  - Iguales
  - Distintas

Ventajas:

- Permite compartir la memoria entre varios procesos

Desventajas:

- Desperdicio de memoria
- Fragmentación interna (dentro de una partición)
- Fragmentación externa (particiones no usadas)

11.5.7 Memory management: swapping

- Sistema con multiprogramación
- Swapping
- La memoria de usuario se divide en particiones de tamaño variable
- Compactación para eliminar la fragmentación
- Se usa un recurso de hardware (registro de reasignación) para la realocación
- Realocación dinámica en tiempo de ejecución

Ventajas:

- Permite compartir la memoria entre varios procesos
- Elimina el desperdicio por fragmentación interna.
- Con la compactación se elimina además la fragmentación externa

Desventajas:

- La tarea de compactación es costosa

11.5.8 Memory management: paging

Both unequal fixed-size and variable-size partitions are inefficient in the use of memory. Suppose, however, that memory is partitioned into equal fixed-size chunks that are relatively small, and that each process is also divided into small fixed- size chunks of some size. Then the chunks of a program, known as pages, could be assigned to available chunks of memory, known as frames, or page frames. At most, then, the wasted space in memory for that process is a fraction of the last page.

11.5.8.1 Administración de memoria paginada simple

- Sistema con multiprogramación
- Se divide el address space del proceso en partes iguales (páginas)
- Se divide la memoria principal en partes iguales (frames)
- Hay una tabla de páginas por proceso

- Hay una lista de frames disponibles
- Se cargan a memoria las páginas del proceso en los frames disponibles (no es necesario que sean contiguos)
- Las direcciones lógicas se ven como número de página y un offset
- Se traducen las direcciones lógicas en físicas con soporte del hardware
- La paginación es transparente para el programador

Ventajas:

- Permite compartir la memoria entre varios procesos
- Minimiza la fragmentación interna (solo existe dentro de la última página de cada proceso)
- Elimina la fragmentación externa

Desventajas

- Se requiere subir todas las páginas del proceso a memoria
- Se requieren estructuras de datos adicionales para mantener información de páginas y frames

11.5.8.2 Administración de memoria paginada simple

- Sistema con multiprogramación
- Solo se cargan las páginas necesarias para la ejecución de un proceso
- Cuando se quiere acceder a una posición de memoria de una página no cargada se produce un page fault
- El page fault dispara una interrupción por hardware atendida por el sistema operativo
- Se levanta la página solicitada desde memoria secundaria (memoria virtual)
- Algoritmos para reemplazo de páginas
- Thrashing : el CPU pasa más tiempo reemplazando páginas que ejecutando instrucciones

Ventajas

- No es necesario cargar todas las páginas de un proceso a la vez
- Maximiza el uso de la memoria al permitir cargar más procesos a la vez
- Un proceso puede ocupar más memoria de la efectivamente instalada en el computador

Desventajas

- Mayor complejidad por la necesidad de implementar el reemplazo de páginas

11.5.9 Administración de memoria por segmentación

- Sistemas con multiprogramación
- Generalmente visible al programador
- La memoria del programa se ve como un conjunto de segmentos (múltiples espacios de direcciones)
- Los segmentos son de tamaño variable y dinámico
- El sistema operativo administra una tabla de segmentos por proceso
- Permite separar datos e instrucciones
- Permite dar privilegios y protección de memoria como por ej . lectura, escritura, ejecución. (segmentation faults como mecanismos de excepción de hardware para accesos indebidos)
- Las referencias a memoria se forman con un número de segmento y un offset dentro de él. Con ayuda de hardware (MMU
- Memory Management Unit) se hacen las traducciones de las direcciones lógicas a físicas
- Se pueden usar para implementar memoria virtual (solo se suben a memoria física algunos segmentos por proceso)

Ventajas:

- Simplifica el manejo de estructuras de datos con crecimiento
- Permite compartir información entre procesos dentro de un segmento

- Permite aplicar protección/privilegios sobre un segmento fácilmente

Desventajas:

- Fragmentación externa en la memoria principal por no poder alojar un segmento
- Hardware más complejo que memoria paginada para la traducción de direcciones

11.5.10 Address Spaces

The x86 includes hardware for both segmentation and paging. Both mechanisms can be disabled, allowing the user to choose from four distinct views of memory:

