

Arquitectura e Ingeniería de Computadores
Ejercicios Resueltos. Curso 2007/2008

AIC

Juan Ignacio Alberola Colomo
jignacio.al@gmail.com
<http://sitio.de/jignacio>

PROBLEMAS TEMA 2BEJERCICIOS DE SUPERESCALARES

2) LIBRO

Para el fragmento de código siguiente y suponiendo que:

1 Cargo 2c

1 Alma 1c

3 S/R 1c

2 Mul 6c

Captar 4 instrucciones cada ciclo.

Emisor 4 instrucciones por ciclo.

Indicar el orden de la emisión de las instrucciones para los siguientes casos:

a) Ventana centralizada. Emisión desordenada ordenada y alineada

b) Ventana centralizada. Emisión desordenada y alineada.

c) Estación de reserva con 3 líneas para cada UF, envío ordenado y ventana deslizante.

Código:

```

lw r1,0x1ac
lw r2,0xc1f
add r3,r0,r0
mul r4,r2,r2
add r3,r3,r4
add r5,r0,0x1ac
add r6,r0,0xcff
sub r5,r5,#4
sub r6,r6,#4
sw(r5),r3
sw(r6),r4
    
```

a) Ventanizada. Ordenada y alineada.

#i	Inst	JF	ID/ISS	EX	ROB	WB
1	lw r1,0x1ac	1		3-4	5	6
2	lw r2,0xc1f	2		5-6	7	8
3	add r3,r0,r0	1		5	6	8
4	mul r4,r2,r2	1	2	7-12	13	14
5	add r3,r3,r4	2	7	13	14	$\boxed{15}^{*3}$
6	add r5,r0,0x..	2	7	13	14	$\boxed{15}$
7	add r6,r0,0x..	2	7	13	14	15
8	sub r5,r5,#4	2	7	14	15	16
9	sub r6,r6,#4	3		15	16	17
10	sw(r5),r3	3		15	16	17
11	sw(r6),r4	3		16	17	18

*1 Cuando se manda la multa ejecución se vacía la ventana y pueden entrar más.

*2 Emisión ordenada

*3 Suponiendo que hay varios buffers de escritura en los registros

b) Centralizada, Desordenada y Alineada

#i	Inst	IF	ID/ISS	EX	ROB	WB
1	lw r1,0x...	1	2	3-4	5	6
2	lw r2,0x...	1	2	5-6	7	8
3	add r3,r0,r0	1	2	3	4	8
4	mul r4,r2,r2	1	2	7-12	13	14
5	add r3,r3,r4	2	7	13	14	15
6	add r5,r0,0x...	2	7	8	9	15
7	add r6,r0,...	2	7	8	9	15
8	sub r5,r5,#4	2	7	9	10	15
9	sub r6,r6,#4	3	13	14	15	16
10	sw (rs),r3	3	13	14	15	16
11	sw (r6),r4	3	13	15	16	17

c) Estación de reserva con 3 líneas para cada UF, envío ordenado y ventana deslizante

#i	Inst	IF	ID/ISS	EX	ROB	WB	Estación
1	lw r1,0x...	1	2	3-4	5	6	RS LD
2	lw r2,0x...	1	2	5-6	7	8	RS LD
3	add r3,r0,r0	1	2	3	4	8	RS1 ADD
4	mul r4,r2,r2	1	2	7-12	13	14	RS MULT
5	add r3,r3,r4	2	3	13	14	15	RS1 ADD
6	add r5,r0,0x...	2	3	4	5	15	RS2 ADD
7	add r6,r0,...	2	3	4	5	15	RS3 ADD
8	sub r5,r5,#4	2	3	5	6	15	RS1 ADD
9	sub r6,r6,#4	3	4	5	6	15	RS2 ADD
10	sw (rs),r3	3	4	14	15	16	RS ST
11	sw (r6),r4	3	4	13	14	16	RS ST

