

examen (1).pdf



Juanrajarote



Arquitectura de Computadores



2º Grado en Ingeniería Informática



Escuela Politécnica Superior de Córdoba
Universidad de Córdoba



[Accede al documento original](#)



Escuela de
Organización
Industrial

Contigo que evoluciones.
Contigo que lideras. Contigo que transformas.

**Esto es EOI.
Mismo propósito,
nueva energía.**



Descubre más aquí



EOI Escuela de
Organización
Industrial

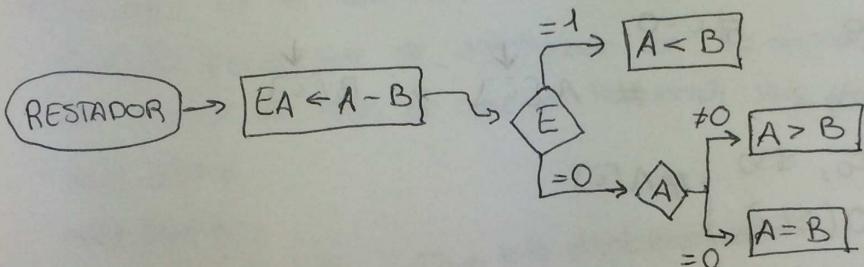
FEBRERO 2014

- 1- Imagine que se tiene dos registros A y B los cuales se desean comparar, es decir, saber si $A > B$, $A < B$ y si $A = B$. Describe como se pueden determinar estos tres casos si:

- Se pone un restador completo.

Restador completo

- $A > B$: Último préstamo = 0 → por lo tanto el resultado es: $A - B$
- $A = B$: Último préstamo = 0 → por lo tanto el resultado es 0.
- $A < B$: Último préstamo = 1 → por lo tanto el resultado es: $C_2 (B - A)$

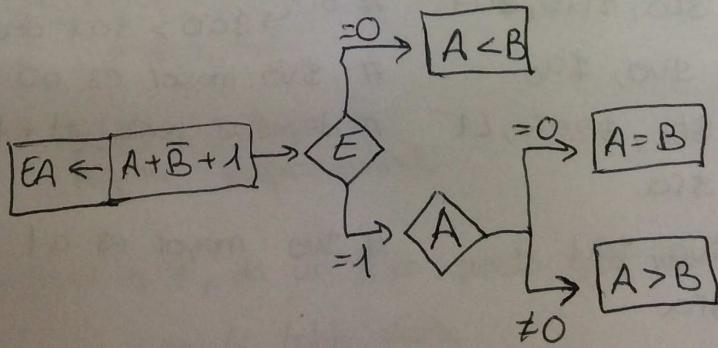


- Se posee un sumador completo con complemento (es decir, permite sumar una de los 2 registros en complemento a 2).

Sumador Completo con complemento

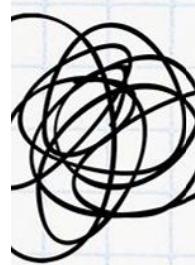
- $A > B$: Último préstamo = 1 → por lo tanto el resultado es: $A - B$.
- $A = B$: Último préstamo = 1 → por lo tanto el resultado es 0.
- $A < B$: Último préstamo = 0 → por lo tanto el resultado es: $C_2 (B - A)$

SUMADOR
 C_2



ali ali ooooh
estoy con l coin me
lo quito yo...

Necesito
concentración



↔

espacio
pierdo

?Cómo consiguió coins? → Planes pro: más coins
Plan Turbo: barato ←

Puedo eliminar la publ de este documento con 1 coin

Importante

3- Se tiene tres vectores A, B y C de 20 elementos cada uno, cuyas direcciones base se encuentran en los registros \$S0, \$S1 y \$S2 respectivamente. Desarrollar un programa mediante instrucciones MIPS que realice lo siguiente:

$$C[i] = \max(A[i], B[i]) \quad (\text{para } i=0, 1, \dots, 19)$$

for ($i=0$; $i < 20$, $i++$)