- Unsegmented unpaged memory: In this case, the virtual address is the same as the physical address. This is useful, for example, in low- complexity, high- performance controller applications.
- Unsegmented paged memory: Here memory is viewed as a paged linear address space. Protection and management of memory is done via paging. This is favored by some operating systems (e.g., Berkeley UNIX).
- Segmented unpaged memory: Here memory is viewed as a collection of logical address spaces. The advantage of this view over a paged approach is that it affords protection down to the level of a single byte, if necessary. Furthermore, unlike paging, it guarantees that the translation table needed (the segment table) is on-chip when the segment is in memory. Hence, segmented unpaged memory results in predictable access times.
- Segmented paged memory: Segmentation is used to define logical memory partitions subject to access control, and paging is used to manage the allocation of memory within the partitions. Operating systems such as UNIX System favor this view.

12 Almacenamiento secundario

12.1 Discos magnéticos

Plato circular construido de un material no magnético, llamado substrato (aluminio o vidrio), cubierto por un material magnetizable

12.1.1 Mecanismos de lectura/escritura magnético

- Cabeza de lectura/escritura única: bobina conductora estática, el disco está girando constantemente debajo de ella. Usado en los viejos discos rígidos y floppy disk

- Escritura : cuando circula electricidad a través de una bobina se produce un campo magnético. Los patrones magnéticos resultantes se graban en la superficie (diferentes patrones para corrientes + y -
- Lectura : un campo magnético que se mueve por una bobina produce corriente eléctrica en ella. Cuando la superficie del disco pasa debajo de la cabeza se genera una corriente de la misma polaridad grabada
- Cabeza de lectura diferenciada de la de escritura
- Tiene un sensor magneto resistivo (MR)
- La resistencia eléctrica del material depende de la dirección de la magnetización del medio que se mueve por debajo
- Se hace pasar una corriente a través del sensor MR y los cambios de resistencia se detectan como señales de voltaje
- Provee mayores densidades de grabación y velocidades de operación que el mecanismo anterior

/*img 187*/

Data are recorded on and later retrieved from the disk via a conducting coil named the head; in many systems, there are two heads, a read head and a write head. During a read or write operation, the head is stationary while the platter rotates beneath it. The write mechanism exploits the fact that electricity flowing through a coil produces a magnetic field. Electric pulses are sent to the write head, and the resulting magnetic patterns are recorded on the surface below, with different patterns for positive and negative currents. The write head itself is made of easily magnetizable material and is in the shape of a rectangular doughnut with a gap along one side and a few turns of conducting wire along the opposite side (Figure 6.1). An electric current in the wire induces a magnetic field across the gap, which in turn magnetizes a small area of the recording medium. Reversing the direction of the current reverses the direction of the magnetization on the recording medium. The traditional read mechanism exploits the fact that a magnetic field moving relative to a coil produces an electrical current in the coil. When the surface of the disk passes under the head, it generates a current of the same polarity as the one already recorded. The structure of the head for reading is in this case essentially the same as for writing and therefore the same head can be used for both. Such single heads are used in floppy disk systems and in older rigid disk systems. Contemporary rigid disk systems use a different read mechanism, requiring a separate read head, positioned for convenience close to the write head. The read head consists of a partially shielded magnetoresistive (MR) sensor. The MR material has an electrical resistance that depends on the direction of the magnetization of the medium moving under it. By passing a current

through the MR sensor, resistance changes are detected as voltage signals. The MR design allows higher-frequency operation, which equates to greater storage densities and operating speeds.

12.1.2 Data Organization and Formatting

- Pistas concéntricas
- El ancho de la pista es igual al ancho de la cabeza lectora/grabadora
- Entre las pistas hay un gap (Intertrack gap) para minimizar errores de desalineamiento de la cabeza e interferencias magnéticas
- La superficie del disco está subdividida en sectores, en general de tamaño fijo (512 bytes)
- Hay un gap (Intersector gap) entre los sectores para evitar errores de sincronización
- CAV (Constant Angular Velocity): el disco gira a velocidad constante
- La cabeza lectora/grabadora puede operar a la misma tasa de transferencia
- Los bits exteriores giran a mayor velocidad que los interiores (velocidad lineal variable)
- Para compensar, los bits exteriores están más espaciados entre sí
- Ventaja : se puede referenciar a cada bloque de información a través de pista/sector
- Desventaja : no se aprovecha el máximo de densidad (bits por pulgada lineal) de la superficie del disco