*1 Cada instrucción a la estación correspondiente

d) Ventana centralizada, Emisión desordenada y no alineada

#i	Inst	LF	ID/ISS	EX	ROB	WB
1	lw r1,0...	1	2	3-4	5	6
2	lw r2,0x...	1	2	5-6	7	8
3	add r3,r0,r0	1	2	3	4	8
4	mul r4,r2,r2	2	2	7-12	13	14
5	add r3,r3,r4	2	3	13	14	15
6	add r5,r0,0x..	2	3	4	5	15
7	add r6,r0,0x..	2	3	4	5	15
8	sub rs,rs,#4	2	3	5	6	15
9	sub r6,r6,#4	3	4	5	6	15
10	su(rs),r3	3	4	14	15	16
11	su(r6),r4	3	4	13	14	16

*1 Debe esperar pq 6 está en ejecución

e) Ventana centralizada, Emisión Ordenada y no alineada

#i	Inst	LF	ID/ISS	EX	ROB	WB
1	lw r1,0x...	1	2	3-4	5	6
2	lw r2,0x...	1	2	5-6	7	8
3	add r3,r0,r0	1	2	5	6	8
4	mul r4,r2,r2	1	2	7-12	13	14
5	add r3,r3,r4	2	3	13	14	15
6	add r5,r0,0x..	2	3	13	14	15
7	add r6,r0,0x..	2	3	13	14	15
8	sub rs,rs,#4	2	3	14	15	16
9	sub r6,r6,#4	3	4	14	15	16
10	su(rs),r3	3	4	14	15	16
11	su(r6),r4	3	4	15	16	17

En el 13 no pq es emisión ordenada
y no pq en 14 su este ocupado

1) LIBRO

Ciclo en Renombrado

Supongamos que las siguientes cuatro instrucciones : (1) multd $f_3, f_1, f_2 ; f_3 = f_1 \cdot f_2$ (3)
 se introducen una tras otra (en los ciclos
 indicados entre paréntesis en un buffer
 de renombrado con acceso asociativo.

- (2) addd $f_2, f_3, f_1 ; f_2 = f_3 + f_1$ (4)
- (3) subd $f_3, f_3, f_1 ; f_3 = f_3 - f_1$ (5)
- (4) addd $f_5, f_1, f_2 ; f_5 = f_1 + f_2$ (5)

- Indicad :
- Evolución del buffer de renombrado
 - Momento y finalización de la ejecución de las instrucciones
 - Valores que quedan en los registros si inicialmente $f_1 = 2.0$ y $f_2 = 3.0$

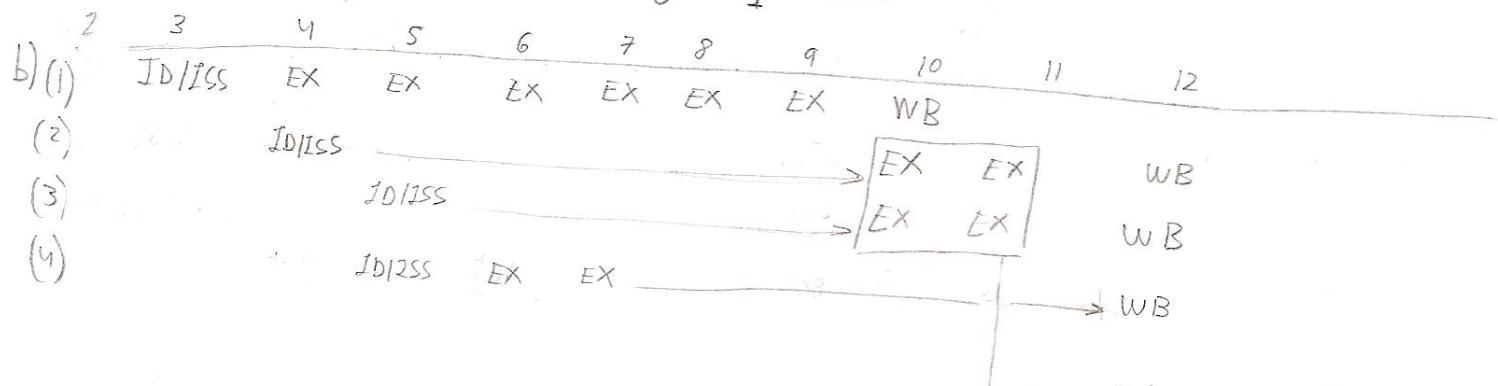
Nota: Multiplicación tarda 6 ciclos.

