

Grado en Ingeniería Informática
2019-2020

Apuntes
Estructura de Computadores

Jorge Rodríguez Fraile¹



Esta obra se encuentra sujeta a la licencia Creative Commons
Reconocimiento - No Comercial - Sin Obra Derivada

¹Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

ÍNDICE GENERAL

I Teoría	3
-----------------	----------

Parte I

Teoría

Introducción.

Estructura computador Von Neumann:

Memoria RAM: Almacena datos e instrucciones, en binario.

Procesador: Contiene un ALU, Banco de Registros (BR) donde está el Contador de Programa (PR) y el Registro de Instrucciones (IR), y Unidad de Control (UC). Su función es ejecutar operaciones de un programa.

Dispositivos de entrada/salida: Para conectar periféricos.

Buses: Líneas que conectan los componentes. Hay buses de datos, de control y de dirección.

Binario: Se emplea para codificar datos e instrucciones.

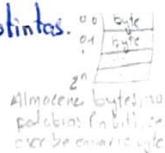
n bit $\rightarrow 2^n$ datos o instrucciones. k objetos $\rightarrow \log_2 k$ bits.

Se basa en Algebra de Boole.

Transistor (Claude Shannon): Deja o no pasar la corriente.

Información: $\left\{ \begin{array}{l} \text{Tamaño mínimo} \rightarrow \text{byte} = 8 \text{ bits} \\ \text{Palabra} \rightarrow n \text{ bits} \end{array} \right.$

$\left\{ \begin{array}{l} 2^n \text{ espacios de dirección/posiciones distintas.} \\ \text{Direcciones de memoria de } n \text{ bits.} \end{array} \right.$



Programa: Secuencia de Instrucciones Maquina (IM): Operación simple.

Codificadas en binario. 96. IM \rightarrow 7 bits $\frac{010100110}{(128 posibles)}$ $\frac{\text{nº de}}{\text{IM}}$ $\frac{\text{Registros}}{\text{Dir. de memoria}}$ $\frac{\text{Números/Datos.}}{\text{Tipos y tamaños}}$

Juego de Instrucciones: Donde vienen definidas las instrucciones y como vienen Codificados para su uso. Ejem: Hoja.

Lenguaje ensamblador: Se utiliza para facilitar la programación, en la que se usa un lenguaje más sencillo, pudiendo utilizar etiquetas para identificar las IM y operaciones (add o fin).

Fases de ejecución de un IM.

- 1) $RI \leftarrow MPE[PC]$ $PC \leftarrow PC + 1$
- 2) Descodifica la instrucción.
- 3) Ejecuta.
- 4) Volver al paso 1)

MAR \leftarrow PC
 Lectura
 MBR \rightarrow Memoria
 PC \leftarrow PC + 1
 RI \leftarrow MBR

Computador: Maquina destinada a procesar datos. Se aplican unas instrucciones y se obtiene unos resultados.

Representación.

Sistemas de representación posicionales

Binario: Base 2 $1 \overset{1}{0} \overset{1}{1}$
 $2^4 + 2^3 + 2^2 + 2^1 = 2^4 + 2^3 + 2^2 + 2^1 = 11$

Hexadecimal: Base 16
 Grupos de 4 bits

Octal: Base 8

Decimal: Base 10

- 20 bits orden de palabra con 60 registros
- que diferencia ha crecido al bit
- Para direcciones se usan 20 bits
- Tamaño de registros, 20 bits
- Bits almacenados en cada posición 8 bits
- Puedo almacenar 2^{10} bytes
- Bits para identificar la dirección 6 bits
- $2^6 = 64$ bytes

n bits $\rightarrow 2^n$ Códigos en binario

Sin signo \rightarrow Binario puro: Solo positivos $0 - 2^{n-1}$

Con signo \rightarrow Positivos y negativos, se dividen los 2^n

Signo magnitud: 1er bit el signo (0 pos. y 1 neg.), el resto indican el número.
 $+ [0, 2^{n-1}]$ - $[-(2^{n-1}-1), 0]$ El 0 duplicado. $\overset{1000}{\text{simétrico}}$

$-2 \Rightarrow -2 + 2^2$ $-3 \Rightarrow -3 + 2^2 - 1$
 $-2 \Rightarrow 1$ $-3 + 4 - 1 = 0$
 $-2 \Rightarrow 001$ $-3 \Rightarrow 000$ Valores en n bits $+ 2^{n-1}$ $+ [0, 2^{n-1}]$ - $[-(2^{n-1}-1), 0]$ El 0 está duplicado.

Complemento a uno: Los positivos en binario puro con $n-1$ bits. 0...

Los negativos, $2^n - \text{valor}$ (lo mismo que cambiar 1 por 0 y 0 por 1). 1...
 $+ [0, 2^{n-1}]$ - $[-(2^{n-1}-1), 0]$ El 0 está duplicado.

Se suma con todos los bits, si hay overflow se le suma (1).

$$\begin{array}{r} \overset{5}{-3+1101} \\ \boxed{10001} \\ + \swarrow \downarrow \searrow \\ \hline 0010 \end{array}$$

Complemento a dos: Los positivos en binario puro con $n-1$ bits. 0...

Los negativos, $2^n - \text{valor}$ (cambiar 1 \rightarrow 0 y 0 \rightarrow 1, sumar 1) $\overset{2^n - x - 1 + 1}{\underset{2^n - x}{\text{asimétrico}}}$
 $+ [0, 2^{n-1}]$ - $[-(2^{n-1}), -1]$ $\overset{-3 \Rightarrow 0011 \text{ car}}{\underset{-1 \text{ car}}{\text{asimétrico}}} \overset{1100 \text{ car}}{\underset{1101 \text{ car}}{\text{asimétrico}}} \text{No duplicado el } 0.$

No se suma el acarreo, se desprecia.

Si me falta un bit para representarlo \rightarrow Desbordamiento. (Al sumar pos. y salga neg.)

Para pasar de menos a más bits, se repite el primer bit. $0011 \rightarrow 0000011$

Representar números reales (IEEE 754)

$$\begin{array}{r} -12.25 \xrightarrow{\text{A binario}} -1100.01 \xrightarrow{\text{notación científica}} -1100.01 \cdot 2^0 \xrightarrow{\text{notación científica normalizada}} -1.10001 \cdot 2^2 \xrightarrow{\text{A Simple.}} \\ \text{neg } \frac{i27+2}{1} \quad \text{mantisa (sin 1)} m \times b^e \\ \hline 11000010 \quad 10001 \dots 0 \end{array}$$

Simple (32 bits, float) 1 bit signo $\begin{cases} 0 & \text{pos} \\ 1 & \text{neg} \end{cases}$ 8 bits exponente $\begin{cases} +128 & \text{exp} \\ -127 & \text{exp} \end{cases}$ 23 bits mantisa $\begin{cases} \text{sin } 1' \\ \text{sin } 1' \end{cases}$

Doble (64 bits, double) 1 bit signo 11 bits exponente $\begin{cases} +1024 & \text{exp} \\ -1023 & \text{exp} \end{cases}$ 52 bits mantisa $\begin{cases} \text{sin } 1' \\ \text{sin } 1' \end{cases}$

Pasar de IEEE 754 a binario decimal:

$$0 \ 10000011 \ 110 \dots 0 \rightarrow +1'11 \cdot 2^4 \rightarrow +11100 \rightarrow +28$$