/*img 188*/ The head is a relatively small device capable of reading from or writing to a portion of the platter rotating beneath it. This gives rise to the organization of data on the platter in a concentric set of rings, called tracks. Each track is the same width as the head. There are thousands of tracks per surface. Figure 6.2 depicts this data layout. Adjacent tracks are separated by gaps. This prevents, or at least minimizes, errors due to misalignment of the head or simply interference of magnetic fields. Data are transferred to and from the disk in sectors (Figure 6.2). There are typically hundreds of sectors per track, and these may be of either fixed or variable length. In most contemporary systems, fixed-length sectors are used, with 512 bytes being the nearly universal sector size. To avoid imposing unreasonable precision requirements on the system, adjacent sectors are separated by intratrack (intersector) gaps. A bit near the center of a rotating disk travels past a fixed point (such as a read– write head) slower than a bit on the outside. Therefore, some way must be found to compensate for the variation in speed so that the head can

read all the bits at the same rate. This can be done by increasing the spacing between bits of information recorded in segments of the disk. The information can then be scanned at the same rate by rotating the disk at a fixed speed, known as the constant angular velocity (CAV). Figure 6.3a shows the layout of a disk using CAV. The disk is divided into a number of pie-shaped sectors and into a series of concentric tracks. The advantage of using CAV is that individual blocks of data can be directly addressed by track and sector. To move the head from its current location to a specific address, it only takes a short movement of the head to a specific track and a short wait for the proper sector to spin under the head. The disadvantage of CAV is that the amount of data that can be stored on the long outer tracks is the only same as what can be stored on the short inner tracks.

12.1.3 Grabación multizona

/*189*/

- La superficie del disco se divide en zonas concéntricas (por lo general 16
- La cantidad de bits por pista dentro de una zona es constante
- Las zonas exteriores contienen más bits por pulgada (más sectores) que las zonas interiores

Ventaja:

- mayor capacidad de almacenamiento

Desventajas:

- mayor complejidad en la circuitería para trabajar con tiempos de lectura/escritura diferentes
- según la zona (la longitud de los bits varía)

Because the density, in bits per linear inch, increases in moving from the outermost track to the innermost track, disk storage capacity in a straightforward CAV system is limited by the maximum recording density that can be achieved on the innermost track. To increase density, modern hard disk systems use a technique known as multiple zone recording, in which the surface is divided into a number of concentric zones (16 is typical). Within a zone, the number of bits per track is constant. Zones farther from the center contain more bits (more sectors) than zones closer to the center. This allows for greater overall storage capacity at the expense of somewhat more complex circuitry. As the disk head moves from one zone to another, the length (along the track) of individual bits changes, causing a change in the timing for reads and writes. Figure 6.3b suggests the nature of multiple zone recording; in this illustration, each zone is only a single track

wide. Some means is needed to locate sector positions within a track. Clearly, there must be some starting point on the track and a way of identifying the start and end of each sector. These requirements are handled by means of control data recorded on the disk. Thus, the disk is formatted with some extra data used only by the disk drive and not accessible to the user.

/**189/ An example of disk formatting is shown in Figure 6.4. In this case, each track contains 30 fixed-length sectors of 600 bytes each. Each sector holds 512 bytes of data plus control information useful to the disk controller. The ID field is a unique identifier or address used to locate a particular sector. The SYNCH byte is a special bit pattern that delimits the beginning of the field. The track number identifies a track on a surface. The head number identifies a head, because this disk has multiple surfaces (explained presently). The ID and data fields each contain an errordetecting code.

12.1.4 Physical Characteristics

Movimiento de la cabeza

- fija: había una cabeza lectora/grabadora por pista (muy costosos, no se usan más) Ej. IBM 2305
- móvil: hay una única cabeza lectora/grabadora por superficie del plato. Se mueve por todas las pistas y está montada en un brazo

Portabilidad

- Discos no removibles: disco rígido (se monta en un disk drive
- Removibles: se puede sacar y poner en la unidad (Ej. floppy disk)

Lados

- Un solo lado: solo es usable una cara
- Dos lados: el recubrimiento magnético está en ambas caras

Platos

- Un solo plato
- Múltiples platos: varios discos en un mismo disk drive
  - Cilindro: conjunto de pistas que están a la misma distancia relativa del entro en los platos de un disk drive

Mecanismo de la cabeza

- Contacto: toma contacto con la superficie del disco (Ej. floppy
- Espacio fijo: se ubica a una posición fija por encima del disco
- Espacio aerodinámico (flotante): se ubica flotante por sobre el disco gracias a la presión de aire que genera la rotación del disco