Suma y resta tardan 2 ciclos

Suficientes UF.

a)

#	EV	Dest	Valor	Válido	Últi	Ciclo
1	1	f_3		0	1	3
2	1	f_3	0	1		ciclo 4 → Se emite multd (4-9)
2	1	f_2	0	1		
1	1	f_3	0	0		ciclo 5 → Pasa add a pipe. (5-6)
2	1	f_2	0	1		
3	1	f_3	0	1		
4	1	f_5	0	1		



c) Desde 2 hasta 12 incluidos. 11 ciclos considerando el WB y IF → Podemos pq nos dicen que hay suficientes UF

Suponemos las siguientes 5 instrucciones que se meten en el buffer de renombrado en los ciclos indicados.

#					
(1)	multd	f_3, f_1, f_2	;	$f_3 = f_1 \cdot f_2$	(3)
(2)	addd	f_2, f_3, f_1	;	$f_2 = f_3 + f_1$	(4)
(3)	subd	f_3, f_2, f_1	;	$f_3 = f_2 - f_1$	(4)
(4)	multd	f_5, f_1, f_4	;	$f_5 = f_1 \cdot f_4$	(5)
(5)	addd	f_6, f_2, f_3	;	$f_6 = f_2 + f_3$	(6)

Mult tarda 5 ciclos
 Add/Sub 2 ciclos

Cada UF 1 ER2 3 líneas

Metemos en el buffer de renombrado.

- a) Indicar como evoluciona el buffer de renombrado.
- b) Momento de inicio y finalización de la ejecución. Etapa / tiempo.
- c) Indicar el número más pequeño de UF ADD/SUB para reducir el tiempo.

a) # EV Dest Valor Válido

1	1	f_3	-	0
---	---	-------	---	---

Última Asignación

1 → Cclo 3.

1	1	f_3	-	0
---	---	-------	---	---

① → Ejecutandose Ciclo 4 mult (4-8)

2	1	f_2	-	0
---	---	-------	---	---

1 → ERADD. Deja 1 linea libre

3	1	f_3	-	0
---	---	-------	---	---

1	1	f_3	-	0
---	---	-------	---	---

Cclo 5

2	1	f_2	-	0
---	---	-------	---	---

1

3	1	f_3	-	0
---	---	-------	---	---

1

4	1	f_5	-	0
---	---	-------	---	---

1

1	1	f_3	-	0
---	---	-------	---	---

Cclo 6

#4
 mult (6-10)