Signo: 0 positivo +

$$\text{Exponente: } 10000011 = 131 - 127 = 4$$

$$\text{Mantisa: } 110 \dots 0 = 1'11$$

Ejemplo de juego de instrucciones

Instrucción	Descripción
$00 + 10 \Rightarrow 11$ 000CCAAABBBBBXXXXX 0001100100/00000000	Suma el registro AA con el BB y deja el resultado en CC
001AA00000000101	Almacena en el registro AA el valor 00000000101
010AA000000001001	Almacena en el registro AA el valor almacenado en la posición de memoria 00000001001
011AA000000001001	Almacena en la posición de memoria 00000001001 el contenido del registro AA
100000000000001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 000000001001
101AABB000001001	Si el contenido del registro AA es igual al del registro BB se salta a ejecutar la instrucción almacenada en 000001001

Siendo A,B,C,D,E,F = 0 ó 1

S=0
 i=0
 while (i < 10)
 {
 S=S+i
 i=i+1
 }

 dir. IM Leng. Ensamblador
 0000 00100 0...0 S=0 li R0 0
 0001 00101 0...0 i=0 li R1 0
 0010 00110 0..1010 d=11 fi R2 1
 0011 00111 0...01 d=1 fi R3 1
 0100 10101 10...01 Compara bne beq R4,R2,fin
 0101 00000 00..00 S+L add R0,R0,R1
 0110 00001 01..10..0 ade R0/R0,R1
 0111 10000 0...00100 b bucle
 1000 Fin vuelvienta d 0100
 fin

Banco de registros (enteros) del MIPS 32

Nombre registro	Número	Uso
zero	0	Constante 0
at	1	Reservado para el ensamblador
v0, v1	2, 3	Resultado de una rutina (o expresión)
a0, ..., a3	4, ..., 7	Argumento de entrada para rutinas
t0, ..., t7	8, ..., 15	Temporal (<u>NO</u> se conserva entre llamadas)
s0, ..., s7	16, ..., 23	Temporal (se conserva entre llamadas)
t8, t9	24, 25	Temporal (<u>NO</u> se conserva entre llamadas)
k0, k1	26, 27	Reservado para el sistema operativo
gp	28	Puntero al área global
sp	29	Puntero a pila
fp	30	Puntero a marco de pila
ra	31	Dirección de retorno (rutinas)

- ▶ Hay 32 registros
 - ▶ 4 bytes de tamaño (una palabra)
 - ▶ Se nombran con un \$ al principio
- ▶ Convenio de uso
 - ▶ Reservados
 - ▶ Argumentos
 - ▶ Resultados
 - ▶ Temporales
 - ▶ Punteros

Llamadas al sistema

Servicio	Código de llamada (\$v0)	Argumentos	Resultado
print_int	1	\$a0 = integer	
print_float	2	\$f12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer en \$v0
read_float	6		float en \$f0
read_double	7		double en \$f0
read_string	8	\$a0=buffer, \$a1=longitud	
sbrk	9	\$a0=cantidad	dirección en \$v0
exit	10		
print_char	11	\$a0 (código ASCII)	
read_char	12		\$v0 (código ASCII)

Casos especiales IEEE 754:

Signo	exp	mantisa.	Valor
0	0...0	0....0	0
1	0...0	0....0	0
0	1...1	0....0	$+\infty$
1	1...1	0....0	$-\infty$
0	1..1	$m \neq 0$	NaN
1	1..1	$m \neq 0$	NaN
0	0...0	$f \neq 0$	No normalizados
1	0...0	$f \neq 0$	$V = (-1)^s \times 0'f \times 2^{E-127}$
0	$0 < E \leq 255$	M	Normalizados
1	$E \geq 255$	M	$V = (-1)^s \times 1'Mx2^{E-127}$

Nº de valores entre dos números = $2^{n-1} - 1$ - Valor del otro



$2(2^{23}-1)$ números no tienen representación

En complemento a 2 podemos representar $2^{32-1} - 1$. Por tanto tiene más, aunque enteros

$$\begin{aligned} & 0..01-127..-126 \\ & \text{Si no se especifica el signo, es el mayor.} \\ & \text{Para que sea la continuación de los pequeños normalizados: Mayor } (1-2^{-23}) \cdot 2^{-126} \\ & \text{Menor } 2^{-149} \cdot 1.2^{-126} \\ & \text{Pero a } 0 \\ & \text{El mayor } (2-2^{-23}) \cdot 2^{127} = +2^{128} \cdot (1-2^{-24}) \\ & \text{El menor } 1'0 \cdot 2^{-126} \end{aligned}$$

13/9/19

Tema 3: Fund. de la programación en ensamblador.

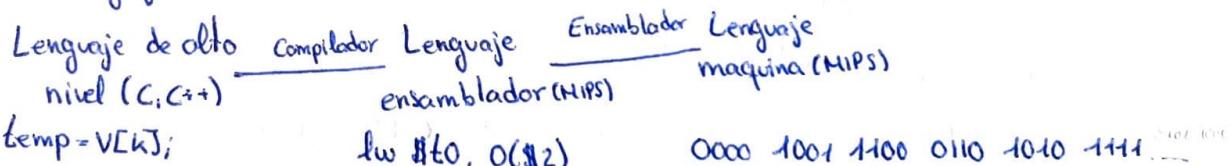
Los datos se representan en binario.

Las instrucciones se representan: MIPS Cod. de operación + Operando (registros, valor inmediato o dirección de memoria)

add: 00100 sssss ttttt lllll llll llll

Se tienen las instrucciones máquina más sencillas posible para poder realizar la mayor cantidad de instrucciones en el menor tiempo (al ser más sencillas)

Niveles de lenguajes:



Arquitectura MIPS 32: Mirar Juego de instrucciones y Lenguaje ensamblador

Procesador de 32 bits. Tiene dos registros, uno para números en Ca2 y otro para números en float (coma flotante). Ambos con sus propias instrucciones. 32 registros de tamaño 4 bytes. Nombrados con \$ al principio.

Registro: \$t0 temporal

Argumento entrada: \$a0

Instrucciones: Tendremos una hoja con el Juego de instrucciones.

Copia de datos (move/li fin ori) de registro a memoria y entre registros.

Aritméticas de enteros o de coma flotante (add/sub/mul/div/rem fin ori ori)

Lógicas (and/or/not/xor fin or or), lo hace los bits. sinsigno: add, no genera excepciones por overflow.

Desplazamientos, lógicos a la derecha (srl) o izq (sll) y antitómico (sra)

sll/srl/sra fin or veces

Rotación a la derecha (rot) o izquierda (rol). rot/rol fin or veces

Comparación. (seq, shl, sge, sgt, sle, slt fin or or)

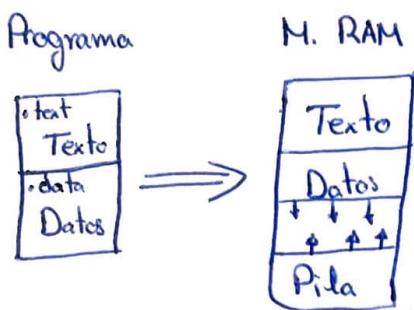
Control de Flyo. Bifurcación o salto condicional (bne si t setea or or dest) o incondicional (j etiqueta)

Condicional (bne, beg, begz, bnez, bgt, bge, blt, ble)

Incondicional (b, j): Salta a la etiqueta.