/*img191*/

12.1.5 Parámetros de performance

- Desempeño del disco: depende del computador, sistema operativo, módulo de E/S y controlador de disco
- Tiempo de seek : tiempo necesario para mover la cabeza lectora/grabadora a la pista deseada
- Tiempo de demora rotacional o latencia: tiempo de espera hasta que el sector deseado pasa por la cabeza lectora/grabadora
- Tiempo de acceso: tiempo necesario para estar enposición para escribir o leer
  - Tiempo acceso = tiempo de seek promedio + tiempo de latencia promedio
- Tiempo de transferencia: tiempo necesario para transferir la información al disco
  - T = b/ rN
  - donde T= tiempo de transf .; b=bytes a transf r=velocidad de rotación en revoluciones por segundo; N=bytes por pista
- Tiempo total lectura/escritura
  - T = Tseek + 1 / 2r + b / rN
    - r: velocidad de rotación en revoluciones por segundo
    - b: bytes a transferir
    - N: bytes por pista

Ejemplo:

- Comparación de tiempos de lectura
- Disco: tiempo seek promedio = 4ms; 15000 RPM;
- sectores de 512 bytes; 500 sectores por pista
- Archivo: 2500 sectores (1,28 MB)

12.1.6 Organización secuencial

12.1.7 Organización aleatorial

12.1.8 RAID

As discussed earlier, the rate in improvement in secondary storage performance has been considerably less than the rate for processors and main memory. This mismatch has made the disk storage system perhaps the main focus of concern in improving overall computer system performance. As in other areas of computer performance, disk storage designers recognize that if one component can only be pushed so far, additional gains in performance are to be had by using multiple parallel components. In the case of disk storage, this leads to the development of arrays of disks that operate independently and in parallel. With multiple disks, separate I/O requests can be handled in parallel, as long as the data required reside on separate disks. Further, a single I/O request can be executed in parallel if the block of data to be accessed is distributed across multiple disks. With the use of multiple disks, there is a wide variety of ways in which the data can be organized and in which redundancy can be added to improve reliability. This could make it difficult to develop database schemes that are usable on a number of platforms and operating systems. Fortunately, industry has agreed on a standardized scheme for multiple-disk database design, known as RAID (Redundant Array of Independent Disks). The RAID scheme consists of seven levels,2 zero through six. These levels do not imply a hierarchical relationship but designate different design architectures that share three common characteristics:

- Vectores de discos que operan en forma independiente y en paralelo
- Se puede manejar un pedido de E/S en paralelo si los datos residen en discos separados
- Hay distintas maneras de organizar la información y agregarle confiabilidad a los datos
- RAID es un estandar que consiste en 7 niveles (0 a 6). Pueden implementarse combinaciones de niveles (ej. RAID 0+1, RAID 5+0, etc.)

12.1.8.1 RAID Características generales

- Es un conjunto de discos que son vistos por el sistema operativo como una única unidad lógica
- Los datos se distribuyen en los discos del vector en un esquema llamado " striping

- Se usa capacidad redundante para guardar información de paridad y garantizar recuperación ante fallas (a excepción de RAID 0

12.1.8.2 RAID Niveles

12.1.8.2.1 Nivel 0 (Stripping

- No incluye redundancia
- Se requieren N discos
- Se distribuyen los datos en el vector de discos en strips (pueden ser sectores, bloques u otra unidad)
- Ventajas:
  - -Simplicidad
  - -Performance
- Desventaja:
  - -Riesgo ante fallos, no hay recuperación posible

12.1.8.2.2 Nivel 1 (Espejado)

- Redundancia por espejado de datos
- Se requieren 2N discos
- Ventajas:
  - -Un pedido de lectura puede resolverse por cualquiera de los dos discos
  - -La escritura se hace en forma independiente en cada disco y no se penaliza
  - -Simple recuperación ante fallas
  - -Alta disponibilidad de datos
- Desventajas:
  - Costo

12.1.8.2.3 Nivel 2 (Redundancia por código de Hamming

- Strips pequeños (un byte o palabra)
- Se calcula redundancia por código autocorrector (ej. Hamming
- Se requieren N + m discos
- Se graban bits de paridad en discos separados
- Se leen/escriben todos los discos en paralelo, en forma sincronizada
- No existe uso comercial
- Ventajas:
  - -Disponibilidad de datos
- Desventaja:
  - -Costos por método de redundancia