$\begin{matrix} \$S0 & \$S1 & \$S2 \\ A & B & C \end{matrix}$

```
{
    C[i] = max(A[i], B[i]);
}
```

move \$S3, \$zero # $i=0$

loop: sll \$t0, \$S3, 2 # $A[i]$ y $B[i]$

add \$t1, \$t0, \$S0 } # $A[i]$

lw \$a0, 0(\$t1)

add \$t1, \$t0, \$S1 } # $B[i]$

lw \$a1, 0(\$t1)

jal maximo

add \$t1, \$t0, \$S2 } # $C[i]$

sw \$v0, 0(\$t1)

addi \$S3, \$S3, 1 # $i++$

slti \$t1, \$S3, 20 # $i < 20$

bne \$t1, \$zero, loop # si \$t1 ≠ \$zero vuelve a loop

FIN

maximo: slt \$t0, \$a0, \$a1

si $\$a0 < \$a1$ devuelve 1
si $\$a0 > \$a1$ devuelve 0

move \$v0, \$a0 # \$v0 mayor es a0.

ble \$t0, \$zero, L1 # devuelve hasta jal + 1

jr \$ra

L1: move \$v0, \$a1 # \$v0 mayor es a1.

jr \$ra

4- Un sistema de memoria virtual paginado tiene un tamaño de página de 2K palabras. Un proceso puede direccionar, como máximo, 16 páginas. La memoria tiene capacidad para alojar 4 bloques. En un momento determinado la tabla de páginas de un proceso contiene las siguientes entradas:

Página	Bloque
1	3
5	1
6	2
14	0

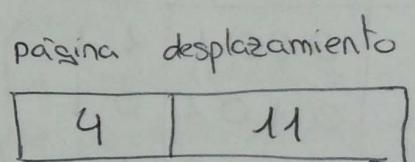
a) Formato de la dirección virtual y formato de la dirección física.

Dirección física que se corresponde con las siguientes direcciones virtuales del proceso, en caso de que la traducción sea posible:

a1) 35F7

a2) 3DF7

a) MV → $2K \rightarrow 2 \cdot 2^{10} = 2^{11} \rightarrow$ bits desplazamiento
 $\rightarrow 16 \text{ páginas} \rightarrow 2^4 \rightarrow$ bits página



a1) 35F7 → 0011 0101 11110111
 desplazamiento
 página

página → 0110 → 6, por lo tanto miro la tabla y para el valor 6 de página la página le corresponde el bloque 2.

1 5 F 7
1010111110111 → la dirección física sería: 15F7

bloque desplazamiento

a2) 3DF7 → 00111101 1110111

 Desplazamiento

 página

página → 0111 → 7, da un fallo, puesto que la página 7 no está en la tabla dada.

WUOLAH

b) Para la realización de las tablas de páginas se pueden usar tanto memorias de acceso aleatorio como asociativas. Representar el contenido de esta tabla de páginas usando los dos tipos de memoria. Calcula la capacidad de cada una de estas memorias.

Acceso aleatorio

BV	Bloque (2 bits)	
1	1 1	
1	0 1	
1	1 0	
1	0 0	

2^4 páginas \rightarrow Capacidad = páginas \cdot (bits bloque + Bits validación)

$$\boxed{\text{Capacidad} = 2^4 \cdot (2+1) = 16 \cdot 3 = 48 \text{ bits}}$$

Acceso asociativo

BV	Página (4 bits)	Bloque (2 bits)	
1	0 0 0 1	1 1	
1	0 1 0 1	0 1	
1	0 1 1 0	1 0	
1	1 1 1 0	0 0	

2^2 bloques \rightarrow Capacidad = bloques \cdot (bits bloque + bits página + bits validación);

$$\boxed{\text{Capacidad} = 2^2 \cdot (2+4+1) = 4 \cdot 7 = 28 \text{ bits}}$$

ali ali 0000
estoy con 1 coin me
lo quito yo...

Necesito
concentración



↔

↓

espacio
perdido

?Cómo consigo coins? → Planes pro: más coins
Plan Turbo: barato

Puedo eliminar la publ de este documento con 1 coin

Importante

SEPTIEMBRE 2013

- 1- Comparación entre microprogramación horizontal y vertical. ¿Qué ventajas tiene la una sobre la otra y qué inconvenientes?

Microprogramación horizontal: formato no codificado, microinstrucciones muy largas, alto grado de paralelismo, alto consumo de memoria de control, más rápidas. Muy flexible y fácil de diseñar/analizar.