17/9/19

Mapa de memoria de un proceso:



Los procesos dividen el espacio de memoria en segmentos lógicos para organizar el contenido:

- Segmento pila: Variables locales y contexto de funciones. Se va ampliando cuando lo necesita.
- Segmento datos: Datos estáticos (globales)
- Segmento de código (texto): Código

Programa de ensamblador:

Directivas de ensamblador:

.data	siguiente elemento al segmento dato	li registro, valor inmediato
.text	siguiente elemento al segmento texto	la registro, memoria d (registro que guarda dirección memoria)
.ascii	Cadena de caracteres No terminada en 0	jal etiqueta → Salta jr \$ra → vuelve a ja
.asciz	Almacena cadena de caracteres terminada en nulo	300,30,303
.byte	Almacena byte en memoria consecutivamente	half media palabra
.globl	Declara etiqueta como global	word palabra
.align	Alinea el siguiente en un límite de 2^n	float float
		double double
		space Reserva espacio de bytes
		extern Declara que etiqueta es global de tamaño etiqueta

Definición de datos estáticos:

id: .word 10 ← Como int id=10 c1: .byte 'a' → char c1='a'
etiqueta tipo de Valor
dirección dato
(directiva)

Llamadas al sistema: Lo hace el sistema operativo en vez de el en ensamblador por seguridad, para los dispositivos de entrada y salida. Teclado, pantalla, ...

Invocación: Código de servicio en \$VO

En la hoja la tabla de valores Otros parámetros concretos

Invocación mediante instrucción máquina syscall

li t5 8

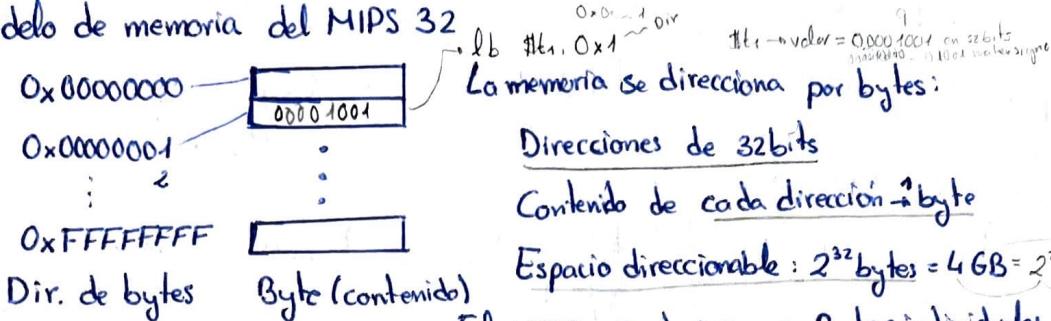
li VO 1 → Código imprimir
move a0 t5 → Parámetro de lo que imprimir en a0
Syscall

Una instrucción en ensamblador se corresponde con una única instrucción máquina.

Una pseudoinstrucción se puede utilizar en el programa en ensamblador, pero no

Se corresponde con una instrucción máquina. li \$t4 => ori \$t4 \$t0 4

Modelo de memoria del MIPS 32



Direcciones de 32bits

Contenido de cada dirección \rightarrow byte

Espacio direccionable: 2^{32} bytes = 4 GB = $2^2 \cdot 2^{20} = 2^{22}$ GB

El acceso puede ser a: Bytes individuales

Palabras (4bytes consecutivos)

Formato de las instrucciones de acceso a memoria:

lw (leer una palabra) donde, lo que gos

sw (guardar una palabra) copia donde, donde

lb Registro, dir. de memoria.

sb leer (guardar un byte) El - significativa

lbu (cargar un byte sin signo)
↳ No completa con signo si no es 0's
 ↳ Completa con extensión de signo + 0's y -1

la Registro, dir. de memoria

Número que representa una dirección. (Hex) 0x0F00002

Etiqueta simbólica que representa la dirección etiq

(Registro) la dirección almacenada en el registro (ojo parentesis) (etiq)

num (Registro) Dirección resultado de sumar num con la dirección almacenada en Registro. (ojo parentesis) x0(16)

Copia en el registro la dirección de memoria, no su contenido.

Ordenamiento de bytes.

Como una palabra ocupa 32bits, 4bytes, ocupa 4 posiciones consecutivas de memoria.

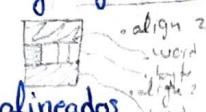
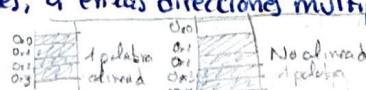
Little-endian (las direcciones más altas tienen signif. más significativas)

Big-endian (el más usado, los más altos tienen signif. más significativas)

Alineación de datos. (Minimizar el nº de accesos a memoria)

Un dato que ocupa k bytes está alineado cuando la dirección D utilizada para accederlo cumple que: $D \bmod k = 0$ Ojo al reservar espacio ocasionar variables globales veracuent ojear cuadros

Los que ocupan 2 bytes en los pares, 4 en las direcciones múltiplo de 4 y 8bytes (double) en las direcciones múltiplo de 8.



En general los computadores no permiten el acceso a datos no alineados.

.align X Alinear a 2^X bytes .align 2 2^2 bytes palabras / .align 3 \Rightarrow 8bytes double float

Tipos de datos en ensamblador: Si no le damos valor, se pone etiqueta: .space n bytes ocup

Booleanos: 1byte 0 false 1 true b1: .byte 0

Caracteres: 1byte C2: .byte 'a'

Enteros: 4bytes = 1 word op1: .word 100

Floot: 4bytes = 1 word = 32bits Simple precisión op2: .word 2.5

Operaciones: add.s, mul.s, sub.s, div.s, abs.s Se guardan en registros \$f0

cuando se ejecutan operaciones

Double: 8 bytes = 64 bits = 2 words op3: .double -10.27

Operaciones: add.d, sub.d, mul.d, div.d, abs.d Ocupan 2 \$fx, van en las partes.

\$f. \$f2. \$f4
0 1 2 3 4 5

Transferencia de datos

Memoria a registro: l.s \$f0 dir1 - Float — Registro a Memoria: s.s \$f0 dir1
l.d \$f2 dir2 - Double — s.d \$f2 dir2

Operaciones con registros (CPU, FPU)

Pasar de \$t (entero) a \$f (float, double) < mfc1

l.mfc1 \$t0 \$f1 Inversa (\$f a \$t): mfc1 \$t0 \$f1
mtc1.d origen final mfc1.d fin ori

Pasar \$f1 a \$f0 (Entre el mismo float dentro) \$f0 → \$f2 → \$f3

↳ mov.s \$f0 \$f1 Float mov.d \$f2 \$f0 Double

Operaciones de conversión (Estando ya en registros \$f, pero sin formato IEEE754)

cvt.s.w \$f2 \$f1 d) mfc1 / mfc1/mov.s/mov.d

Convertir \$f1 entero a \$f2 como float (entero → float)

cvt.w.s (float → entero) cvt.d.w (entero → double) cvt.w.d (double → entero)

cvt.d.s (float → double) cvt.s.d (double → float)

Operaciones de carga:

l.i.s \$f4, 8.0 Cargar float 8.0 a \$f4 l.i.d \$f2, 12.4 Cargar el double 12.4 a \$f2 y \$f3

Vectores: Conjunto de elementos ordenados consecutivamente en memoria.