12.1.8.2.4 Nivel 3 (Paridad por intercalamiento de bits)

- Solo se usa un disco de paridad
- Se requieren N+1 discos
- La paridad se calcula mediante un bit a través del conjunto individual de bits de la misma posición de todos los discos
- Se leen/escriben todos los discos en paralelo, en forma sincronizada
- Ventajas:
  - -Cálculo sencillo de paridad
  - -No hay impacto significativo de performance ante fallas
- Desventajas:
  - -Controlador complejo

12.1.8.2.5 Nivel 4 (Paridad por intercalamiento de bloques)

- Se accede en forma independiente a cada disco
- Se requieren N+1 discos
- Se puede dar servicio a pedidos de E/S en paralelo
- Se usan strips grandes.
- Los bits de paridad se calculan igual que en RAID 3 y se guarda un strip de paridad
- No hay uso comercial
- Ventajas:
  - -Altas tasas de lectura
- Desventaja:
  - -Dos lecturas y dos escrituras en caso de update de datos
  - -Cuello de botella por disco de paridad

12.1.8.2.6 Nivel 5 (Paridad por intercalamiento distribuido de bloques)

- Se accede en forma independiente a cada disco
- Se requieren N+1 discos
- Los strips de paridad se distribuyen en todos los discos
- Ventajas:
  - Resuelve el cuello de botella del nivel 4
- Desventajas:
  - -Controlador complejo

12.1.8.2.7 Nivel 6 (Doble paridad por intercalamiento distribuido de bloques)

- Se accede en forma independiente a cada disco
- Se requieren N+2 discos
- Se usan dos algoritmos de control de paridad
- Ventajas:
  - -Provee disponibilidad de datos extremadamente alta
- Desventaja:
  - -Controlador complejo
  - -Costos por doble paridad

/*raid comparison 204*/

12.2 Medios Opticos

In 1983, one of the most successful consumer products of all time was introduced: the compact disk (CD) digital audio system. The CD is a nonerasable disk that can store more than 60 minutes of audio information on one side. The huge commercial success of the CD enabled the development of low-cost optical-disk storage technology that has revolutionized computer data storage. A variety of optical-disk systems have been introduced (Table 6.6). We briefly review each of these.

CD-ROM Both the audio CD and the CD-ROM (compact disk read-only memory) share a similar technology. The main difference is that CD-ROM players are more rugged and have error correction devices to ensure that data are properly transferred from disk to computer. Both types of disk are made the same way. The disk is formed from a resin, such as polycarbonate. Digitally recorded information (either music or computer data) is imprinted as a series of microscopic pits on the surface of the polycarbonate. This is done, first of all, with a finely focused, highintensity laser to create a master disk. The master is used, in turn, to make a die to stamp out copies onto polycarbonate. The pitted surface is then coated with a highly reflective surface, usually aluminum or gold. This shiny surface is protected against dust and scratches by a top coat of clear acrylic. Finally, a label can be silkscreened onto the acrylic. Information is retrieved from a CD or CD-ROM by a low-powered laser housed in an optical-disk player, or drive unit. The laser shines through the clear polycarbonate while a motor spins the disk past it (Figure 6.12). The intensity of the reflected light of the laser changes as it encounters a pit. Specifically, if the laser beam falls on a pit, which has a somewhat rough surface, the light scatters and a low

intensity is reflected back to the source. The areas between pits are called lands. A land is a smooth surface, which reflects back at higher intensity. The change between pits and lands is detected by a photosensor and converted into a digital signal. The sensor tests the surface at regular intervals. The beginning or end of a pit represents a 1; when no change in elevation occurs between intervals, a 0 is recorded. Recall that on a magnetic disk, information is recorded in concentric tracks. With the simplest constant angular velocity (CAV) system, the number of bits per track is constant. An increase in density is achieved with multiple zoned recording, in which the surface is divided into a number of zones, with zones farther from the center containing more bits than zones closer to the center. Although this technique increases capacity, it is still not optimal. To achieve greater capacity, CDs and CD-ROMs do not organize information on concentric tracks. Instead, the disk contains a single spiral track, beginning near the center and spiraling out to the outer edge of the disk. Sectors near the outside of the disk are the same length as those near the inside. Thus, information is packed evenly across the disk in segments of the same size and these are scanned at the same rate by rotating the disk at a variable speed. The pits are then read by the laser at a constant linear velocity (CLV). The disk rotates more slowly for accesses near the outer edge than for those near the center. Thus, the capacity of a track and the rotational delay both increase for positions nearer the outer edge of the disk. The data capacity for a CD-ROM is about 680 MB.