Microprogramación vertical: formato codificado, microinstrucciones cortas, bajo paralelismo, menor consumo de la memoria de control, menos rápidas. Más lenta y costosa en circuitería.

- 2- Explicar cómo se soluciona el problema de sobrereflujo de división en el algoritmo correspondiente para números en representación de punto flotante. Señala en el diagrama donde se encuentra esta corrección.

No existe el problema del sobrereflujo de división. Inicialmente uno comprueba si los N bits más significativos del dividendo representan un número mayor que los N bits del divisor:

- Si no es mayor, se procede al algoritmo de división.
- Si es mayor, la solución consiste en realizar un desplazamiento a la derecha sobre el dividendo, con su correspondiente incremento del exponente. Como son números normalizados, después de este desplazamiento, los N bits más significativos del dividendo representan un número menor que los N bits del divisor, con lo que no hay sobrereflujo de división.

- 3- Supongamos que, en el siguiente esquema de la computadora mejorada, se modifica el registro OPR pasando de 4 a 5 bits. Indicar las siguientes cuestiones justificando brevemente su respuesta:

a) Máximo número de instrucciones en el repertorio de la máquina.

Sería según los bits que tenga el registro OPR, en este caso son 5 bits, por lo tanto el máximo número de instrucciones sería

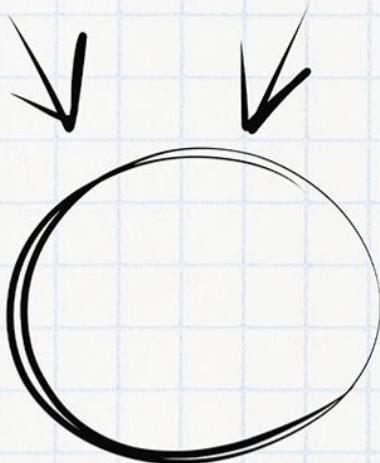
$$2^5 = 32 \text{ instrucciones.}$$

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
diamond Descargas sin publi al mes	10 🟡	40 🟡	80 🟡
clock Elimina el video entre descargas	✓	✓	✓
folder Descarga carpetas	✗	✓	✓
download Descarga archivos grandes	✗	✓	✓
circle Visualiza apuntes online sin publi	✗	✓	✓
glasses Elimina toda la publi web	✗	✗	✓
€ Precios	Anual <input type="checkbox"/>	0,99 € / mes	3,99 € / mes
			7,99 € / mes

Ahora que puedes conseguirlo,
¿Qué nota vas a sacar?



WUOLAH

b) Tamaño de la memoria principal expresado en número de palabras por ancho de palabra.

$$2^7 \times 12 \text{ bits.}$$

4- Se tienen 2 vectores A y B de 30 elementos cada uno, cuyas direcciones base se encuentran en los registros \$S0 y \$S1 respectivamente. Implementar un programa mediante instrucciones MIPS que realice lo siguiente:

$$A[0] = B[29], A[1] = B[28], A[2] = B[27], \dots, A[28] = B[1], A[29] = B[0]$$

for ($i=0; i < 30; i++$) Siendo $\$S0 = A, \$S1 = B, \$S2 = i$

$$\left. \begin{array}{l} \\ A[0] = B[29] \\ \vdots \\ A[29] = B[0] \end{array} \right\}$$

move \$S2, \$zero # $i=0$

addi \$t0, \$t0, 29 # $\$t0 = 29$

loop: sll \$t1, \$S2, 2 # $A[i]$

sll \$t2, \$t0, 2 # $B[t_0]$

add \$t3, \$t2, \$S1 # $B[29]$

lw \$t3, 0(\$t3)

add \$t1, \$t1, \$S0 # $A[0]$

lw \$t1, 0(\$t1)

add \$S2, \$S2, 1 # $i++$

addi \$t0, \$t0, -1 # t_0--

slti \$t4, \$S2, 30 # $i < 30$

bne \$t4, \$zero, loop # Vuelve al loop

FIN

WUOLAH

- 5- Una memoria caché asociativa por conjuntos de dos vías utiliza bloques de 4 palabras, pudiendo almacenar 1K palabras de memoria principal.
- Determinar la organización de la memoria caché
 - Determinar el tamaño de la memoria caché
 - Teniendo en cuenta que en memoria principal se encuentran los vectores de elementos de 16 bits, almacenados en las direcciones representadas por el siguiente esquema:

Vector A: Dirección de comienzo: 01C70

Contenido	8	13	21	0	77	3	98	52	1	47
-----------	---	----	----	---	----	---	----	----	---	----

Vector B: Dirección de comienzo: 03270

Contenido	122	40	28	22	17	41	12	68	15	80
-----------	-----	----	----	----	----	----	----	----	----	----

Representar gráficamente el contenido de la memoria caché después de ejecutar la línea de código:

$$l = A[2] - B[3]$$

(Representar únicamente las líneas de caché que han sido modificadas por la línea de código ejecutado)

- La memoria principal la calculo sabiendo que la dirección de cada vector tiene 5 elementos de 4 bits cada uno por lo tanto sería 20 bits en total. $2^{20} = 1 \text{ MB} \times 16$

$$\begin{array}{l} MC \xrightarrow{\text{2 vías}} \\ \xrightarrow{\text{4 palabras}} 2^{\text{②}} \xrightarrow{\text{bits palabra}} \end{array}$$

$$1K \approx \text{palabras} = 2^{10} / 2 \xrightarrow{\text{vías}} 2^9 / 2^2 = 2^7 \xrightarrow{\text{bits línea}}$$

$$MP \rightarrow 1M = 2^{\text{②}} \text{ bits total}$$

total = etiqueta + (línea + palabra); 20 = etiqueta + (7 + 2); 20 = etiqueta + 9;

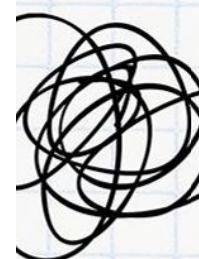
$$\text{etiqueta} = 20 - 9; \boxed{\text{etiqueta} = 11}$$

Etiqueta Línea Palabra

11	7	2
----	---	---

ali ali 0000
estoy con l coin me
lo quito yo...

Necesito
concentración



2=



espacio
pierdo

Planes pro: más coins
Plan Turbo: barato
?Cómo consigo coins?

Puedo eliminar la publi de este documento con 1 coin

Importante

b) $\boxed{\text{tamaño}} = (\text{bits validación} + (\text{nº palabras} \cdot \text{bits palabra}) + \text{etiqueta}) \cdot \text{vías} \cdot \text{bloque} =$
 $= (1 + (4 \cdot 16) + 11) \cdot 2 \cdot 2^7 = (1 + 64 + 11) \cdot 2 \cdot 128 = (76) \cdot 256 =$
 $= 19.456 \text{ bits} = \boxed{19 \text{ kb}}$

c) $A[0] = 01C70 \rightarrow \boxed{0000\ 0001\ 1100\ 0111\ 0000}$
 etiqueta linea

$\text{etiqueta} \rightarrow \underbrace{0000}_{0}\ \underbrace{0001}_{0}\ \underbrace{1100}_{E} \rightarrow 00E$

$\text{línea} \rightarrow \underbrace{0011}_{1}\ \underbrace{100}_{C} \rightarrow 1C$

$B[0] = 03270 \rightarrow \boxed{0000\ 0011\ 0010\ 0111\ 0000}$
 etiqueta linea

$\text{etiqueta} \rightarrow \underbrace{0000}_{0}\ \underbrace{001}_{1}\ \underbrace{1001}_{9} \rightarrow 019$

$\text{línea} \rightarrow \underbrace{0011}_{1}\ \underbrace{100}_{C} \rightarrow 1C$

Una vez tengo la posición 0 de cada vector, le pongo ahora la posición pedida.

Línea	BV	Etiqueta	Pd10	Pd11	Pd12	Pd13	Línea	BV	Etiqueta	Pd10	Pd11	Pd12	Pd13
1C	1	00E	8	13	21	0	1C	1	019	122	40	28	22

Por lo tanto, ya tengo lo que me piden:

$A[2] = 21$

$B[3] = 22$

6- Considérese la computadora mejorada del ejercicio 3. Implemente la instrucción ABSm, que tiene el contenido de la dirección m y lo devuelve a la misma dirección en valor absoluto (el dato se encuentra en complemento a 2).

- a) Mediante control microprogramado.
- b) Mediante control cableado.