La dirección del elemento j-ésimo es: dir_inicial + j · tamaño_elemento

Ocupa: n°_elementos · tamaño_elemento Vec.: space 20

VECT
160 → 20.54

entero=4 float double=8
bytes=4 bytes=8
elemento=4bytes

Cadenas de caracteres:

.asciiiz 1 byte por letra y 1 byte de 0's final } Vector de bytes
.ascii 1 byte por letra. } leer con lbyt
(leer con lbyt)
los bytes (el resto)

cl:.ascii "hola"

i: 1byte
h
o
l
a
0
5

Matrices: Una matriz mxn se compone de m vectores de longitud n

Normalmente se almacena por filas.

bajar filas

En la fila (columna)

Dirección del elemento aij: dir_inicial + fila · tamaño_elem · n°_columnas + columna · tamaño_elem

entorno: $\begin{pmatrix} X & X & Y \\ X & X & \otimes \\ \otimes & \otimes & \end{pmatrix}$ $y = 4 \cdot 1 \cdot 2 + 2 \cdot 4$
 $y = 20$
 $y = (1 \cdot 3 + 2) \cdot 4$

dir_inicial + (i · n°_Colum + j) · tamaño_elem

mat: .word 11,12,13

.word 21,22,23

$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{pmatrix}$
mat + (1 · 3 + 0) · 4 → mat + 12

mat
12
21
22
23
mat + 12

Sumar en IEEE 754 / Restar en IEEE 754

1) Insertar el bit implícito (1,)

2) Igualar los exponentes, el menor al mayor. Moviendo la mantisa.

3) Sumar las mantisas, dejar los exponentes igual.

4) Si hay desbordamiento normalizar la mantisa (1,M) y arreglar exponente.

Multiplicar en IEEE 754 / Dividir en IEEE 7541) Sumar exponentes, restando una vez el sesgo (2^{n-1}) → Restarselo a la suma

2) Insertar el bit implícito (1,) → Restarselo al exponente más bajo del valor.

3) Multiplicar la mantisa.

4) Ajustar la mantisa. → $\frac{1.11}{0.11} \cdot 2^{\text{exp}}$ Convenio MIPS: Si no cabe en los decretos se meteran en pila.Parametros: Enteros \$a0,...,\$a3. Coma flotante #f12,...,#f15Resultados: Enteros \$v0,\$v1. Coma flotante #f0,...,#f3A solucion (soje): Enteros \$s0,...,\$s7. Coma flotante #f20,...,#f31. Ojo antes de dejar guardar valor en pila y al final rescatarlo.Temporales: Enteros \$t0,...,\$t9. Coma flotante #f4,...,#f11,#f16,...,#f19Funciones:Llamada: jal nombrFuncion en \$ra se guarda la siguiente instrucción a ejecutar, y salta.Formato: nombrFuncion: num
jr \$ra → Guardar a dónde (Ojo no perder el valor)
ir llamada

Se sacaran \$tx dentro si es una función terminada que no llama a otra, sino usar \$sx, guardando y despues restableciendo el valor que tenían

Dar los parametros en los registros o pila correspondientes antes de llamar la función, el resultado se depositará en los registros conocidos.

Pila: Aumenta hacia direcciones de memoria más bajas. (por eso resto)

Dirección de memoria del pico: #sp → Almacenada en un registro

Meter datos (push): addu \$sp, #sp, -4 → Almacena dato en el espacio de \$sp → \$sp - 4 → Hacer push → Guardar

2 palabras: addu \$sp, -8
sw \$t0, (\$sp)
sw \$t1, 4(\$sp)

Sacar datos (pop): lw \$t0, (\$sp) → Sacar

addu \$sp, #sp, 4 → Restablecer

4/10/19

\$fp Marco de pila: Es el mecanismo que utiliza el compilador para activar los procedimientos (subrutinas) en los lenguajes de alto nivel. Lo construye en la pila el procedimiento llamante y el llamado. Se utiliza en el llamado para:

Al principio en \$sp guarda la dirección previa en el superior y \$fp se pasa a apuntar a la dir. vieja para poder accederlos superiores e inferiores. Finalmente, se pone en \$fp el valor de \$fp y se borra la pila, incluyendo el viejo fp.

Acceder a los parámetros que se pasan en la pila. Ya que \$sp puede cambiar en la ejecución.

Acceder a las variables locales del procedimiento.

Almacena el \$fp previo para cuando acabe la llamada.

\$ra antiguo <--> \$sp+4

Diagrama de la pila:

Arriba	↓	\$sp	↓	\$sp
↓	↓	\$fp	↓	\$fp
↓	↓	\$fp previo	↓	\$fp anterior
↓	↓	\$fp anterior	↓	\$fp anterior
↓	↓	Registros	↓	Registros
↓	↓	salvados	↓	salvados

\$sp+x parámetro del llamante
\$sp-x variables locales

Como diseñar un juego de instrucciones:

Una instrucción en ensamblador se corresponde con una instrucción máquina.

Hay tambien pseudoinstrucciones que corresponden a varias ins. máquina.

Cada instrucción ocupa 32bits en MIPS.

Una instrucción se divide en:

Operación a realizar: + Instrucción y formato

Operando utilizar: Op rs rt rd shmt. func.

Ubicación operando/resultados/ siguiente instrucción.

El formato de una instrucción indica los campos y su tamaño:

Uso de formato sistemático.

Tamaño de un campo limita los valores que codifica.

Según el código de operación se conoce el formato asociado. Se los pone en la parte de la operación.

Cada instrucción pertenece a un formato.

Tipo R: Aritmetica ⁰⁰⁰⁰ op 6b/rs sb/rt sb/rd sb/shamt 5b /func. 6b

Tipo I: transferencia inmediata. op 6b/ rs sb/rt sb/offset 16b 6bits → 64op.

Tipo J: Salto op 6b/ offset 26b

Modos de direccionamiento: Es el procedimiento que permite determinar la ubicación de un operando, un resultado o una instrucción.

Implicito: No se codifica en la instrucción, pero forma parte de ella.

Rápido, pero posible en pocos casos.

Inmediato: El operando forma parte de la instrucción.

Rápido, pero no siempre cabe el valor en una palabra.

Directo a registro: Se encuentra en el registro.

Nº limitado de registros, pero es rápido y al haber pocos se codifica con pocos bits.

Directo a memoria: Se encuentra en memoria, y en la instrucción se codifica la dir.

Acceso a memoria es lento y dir. largas, pero hay un gran espacio de direcciones.

Indirecto de registro: Se indica en la instrucción el registro con la dirección del operando.

Amplio espacio de direcciones, instrucciones cortas.

Relativo a registro base: La posición de memoria almacenada en el registro más desplazamiento.

Relativo a registro índice: Suma a la dir. el valor del registro.

Relativo al contador de programa: Desplazamiento desde la dir. de memoria donde está la instrucción, hasta la posición de memoria indicada en etiqueta.

b etiqueta → El desplazamiento que sumará PC

j etiqueta → Dirección directa a la que saltar (poner en PC)

Juego de instrucciones:

Queda definido por:

Conjunto de instrucciones.	{	Tipos de representación Operación Operando, resultados	Campos Normales cada tipo ocupa 1 palabra, tiene un tamaño, pero puede ocupar más consecutivas
Formato de instrucciones.			