- CD: Compact Disk. A nonerasable disk that stores digitized audio information. The standard system uses 12-cm disks and can record more than 60 minutes of uninterrupted playing time.
- CD-ROM: Compact Disk Read-Only Memory. A nonerasable disk used for storing computer data. The standard system uses 12-cm disks and can hold more than 650 Mbytes.
- CD-R: CD Recordable. Similar to a CD-ROM. The user can write to the disk only once.
- CD-RW: CD Rewritable. Similar to a CD-ROM. The user can erase and rewrite to the disk multiple times.
- DVD: Digital Versatile Disk. A technology for producing digitized, compressed representation of video information, as well as large volumes of other digital data. Both 8 and 12 cm diameters are used, with a double-sided capacity of up to 17 Gbytes. The basic DVD is read-only (DVD-ROM).
- DVD-R: DVD Recordable. Similar to a DVD-ROM. The user can write to the disk only once. Only one-sided disks can be used.
- DVD-RW: DVD Rewritable. Similar to a DVD-ROM. The user can erase and rewrite to the disk multiple times. Only one-sided disks can be used.

- Blu-ray DVD: High-definition video disk. Provides considerably greater data storage density than DVD, using a 405-nm (blue-violet) laser. A single layer on a single side can store 25 Gbytes.

/*211*/ /*212*/

Data on the CD-ROM are organized as a sequence of blocks. A typical block format is shown in Figure 6.13. It consists of the following fields:

- Sync: The sync field identifies the beginning of a block. It consists of a byte of all 0s, 10 bytes of all 1s, and a byte of all 0s.
- Header: The header contains the block address and the mode byte. Mode 0 specifies a blank data field; mode 1 specifies the use of an error-correcting code and 2048 bytes of data; mode 2 specifies 2336 bytes of user data with no error-correcting code.
- Data: User data.
- Auxiliary: Additional user data in mode 2. In mode 1, this is a 288-byte errorcorrecting code.

With the use of CLV, random access becomes more difficult. Locating a specific address involves moving the head to the general area, adjusting the rotation speed and reading the address, and then making minor adjustments to find and access the specific sector. CD-ROM is appropriate for the distribution of large amounts of data to a large number of users. Because of the expense of the initial writing process, it is not appropriate for individualized applications. Compared with traditional magnetic disks, the CD-ROM has two advantages:

- The optical disk together with the information stored on it can be mass replicated inexpensively—unlike a magnetic disk. The database on a magnetic disk has to be reproduced by copying one disk at a time using two disk drives.
- The optical disk is removable, allowing the disk itself to be used for archival storage. Most magnetic disks are nonremovable. The information on nonremovable magnetic disks must first be copied to another storage medium before the disk drive/disk can be used to store new information. The disadvantages of CD-ROM are as follows:
- It is read-only and cannot be updated.
- It has an access time much longer than that of a magnetic disk drive, as much as half a second.

12.2.0.1 CD RECORDABLE

To accommodate applications in which only one or a small number of copies of a set of data is needed, the write-once read-many CD, known as the CD recordable (CD-R), has been developed. For CD-R, a disk is prepared in such a way that it can be subsequently written once with a laser beam of modest -intensity. Thus, with a some what more expensive disk controller than for CD-ROM, the customer can write once as well as read the disk. The CD-R medium is similar to but not identical to that of a CD or CD-ROM. For CDs and CD-ROMs, information is recorded by the pitting of the surface of the medium, which changes reflectivity. For a CD-R, the medium includes a dye layer. The dye is used to change reflectivity and is activated by a high-intensity laser. The resulting disk can be read on a CD-R drive or a CD-ROM drive. The CD-R optical disk is attractive for archival storage of documents and files. It provides a permanent record of large volumes of user data.

12.2.0.2 CD REWRITABLE

The CD-RW optical disk can be repeatedly written and overwritten, as with a magnetic disk. Although a number of approaches have been tried, the only pure optical approach that has proved attractive is called phase change. The phase change disk uses a material that has two significantly different reflectivities in two different phase states. There is an amorphous state, in which the molecules exhibit a random orientation that reflects light poorly; and a crystalline state, which has a smooth surface that reflects light well. A beam of laser light can change the material from one phase to the other. The primary disadvantage of phase change optical disks is that the material eventually and permanently loses its desirable properties. Current materials can be used for between 500,000 and 1,000,000 erase cycles. The CD-RW has the obvious advantage over CD-ROM and CD-R that it can be rewritten and thus used as a true secondary storage. As such, it competes with magnetic disk. A key advantage of the optical disk is that the engineering tolerances for optical disks are much less severe than for high-capacity magnetic disks. Thus, they exhibit higher reliability and longer life.