Incluya el ciclo de búsqueda en ambos casos.

a) LCB

<u>S₂ S₁ S₀ F</u>	<u>I</u>	<u>B</u>	<u>R</u>	<u>LCB</u>	<u>Dir. Bifur.</u>
0 0 0 x	1	0	0		
0 0 1 x	0	1	0		
1 1 1 x	0	0	100		
1 0 1 0	10	0	100		
Dirección	Microoperación				
	Ciclo de Búsqueda				
ADDR (FETCH)	PC → MAR		0 0 0		
ADDR (FETCH)+1	M → GPR; PC+1 → PC		0 0 0		
ADDR (FETCH)+2	GPR (OP) → OPR		1 1 1		
	Ciclo de ejecución				
ADDR (ABS m)	GPR (AD) → MAR; 0 → ACC		0 0 0		
ADDR (ABS m)+1	M → GPR		0 0 0		
ADDR (ABS m)+2	GPR + ACC → ACC		0 0 0		
ADDR (ABS m)+3 +4	ROL F Acc si F=1 ROR F Acc		0 1 0 0 0 0	ADDR (ABS m)+7	
ADDR (ABS m)+5	ACC → ACC		0 0 0		
ADDR (ABS m)+6 +7	ACC + 1 → ACC BOR F ACC		0 0 1 0 0 0	ADDR (ABS m)+8	
ADDR (ABS m)+8	ACC → GPR		0 0 0		
ADDR (ABS m)+9	GPR → M		0 0 0 1		ADDR (FETCH)

El bloque LCB, I, B, R, E. Explicar todo.

Bloque LCB: lógica de control de bifurcación.

Incremento (I): Pasa a ejecutar la microinstrucción que se encuentra en la dirección CROM consecutiva.

Bifurcación (B): Pasa a ejecutar la microinstrucción que se encuentra en una dirección CROM específica.

Carga de rutina (R): Pasa a ejecutar la primera microinstrucción asociada al ciclo de ejecución de una instrucción determinada.

Enable (E): Bloque para habilitar o inhabilitar la ejecución de microoperaciones punto flotante. Te dan los algoritmos de división y multiplicación, y pregunta porque en uno se resta el sesgo y en el otro se le suma.

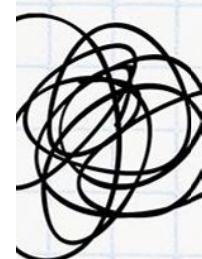
En el caso de la multiplicación, el exponente del resultado será la suma de los exponentes de los operandos. Para ello esto hay que tener una consideración. Y es que cada uno de los exponentes de los operandos está sesgado. Para tener el exponente del producto correctamente sesgado, debe restársele el sesgo a la suma de los dos exponentes.

En el caso de la división, el exponente del resultado se obtiene restándole al exponente del dividendo el exponente del divisor. Hay que tener en cuenta que al restar dos números sesgados, desaparece el sesgo. Luego para tener la representación de la división correctamente sesgada, hay que sumarle el sesgo a la resta de exponentes.

wuolah

ali ali ooooh
esto con l coin me
lo quito yo...

Necesito
concentración



≡

← →

pierdo
espacio

?Cómo consigo coins? → Planes pro: más coins
Plan Turbo: barato ←

Puedo eliminar la publ de este documento con 1 coin

Importante

WUDAH

for ($i = 0; i < 20; i++$){
 if ($A[i] > 0$){
 $B[i] = 1;$
 }

else

{
 $B[i] = 0;$ A B i
\$50 \$51 \$52move \$52, \$zero # $i = 0$ addi \$t5, \$zero, 1 # $t5 = 0 + 1$ loop: sll \$t0, \$52, 2 # $A[i]$ y $B[i]$ add \$t1, \$50, \$t0 } $A[i] \rightarrow t2$

lw \$t2, 0(\$t1)

slt \$t3, \$t2, 0 # $A[i] > 0$ add \$t1, \$51, \$t0 } # $B[i] = 1$

sw \$t5, 0(\$t1)

bne \$t3, \$zero, L1

addi \$52, \$52, 1

slt \$t3, \$52, 20 }

bne \$t3, \$zero, loop } # for

FIN

L1: sw \$zero, 0(\$t1) # $B[i] = 0$

addi \$52, \$52, 1

slt \$t3, \$52, 20 }

bne \$t3, \$zero, loop } # for

FIN