2	1	f_2	-	0
---	---	-------	---	---

1

3	1	f_3	-	0
---	---	-------	---	---

1

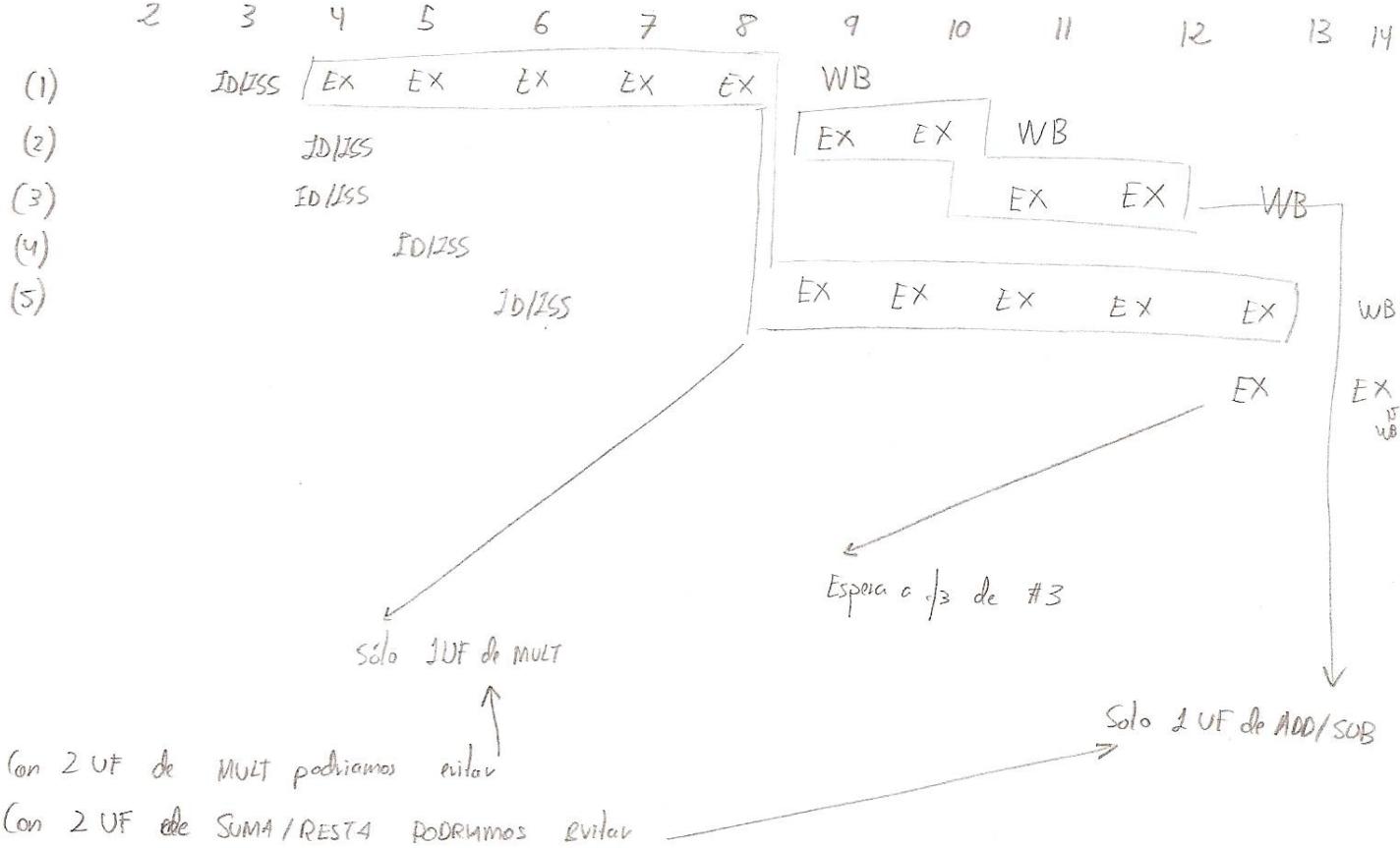
4	1	f_5	-	0
---	---	-------	---	---

1

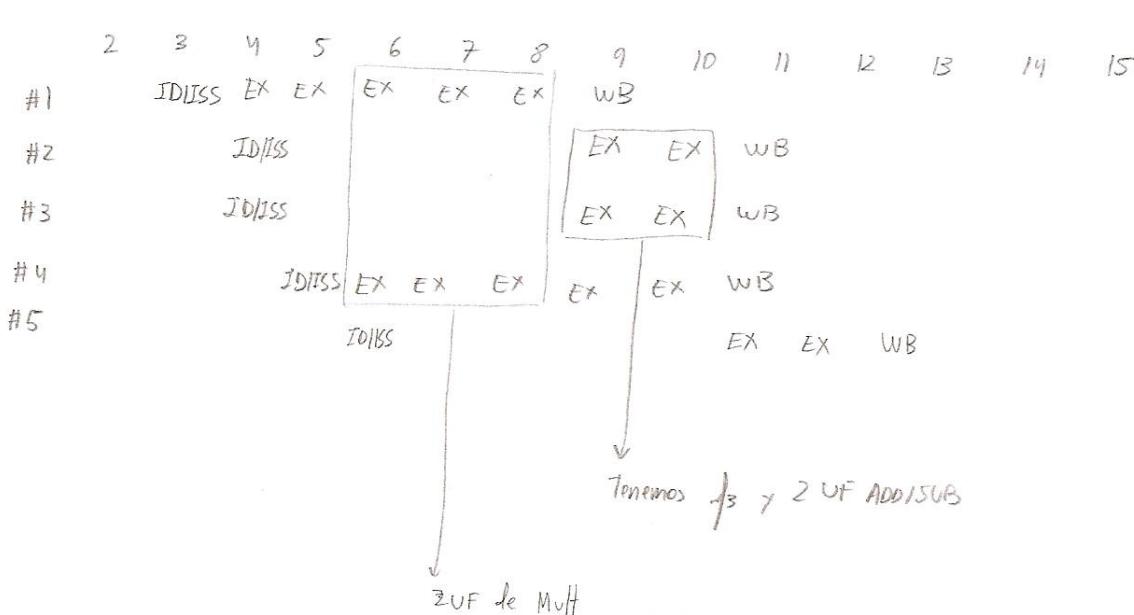
5	1	f_6	-	0
---	---	-------	---	---

1 → llena la ERADD

b)



c)



Antes acabábamos en 15 y ahora en el ciclo 13 (Retirando del cálculo, finalización de las instrucciones). El fin de ejecución (ninguna UF ocupada) surge en 12 frente a 14 del caso anterior.