12.2.0.3 Digital Versatile Disk

With the capacious digital versatile disk (DVD), the electronics industry has at last found an acceptable replacement for the analog VHS video tape. The DVD has replaced the videotape used in video cassette recorders (VCRs) and, more important for this discussion, replace the CD-ROM in personal computers and servers. The DVD takes video into the digital age. It delivers movies with impressive picture quality, and it can be randomly accessed like audio CDs, which DVD machines can also play. Vast volumes of data can be

crammed onto the disk, currently seven times as much as a CD-ROM. With DVD's huge storage capacity and vivid quality, PC games have become more realistic and educational software incorporates more video. Following in the wake of these developments has been a new crest of traffic over the Internet and corporate intranets, as this material is incorporated into Web sites. The DVD's greater capacity is due to three differences from CDs (Figure 6.14):

- Bits are packed more closely on a DVD. The spacing between loops of a spiral on a CD is 1.6 μm and the minimum distance between pits along the spiral is 0.834 μm.

  /*214*/

  The DVD uses a laser with shorter wavelength and achieves a loop spacing of 0.74 μm and a minimum distance between pits of 0.4 μm. The result of these two improvements is about a seven-fold increase in capacity, to about 4.7 GB.

- The DVD employs a second layer of pits and lands on top of the first layer. A dual-layer DVD has a semireflective layer on top of the reflective layer, and by adjusting focus, the lasers in DVD drives can read each layer separately. This technique almost doubles the capacity of the disk, to about 8.5 GB. The lower reflectivity of the second layer limits its storage capacity so that a full doubling is not achieved.

- The DVD-ROM can be two sided, whereas data are recorded on only one side of a CD. This brings total capacity up to 17 GB. As with the CD, DVDs come in writeable as well as read-only versions (Table 6.6).

12.2.0.4 High-Definition Optical Disks

High-definition optical disks are designed to store high-definition videos and to provide significantly greater storage capacity compared to DVDs. The higher bit density is achieved by using a laser with a shorter wavelength, in the blue-violet

/*215*/ range. The data pits, which constitute the digital 1s and 0s, are smaller on the highdefinition optical disks compared to DVD because of the shorter laser wavelength. Two competing disk formats and technologies initially competed for market acceptance: HD DVD and Blu-ray DVD. The Blu-ray scheme ultimately achieved market dominance. The HD DVD scheme can store 15 GB on a single layer on a single side. Blu-ray positions the data layer on the disk closer to the laser (shown on the right-hand side of each diagram in Figure 6.15). This enables a tighter focus and less distortion and thus smaller pits and tracks. Blu-ray can store 25 GB on a single layer. Three versions are available: read only (BD-ROM), recordable once (BD-R), and rerecordable (BD-RE).

12.2.0.5 Sony / Panasonic: Tecnología " Archival Disc"

$https://en.wikipedia.org/wiki/Archival_Disc$

- Primera generación
- Segunda generación
- Tercera generación

12.2.0.6 Sony " Optical Disc Archive"

$https://en.wikipedia.org/wiki/Optical_Disc_Archive$

12.3 SSD

One of the most significant developments in computer architecture in recent years is the increasing use of solid state drives (SSDs) to complement or even replace hard disk drives (HDDs), both as internal and external secondary memory. The term solid state refers to electronic circuitry built with semiconductors. An SSD is a memory device made with solid state components that can be used as a replacement to a hard disk drive. The SSDs now on the market and coming on line use NAND flash memory, which is described in Chapter 5.

- Historia
  - Basados en RAM (volátiles – energía auxiliar)
- Texas memory: 16KB (1978)
  - Basados en flash (no volátiles)
- M-Systems (1995)
- Tecnología actual
  - NAND Flash

12.3.1 SSD Compared to HDD

As the cost of flash-based SSDs has dropped and the performance and bit density increased, SSDs have become increasingly competitive with HDDs. Table 6.5 shows typical measures of comparison at the time of this writing. SSDs have the following advantages over HDDs:

- High-performanceinput/output operations per second (IOPS): Significantly increases performance I/O subsystems.
- Durability: Less susceptible to physical shock and vibration.
- Longer lifespan: SSDs are not susceptible to mechanical wear.
- Lower power consumption: SSDs use considerably less power than comparable-size HDDs.
- Quieter and cooler running capabilities: Less space required, lower energy costs, and a greener enterprise.
- Lower access times and latency rates: Over 10 times faster than the spinning disks in an HDD.