Registros	{	Implicito Indirecto Directo Indirecto Relativo	
Modos de direccionamiento.			

Tipos de datos y formatos.	{		

Distintas formas para la clasificación de un juego de instrucciones:

Complejidad del juego de instrucciones: CISC vs RISC

Modo de ejecución: Pila/Registro/Registro-Memoria, Memoria-Registro.

Tema 4: Procesador.

Su función es ejecutar instrucciones máquina

Funcionamiento básico:

- 1º Leer la dirección de la instrucción en PC, acceder a la dirección y leerla. $PC \rightarrow PC+4$
- 2º Decodificar la instrucción
- 3º Ejecutar la instrucción.

Componentes del procesador:

- Banco de registros: Agrupación de registros, típicamente un número potencia de 2. Los registros almacenan un conjunto de bits, y pueden ser visibles al programador, no visibles o de control y estado.

En el mismo ciclo de reloj
en el bucle no pueden
coincidir datos.
La carga en un registro tarda
un ciclo.

Entradas: C: Entrada de datos.

RA: Indica el registro a sacar por A, y RB lo mismo pero con B.

RC: Indica el registro donde guardar lo que entra por C.

LC: Deja o no cargar datos.

Salidas: A y B

- Unidad Aritmetico Lógica (ALU): Realiza operaciones.

Entradas: A y B los operandos.

Cop: Código de la operación a realizar.

Salidas: R el resultado

Registro de estado: Si es cero, negativo, desbordamiento o acarreo.

- Memoria.

R: Leer

W: Escribir

ADDR: Dirección en la que operar.

Data: Entra el dato a guardar o Sale el dato elegido.

- Unidad de Control (UC): Genera las señales de control que van al resto de componentes. Hay un valor de control para cada ciclo de reloj.
 - Mux
 - Sonda de selección
 - Acceso a memoria
 - Puertas triestado
 - Carga en registro
- Para que no haya cortocircuitos, se utilizan Búfer Triestado:

E	C	S
0	0	X
1	0	Z
0	1	
1	1	0

 Permite multiples conexiones a un mismo punto.

Lenguaje de nivel de transferencia de registros (RT)

Registro 2 \leftarrow Registro 1 Especifica lo que ocurre en el computador.

- Operaciones de transferencia: MAR \leftarrow PC

- Operaciones de proceso: R1 \leftarrow R2 + RTZ

Contador de Programa (PC): Almacena la dirección de la siguiente operación.

Para que pase a la siguiente hay que hacer que se sume 4 \Rightarrow M2, C2

Para cargar una dirección nueva \Rightarrow !C_i (M2=0)

Registro de instrucción (RI): Almacena la instrucción actual.

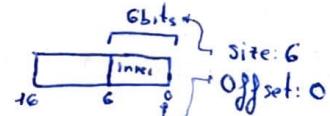
C3 - Cargar instrucción.

Sacar dato en la instrucción (inmediato):

Size: Tamaño del dato en bits

SE: Extensión de signo 1 si / 0 no

Offset: Desplazamiento, donde está el bit de inicio (menos significativo)



27/10/19
Registro de estado: Almacena información (bits de estado) sobre el estado del programa que se está ejecutando en el procesador:

Resultado de la última operación en la ALU: C/V/N/Z

Si está ejecutando en modo núcleo o modo usuario (U)

Si las interrupciones están habilitadas o no (I)

C7 - Del bus interno al SR
SetP, M7 - Jdon; ALU, 1 o 0 u sra
T 8 - del SR al bus interno

Fases de ejecución de una instrucción:

- Lectura de la instrucción, captación o fetch: Leer de PC la dir. della instrucción y pasárla a RI.

C1 MAR \leftarrow PC

T2, CO

C2 MBR \leftarrow MAR, PC \leftarrow PC+4

R, Ta, BW=11, M1, C1 / M2, C2

C3 IR \leftarrow MBR

T1, C3

- Decodificación. C4

- Ejecutar la instrucción: Marcando las instrucciones de la Unidad de Control.

Si hace un salto condicional, ponerlo en un ciclo, como si lo hiciera la UC con pseudocódigo.

* Ojo: el branch (breq) usa desplazamiento PC + PC + 4, etiqueta

no de palabra de despl.
la byte de desplazamiento

Modos de ejecución:

Modo usuario: No puede ejecutar instrucciones privilegiadas (entrada y salida de datos). Si lo intenta se produce una interrupción.

Modo núcleo: Reservado al sistema operativo, puede ejecutar todo el repertorio de instrucciones.

Se indica el modo en el bit U del registro de estado.

Interrupciones: Señal que llega a la unidad de control y que rompe la secuencia normal de ejecución. Detiene el programa actual y se ejecuta otro.

Causas:

- Error en la ejecución.
- Instrucciones ilegales o accesos a memoria ilegales.
- El reloj. Permite que funcionen simultáneamente varios programas y no monopolice la CPU, mediante el planificador que elige el proceso.
- Periférico que solicita la atención del procesador.

Clasificación:

Excepciones hardware sincrónas: División por 0, acceso a memoria ilegal.

Excepciones hardware asíncronas: Fallos o errores en el HW

Interrupciones externas: Periféricos, interrupción del reloj.

Llamadas al sistema: Instrucciones máquina especiales.

Ciclo de reconocimiento de la interrupción:

Durante el ciclo la Unidad de Control realiza los siguientes pasos:

- Comprobar si hay una señal de interrupción activada.
- Si está activada:

Guardar el PC y el SR

 Pasar de modo usuario a modo núcleo.

 Obtener la dirección de la rutina de tratamiento.

 Pasar la dirección al PC, para que se ejecute la rutina.

Rutina de tratamiento de la interrupción:

Forma parte del sistema operativo.

Salva todos los registros del procesador (en una tabla, en la que están todos los programas 

Atiende la interrupción.

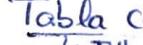
Restaura los registros del procesador utilizados por el programa interrumpido.

Ejecuta una instrucción máquina especial: RETI

Restaura el registro de estado (SR) del programa interrumpido, pasando a modo usuario.

Restaura el PC, de tal manera que siga el programa interrumpido.

Interrupciones vectorizadas:

Tabla con las direcciones de la primera instrucción de las rutinas de tratamiento 

La UC lee el contenido de esa entrada y lo carga al PC

Cada sistema operativo tiene su propia tabla.

Diseño de la Unidad de Control

Para cada instrucción máquina:

1) Definir el comportamiento en lenguaje de transferencia de registros

2) Traducir el comportamiento a señales de control.

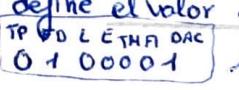
3) Diseñar el circuito que genere las señales de control en cada ciclo.

Técnicas de control:

UC cableada: Rapida pero poco flexible.

Tabla con entradas y salidas. Mediante puertas lógicas y diseño lógico.

UC microprogramada: Emplear una memoria donde almacenar las señales de cada ciclo de cada instrucción.

Microinstrucción: A cada palabra que define el valor de cada señal de control en un ciclo de instrucción.  1 bit por cada señal.