Currently, HDDs enjoy a cost per bit advantage and a capacity advantage, but these differences are shrinking.

/*215*/ The interface component in Figure 6.8 refers to the physical and electrical interface between the host processor and the SSD peripheral device. If the device is an internal hard drive, a common interface is PCIe. For external devices, one common interface is USB.

- Controller: Provides SSD device level interfacing and firmware execution.
- Addressing: Logic that performs the selection function across the flash memory components.
- Data buffer/cache: High speed RAM memory components used for speed matching and to increased data throughput.
- Error correction: Logic for error detection and correction.
- Flash memory components: Individual NAND flash chips.

12.3.2 Comparación con discos magnéticos

Ventajas

- Arranque más rápido
- Gran velocidad de lectura y escritura
- Baja latencia de lectura y escritura
- Menor consumo de energía
- Menor producción de calor
- Sin ruido
- Mejor MTBF (tiempo medio entre fallas)
- Mayor seguridad de datos
- Rendimiento determinístico
- Menor peso y tamaño
- Mayor resistencia a golpes, caídas y vibraciones

Desventajas

- Precio xGB
- Menos recuperación ante fallos
- Capacidad
- Vida útil

12.4 MAGNETIC TAPE

12.4.1 Definicion

Magnetic tape is a medium for magnetic recording, made of a thin, magnetizable coating on a long, narrow strip of plastic film.

12.4.2 Medio

- Poliester flexible cubierto de material magnetizable
  - Carretes abiertos
  - Paquetes cerrados (cartuchos)
- Ancho de cinta entre 0.38 cm (0.25 pulgadas) y 1.27 cm (0.5 pulgadas)
- Acceso secuencial a la información: si estoy en el registro 1 y quiero llegar al N tengo que "leer" los N-1 del medio
- Si quiero leer un registro anterior tengo que rebobinar y volver a buscar el registro

12.4.3 Técnicas de grabación

Grabación en paralelo

- Técnica usada originalmente
- Cabeza de grabación estacionaria
- Se graban pistas en paralelo a lo largo de la cinta
- Al principio eran de 9 pistas (8 bits de datos y 1 bit de paridad para detectar errores)
- Luego fueron 18 (palabra) o 36 (doble palabra) pistas

Grabación en serie

- Sistema moderno de grabación
- Cabeza de grabación estacionaria
- Se escriben los datos a lo largo de una pista primero hasta llegar al final de la cinta y luego se pasa a otra
- Grabación en "serpentina"
- Pueden grabarse n pistas adyacentes en simultáneo (n entre 2 y 8)

Grabación helicoidal

- Cabeza de grabación rotatoria
- Símil video casseteras
- Evita problema de movimiento veloz de la cinta de las otras técnicas
- La cinta se mueve en forma lenta mientras que la cabeza rota en forma rápida
- Las pistas pueden estar más cercanas unas a otras

12.4.4 Modos de operación

- Modo start-stop por bloque
  - Viejo uso de grabación por registro/bloque
  - La cinta se usaba para guardar archivos para procesamiento posterior
  - Se podía actualizar un registro/bloque particular siempre y cuando no cambiara su tamaño
  - Los datos se grababan en bloques físicos
  - Entre los bloques había espacios (IRG – Inter Record Gap) para sincronización de la unidad
- Modo streaming
  - Uso para backup o archivo de información
  - No se requiere operación de start-stop por bloque
  - No se requiere actualización de bloques particulares dentro de un archivo
  - Se escriben archivos completos como un "stream" de datos contiguo
  - La información de graba físicamente en bloques pero no se pueden localizar o modificar bloques particulares

12.4.5 Usos y características

- Fue el primer medio de almacenamiento secundario
- Aun es usado para backup y archivo de información (30 años o más de duración) dado su bajo costo por byte y su capacidad de almacenamiento
- Es el medio más lento de la pirámide de jerarquía de memoria
- Marcas físicas en las cintas
  - BOT (Beginning of tape)
  - EOT (End of tape)