Microprograma: Conjunto ordenado de microinstrucciones. Ejem: fetch, suma...

Microcódigo: Conjunto de los microprogramas de una máquina.

Condiciones:

- Memoria capaz de almacenar el microprograma de cada instrucción.

- Que estén ligadas las instrucciones con su microprograma.

- Mecanismo que permita ir leyendo las sucesivas microinstrucciones y bifurcar.

Arranque del computador:

El reset carga en los registros sus valores preferidos.

PC + dirección de arranque del programa iniciador (en memoria ROM)

Se ejecuta el programa iniciador:

Test del sistema. (POST)

Carga en memoria el cargador del sistema operativo (MBR)

Se ejecuta el cargador del sistema operativo.

Establece opciones de arranque.

Carga el programa de carga.

Se ejecuta el programa de carga:

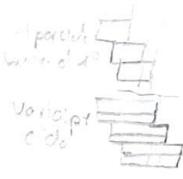
Establece estado inicial para el S.O.

Carga el sistema operativo y lo ejecuta.

Tiempo de ejecución de un programa:

$$\text{Tiempo de ejecución} = \text{Nº Instrucciones} \cdot \text{Ciclos por instrucción} \cdot \text{tiempo por ciclo} + \text{Nº inst.} \cdot \text{Accesos por instrucción a memoria} \cdot \text{tiempo de acceso a memoria} = \text{NI} \cdot \text{CPI} \cdot t_{ciclo} + \text{NI} \cdot \text{AMI} \cdot \frac{t_{ciclo}}{\text{CPU}} \cdot t_{memoria}$$

Paralelismo a nivel de instrucción:



Procesadores segmentados: Utilizan pipeline, varias instrucciones simultáneamente.

Procesadores superescalares: Permite ejecutar varias instrucciones en paralelo.

Procesadores multicore: Varias procesadoras en un encapsulado.

Tema 5: Jerarquía de Memoria.

Tipos de memoria físicas:

- | | |
|-------------------|--|
| Capacidad: ~ | Magnéticas: Información sobre una superficie magnetizada. Discos duros. |
| Tiempo acceso: ms | Ópticas: Información grabada con un láser que genera perforaciones. CD, DVD, ... |
| | Semiconductores: Circuitos electrónicos. RAM, ROM y Flash |

Memoria principal: Capacidad GB, tiempo acceso 40-100 ns

Banco de registros (Procesador): Almacena pocos datos, tiempo de acceso del orden de 1 ns.

Principales características:

Permanencia de los datos: Volátiles / No volátiles

Tipo de operaciones: Lectura ^{RAM} y Escritura ^{ROM} / Solo lectura

Organización: Unidad de almacenamiento (bits, palabras, bloques...)

Modo de acceso: Secuencial / Aleatorio.

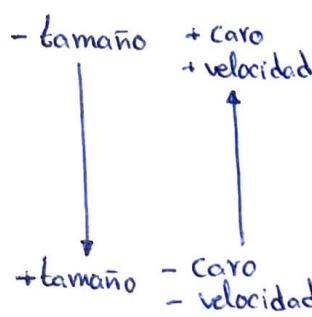
Prestaciones: Tiempo de acceso y Ancho de banda.

Otras: Capacidad y Coste.

Jerarquía de memoria:

Memoria ideal: Mínimo coste y tiempo de acceso, y máxima capacidad.

Memoria real: + Velocidad → - tamaño



- Solo en memoria lo que se necesite en un instante dado.

- Si no está, se copia de un nivel a otro la porción necesaria.

- Cuando no se necesite, se borra la copia realizada.

Memoria de semiconductores:

Memoria de solo lectura (ROM): Persistente y No necesita alimentación.

Memoria de lectura/escritura (RAM): No persistente, Necesita alimentación y más rápida que la ROM. Hay que refrescar la información.

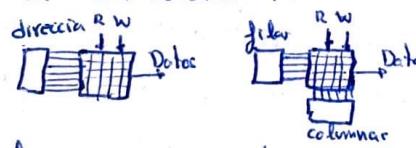
Matriz de memoria semiconductor:

Cada celda almacena un 1 o un 0

Dirección: Posición de una unidad de datos en la matriz.

Capacidad: N° total de unidades de datos que se pueden almacenar.

Tipos de direccionamiento:



Para almacenar un byte, se ponen 8 matrices en paralelo y se lee la misma fila y columna para formar un byte completo.

Memoria RAM:

DRAM Dinámica: Almacena en condensadores. Tiende a descargarse: necesita refrescos periódicos. Más almacenamiento, pero más lenta. Utilizada en memorias principales.

SRAM Estática: Almacena bits como interruptores en on/off. No se descarga. Más rápida, pero menos almacenamiento. Utilizada en memorias cache.

Memoria caché:

Se premia el acceso a posiciones consecutivas de memoria.

Cuando se accede a una posición de memoria se transfieren esos datos y los contiguos, y así se aprovecha el acceso a datos contiguos.

Se transfiere de la memoria principal un bloque de palabras.

La memoria cache es una cantidad pequeña de memoria rápida SRAM, está integrada en el procesador. Más rápida y cara que la memoria principal DRAM.

Funcionamiento general:

- El procesador solicita contenidos de una posición de memoria.

- La cache comprueba si ya están los datos de esta posición:

Si está (Acierto): Se pasa desde la cache al procesador.

Si no está (Fallo): Se transfiere de la memoria principal el bloque de la posición deseada a la cache. Después pasa de la cache al procesador.

Tiempo medio de acceso a cache:

$$T_m = h \cdot T_a + (1-h) \cdot (T_a + T_f) = T_a + (1-h) \cdot T_f$$

T_a : tiempo acceso a cache

T_f : tiempo en tratar el fallo, reemplazar y traer el bloque.

h : tasa de aciertos.

Tasa de aciertos:

$$T = \frac{\text{Ac. Aciertos}}{\text{Ac. Total}}$$

Niveles de memoria cache

L1: Caché interna, la más cercana al procesador. Se suele separar una para datos y otra para instrucciones.
Pequeño tamaño y máxima velocidad.

L2: Caché interna, entre la L1 y L3

Tamaño medio y menor velocidad que L1.

L3: Típicamente último nivel antes de M. Principal.

Tamaño mayor y menor velocidad que L2.

24/11/19

Organización de la memoria principal:

Se divide en bloques lógicos (no físicos) de tamaño k , varía k según la cache.

Ejem: 1GB Mem. Princ. $\frac{1 \cdot 2^{30}}{4 \cdot 4} = \frac{2^{30}}{2^4} = 2^{26}$ bloques.
4 palabras por bloque.
"4 bytes por palabra"

Organización de la memoria caché:

Se organiza en líneas, al bloque de memoria cache se le llama línea de cache.

El tamaño del bloque de M. Princ. es igual al tamaño de la línea.

El tamaño y número de bloques que caben es muy pequeño.

Función de correspondencia:

Algoritmo que determina en qué lugares de la memoria caché se puede almacenar un bloque concreto de la memoria principal.

Un mecanismo que permite saber qué bloque concreto de memoria principal está en una línea de la memoria caché.

- Función de correspondencia directa:

El bloque de memoria k se almacena en la línea: $k \bmod (\# \text{ líneas})$

Cuando se ha llenado la caché se vuelve a escribir encima, sobrescribe.

Formato:	Etiqueta	Línea	Byte
	log ₂ etiqueta	$\frac{\# \text{ líneas}}{2^{\text{bytes de la línea}}}$	$\frac{2^{\text{bytes de la línea}}}{\text{tamaño bloq}}$

- Función de correspondencia asociativa:

Cada bloque puede ir en cualquier línea de la caché.

Formato:	Etiqueta	Byte
	log ₂ etiqueta	$\frac{2^{\text{bytes tamaño bloq}}}{\text{línea bloq}}$

- Función de correspondencia asociativa por conjuntos:

La memoria se organiza en conjuntos de líneas

Una memoria caché asociativa por conjuntos de k vías, cada conjunto k líneas.

Cada bloque siempre se almacena en el mismo conjunto, el bloque B en: $B \bmod (\# \text{ conjuntos})$

Dentro del conjunto puede almacenarse en cualquier línea del conjunto.

+ tamaño conjunto + tasa aciertos - rápido.

Formato:	Etiqueta	Conjunto	Byte
	Sobr	$\frac{2^{\text{conjunto}}}{\text{nº conjuntos}}$	$\frac{2^{\text{bytes tamaño bloq}}}{\text{línea bloq}}$

Más flexible que la directa (permite almacenar a directa variaciones)

Menos costosa que la asociativa (algo + prof)

Sustitución de bloques:

Cuando todas las entradas de la caché contienen bloques de memoria principal:

Hay que seleccionar una línea que dejar libre para traer el bloque:

Directa: No hay posible elección, siempre es el que corresponde. $k \bmod \# \text{ líneas}$

Asociativa: Selecciona una línea de la caché

Asociativa por conjuntos: Selecciona una línea del conjunto.

Algoritmos de sustitución:

FIFO: El que más tiempo lleva en caché.

LRU: El que más tiempo lleva sin usarse.

LFU: El menos usado.

Políticas de escritura:

- Cuando se modifica un dato en MC, hay que actualizar en algún momento la M.P.
- Escritura inmediata: Se escribe en ambos sitios en cuanto se modifica.
 - Post-escritura: Escribe en cache solo, pero antes de borrarlo mira el bit de modificación y si se ha modificado se copia en MP antes.

Tema 6: E/S y dispositivos periféricos.

Periférico: Todo aquel dispositivo externo que se conecta a un procesador a través de la unidad o módulo de entrada/salida. Permiten almacenar información o comunicar el computador con el mundo exterior.

Comunicación: Hombre - máquina, máquina - máquina y medio físico.

Almacenamiento: Acceso directo y Acceso secuencial.

Módulo entrada/salida: Controlador, interfaz entre dispositivo y el procesador.

Anatomía de un disco duro:

Motor: Gira los discos

Discos: Donde se almacena la información.

Cabezas lectoras/escritoras: Leer de los discos y escriben
Módulo de control y mecánica.

Cilindro: Información accedida por todas las cabezas en una rotación.

Pista: Anillo del plato O

Sector: Divisiones del anillo, todas iguales 512 bytes

Bloques: Grupo de sectores

Medidas de la capacidad:

Bits por pulgada cuadrada: Dependen de la cabeza, del medio, la rotación y velocidad del bus.

Pistas por pulgada: Dependen de la cabeza, el medio de grabación, la precisión de la cabeza y la capacidad del disco para girar.

Capacidad de almacenamiento:

Para discos con velocidad angular constante:

$$\text{Capacidad} = \underbrace{n^{\circ} \text{ de superficies}}_{\text{discos} = 2 \text{ super}} \cdot \underbrace{n^{\circ} \text{ de pistas}}_{\text{por disco} = 2} \cdot \underbrace{n^{\circ} \text{ sectores}}_{\text{super} = 1} \cdot \text{bytes por sector}$$

Para discos con múltiples zonas:

$$\text{Cap} = n^{\circ} \text{ de discos} \cdot \text{bytes por sector} \cdot \sum_{i=1}^{n^{\circ} \text{ zones}} (\text{pistas de } i \cdot \text{sectores de } i)$$

En los discos magnéticos:

Cada bit de una pista son múltiples granos magnéticos.

El tamaño de los granos no se pueden reducirse más de 10 nm por el

efecto superparamagnético y la temperatura afecta al estado de los granos.

Tipos de direccionamiento:

Direccionamiento físico: Cilindro - pista - sector

Direccionamiento de bloque lógico: Cada sector se identifica con un bloque y la correspondencia la hace el disco.

Tiempo de acceso:

$$T_{acceso} = T_{busqueda} + T_{latencia} + T_{transferencia}$$

↳ Hacer la ↳ Dentro del dia ↳ Recorrer el sector
cabeza a la llegad principio del sector
pista. ↑
 $T_{busqueda}$

Tiempo de posicionamiento:

$$D_{avergae} = \frac{1}{2} \alpha_{acc} \cdot t_{acc}^2 + \alpha_{dec} \cdot t_{dec}^2$$

Funcionamiento de la memoria cache:

El procesador pasa a la cache una dirección y si la almacena la pasa al procesador.

- Si el dato estaba en la cache se denomina \Rightarrow ACIERTO.

- Si no está el dato, ~~y no se ha podido pasar~~ \Rightarrow FALSO

↳ Pide a memoria el bloque que falta para poderlo pasar. \Rightarrow ACIERTO

Al tiempo total hay que sumarle el tiempo en pasar un bloque por el resto de fello

$$T = \text{accesos cache} \cdot \frac{\text{tiempo cache}}{\text{cache}} + \text{bloques que faltaban} \cdot \frac{\text{tiempo en llevar a cache dato memoria.}}{\text{memoria.}}$$

$$\text{Tasa de acierto} = \frac{4002}{4004} = \frac{\text{Aciertos}}{\text{Total}}$$

$$\text{Total} = \text{Acierto} + \text{Fallo.}$$

$12010 \text{ ns} + 480 \text{ ns} + 800 \text{ ns}$

Acceso cache

$\frac{6005}{6005} \text{ Total acceso} \Rightarrow$

$\frac{\text{Acierto}}{\text{Total}} = \frac{6001}{6005} \cdot 100 = 99.93\%$

$T_{total} = 13290 \text{ ns}$

pr instrucción por acceso directo

$$6003 + 1000 = 7003 \text{ accesos total}$$

$$14010 \text{ ns} + 253 \text{ ns}$$

cache

acceso a memoria RAM

$$\text{Tasa} = \frac{6752}{7005} = 0.9638 \Rightarrow 96.38\%$$

250 fello para llevar los datos del vector.
 3 fello para llevar las instrucciones.
 10 ns / 4 ns = 3
 2º reg. siguiente al 1º
 3º addi se accede al 1º
 253 fello totales.

Tiempo medio de acceso a cache:

Para una memoria con 2 niveles:

$$T_m = h \cdot T_a + (1-h) \cdot (T_a + T_f) = T_a + (1+h) T_f$$

\hookrightarrow Tiempo acceso cache

\hookrightarrow Tiempo total fello, reemplazar bloques y tratar de MP

\hookrightarrow Tasa de acierto.

Dirección
Si acierta
Indirecta, fello, que
necesita cache y principio

Niveles de memoria cache:

L1: La más cercana al procesador.

L2:

LS:

Dispositivos de:

Tema 6: Sistemas Entrada/Salida.

Disp. periférico: Aquel dispositivo externo que se conecta a un procesador a través de las unidades o módulos E/S

Compuesto: Dispositivo: Hardware interno

Módulo E/S: Controlador

Anatomía de un disco duro:

Cuando se lee, se lee como mínimo un sector (512 bytes)

Las pistas exteriores tienen más sectores que las internas, pero todos del mismo tamaño.

Medida de la capacidad:

Datos por pulgada cuadrada: Dependen de la cabeza de lectura/escritura, del medio de grabación, de la rotación del disco y velocidad a la que fluye el bus.

Pistas por pulgada:

Capacidad de almacenamiento:

$$\text{Capacidad} = \text{nº de pistas} \times \dots$$

Direccionalamiento:

Direccionalamiento físico: cilindro-pista-sector

Direccionalamiento de bloques lógicos: Cada sector se identifica con un bloque lógico, la correspondencia la hace el propio disco, con el controlador se traduce de bloques lógicos a direcciones físicas.

Tiempo de acceso

$$T_{\text{acceso}} = T_{\text{busqueda}} + T_{\text{latacua}} + T_{\text{transferencia}}$$

mover la
cabecera
al cilindro

 host que gira
llega al sector
de Estado.

 tiempo en leer el
sector y pasar
los datos.

 T media
vuelt

Ejercicio:

7200 rpm

604 sectores/pista

Tiempo medio de búsqueda: 4ms

$$T_{\text{acceso}} = 4\text{ms} + \frac{8'3}{2} + \frac{8'3}{604} = 8'16\text{ms}$$

medio
vel

Ejercicio:

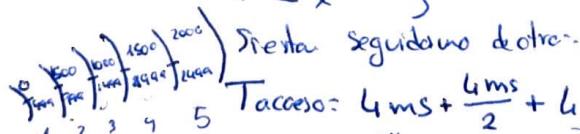
$T_b = 4\text{ms}$ 15000 rpm sectores de 512bytes con 500 sectores por pista.

Se quiere leer 2500 sectores con un total de 1.22MB

$$T_{\text{acceso}} = 2500 \times \left(2\text{ms} + 4\text{ms} + \frac{4}{500} \right) = 15\text{s}$$

15000 - $60 \cdot 10^3 \text{ ms}$
 1 - x

En acceder a 2500 sectores, si están de forma aleatoria, dependiendo el siguiente.



Resta seguidos de otro:

$$T_{\text{acceso}} = 4\text{ms} + \frac{4\text{ms}}{2} + 4\text{ms} = 10\text{ms}$$

mover
cabecera
en pista
para acceder
el proxim

El resto $\sim \frac{1}{2}$ vueltas para las pistas.

$$T_a = (0\text{ms} + 2\text{ms} + 4\text{ms}) \times 4$$

Primer pista $T_a = 24\text{ms}$ el resto de pistas

$$T_{\text{total}} = 10 + 24 = 34\text{ms}$$

entonces en cada secuencia

Ejercicio propuesto:

- 15 L1
- Ins. 32kB 64 bytes línea
- Datos 32kB 8 vías porconjunto.

```
double M[4096][4096]
for (i=0; i<4096; i++)
  for(j=0; j<4096; j++)
    S = S + VC[j][i]
```

$$\text{nº de líneas} = \frac{32\text{ kB}}{64\text{ bytes}} = \frac{2^5 \cdot 2^{10}\text{ bytes}}{2^6 \text{ bytes}} = 2^4 = 16 \text{ líneas.}$$

$$\text{nº de conjuntos} = \frac{16}{8} = 2 \text{ conjuntos.}$$

double: 8bytes

$$\text{nº de elementos por línea} = \frac{64\text{ bytes}}{8\text{ bytes}} = 8$$

0	0
0	1
0	2
:	:
0	4096
1	0
1	1
1	2
:	:

Primer acceso al 0,0 → Fallo Cargo 00 ... 07
 Luego al 1,0 → Fallo Cargo 10 ... 17

511,0 → Fallo
 512,0 → Fallo Se han leído y se han borrado las d 0,0
 4096,0 → Fallo Borrado de 10

No hay ningún acierto.

- Si fuese $VC[i][j]$

0,0 → Fallo	0,0 → ... 0,7
0,1 → Acierto	
0,2 → A	
:	
0,7 → A	
0,8 → Fallo	
:	

7/8 = 88% de aciertos.
7 Aciertos
1 fallo
8 Total

3/12/19

Planificador de disco:

Los dispositivos actuales incluyen un programador de las peticiones a disco cuya objetivo es mejorar el tiempo de respuesta y minimizar los movimientos de la cabeta.

Políticas:

SCAN / / / , Solo hacia un sentido, de extremo a centro
 C-SCAN M M En ambos sentidos. ↗

Look: Se queda en que le pide.
 Para la velocidad de disco se usa el SI en base 10. $300\text{ MB} = 300 \cdot 10^6 \text{ bytes}$

Estados de consumo de energía

Active: Se mueve y funciona

Idle: Gira el disco más lento y cabecillas retiradas

Standby: Se para el giro y cabecillas fuera de su disco.

Memoria Flash flotante

$\rightarrow \downarrow$ No hay electrones $\Rightarrow 1$

$\downarrow \uparrow$ Hay electrones $\Rightarrow 0$

Se puede escribir, poniendo e^-

Borrar se sacan todos los 0, y se pone a 1, y cuando se escriben todos menos el que queramos borrar.

$$1111 \xrightarrow{\text{4 bits de tira}} 1010 \xrightarrow{\text{quinto bit}} 1111 \rightarrow 1011$$

Se utiliza una tensión baja para leer en la puerta de control.

Para borrar se aplica una tensión alta y salen todos los e^- .

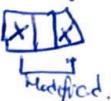
También existen celdas de varios niveles, segun el porcentaje de electrones son: 10, 01, 00, 11.

Según se van utilizando se desgasta el oxígeno que permite que funcione la celda.

Nivelación del desgaste: Para maximizar la duración de la memoria Flash.

Los datos se van moviendo entre bloques para que se desgasten por igual.

Por ejem: Si modificamos un valor lo cambiamos a otra block.



Bus(es):

Ancho: N° de bit simultáneamente

Paralelo $\Rightarrow \downarrow \uparrow$ Mas rápido.

Serie: $\rightarrow 101 \rightarrow$

Frecuencia: N° de bit por unidad de tiempo

$$\text{Ejui: Bus 32 bits con } f=66 \text{ MHz} \text{ Ancho} = \frac{32 \cdot 66}{8} = 264 \text{ MB}$$

Método de arbitraje:

Módulo E/S: Realizan la conexión del procesador con los dispositivos periféricos.

Controlan la velocidad entre dispositivos con distinta velocidad.

Registro de control: ¿Pide acción? Ordenes al procesador.

R. de datos: Los datos del dispositivo

R. de estado: Si ha terminado una

... programada: Mono peltar, la mayoría del tiempo está esperando a que termine, bus ocupado.

... Posible: Se encargado de planificar hacer otras cosas y cuando termina envía una interrupción señal S.O

... DMA: Para acceder al bus a memoria tiene que pedir permiso al procesador con Bus RD si lo permite lo hace enviando Bus ACK