3) LIBRO.

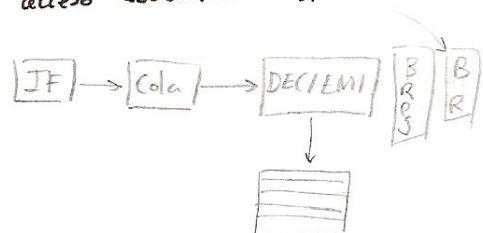
¿Cómo renombraría un compilador los registros en la secuencia de instrucciones siguiente para que no existan ni riesgos WAW ni WAR usando el mínimo número de registros posibles?

Algoritmo de Tomasulo.

#1 add r3, r2, r1	add r3b, r2a, r1a	Solo riesgos RAW
#2 sub r2, r3, r2	sub r2b, r3b, r2a	
#3 add r4, r2, r3	add r4a, r2b, r3b	Add consume 2 ciclos
#4 mult r3, r5, r3	mult r3c, r5a, r3b	Sub → Mult → 5 ciclos!
#5 sub r2, r4, r6	sub r2c, r4a, r6a	

¿Cómo se realizaría el renombrado en un buffer de renombrado con acceso asociativo si se aceptan, decodifican hasta tres instrucciones por ciclo?

Ciclo i⁽²⁾



#	EV	Dest	Valor	Valido	ÚltAsig
1	1	r3	-	0	1
2	1	r2	-	0	1
3	1	r4	-	0	1

Ciclo i+1⁽³⁾

#	EV	Dest	Valor	Valido	ÚltAsig
1	1	r3	0	0	
2	1	r2	0	0	
3	1	r4	0	1	
4	1	r3	0	1	
5	2	r2	0	1	

Ciclo i+2⁽⁴⁾ → Se mantiene. Segundo ciclo de ejecución de #1

Ciclo i+3⁽⁵⁾ → Ya se tiene r3. Puede pasar a ejecutarse r2.

#	EV	Dest	Valor	Valido	ÚltAsig
1	1	r3	v3	1	0
2	1	r2	0	0	
3	1	r4	0	1	
4	1	r3	0	1	
5	1	r2	0	1	

Primer ciclo de ejecución de #2

Ciclo i+4⁽⁶⁾. Se mantiene igual. Segundo ciclo de ejecución de #2

Ciclo i+5⁽⁷⁾

#	EV	Dest	Valor	Valido	UltAsig
1	1	v3	v3	1	0
2	1	v2	v2	1	0
3	1	v4	v4	0	1
4	1	v3	v3	0	2
5	1	v2	v2	0	2

Primer ciclo de ejecución para #3
 " " " " " " para #4

Ciclo i+6⁽⁸⁾

Segundo ciclo para #3 y #4

#3 acaba y a #4 le quedan 4 ciclos

Ciclo i+7⁽⁹⁾

Empleza #5 y es el 3º ciclo para mult

#	EV	Dest	Valor	Valido	UltAsig
1	1	v3	v3	1	0
2	1	v2	v2	1	0
3	1	v4	v4	1	1
4	1	v3	v3	0	1
5	1	v2	v2	0	1

Ciclo i+8⁽¹⁰⁾. Segundo ciclo para #5 y cuarto para mult

Ciclo i+9⁽¹¹⁾

#	EV	Dest	Valor	Valido	UltAsig
1	1	v3	v3	1	0
2	1	v2	v2	1	0
3	1	v4	v4	1	1
4	1	v3	v3	0	1
5	1	v2	v2	1	1

Acaba la multiplicación

Ciclo i+10⁽¹²⁾

#	EV	Dest	Valor	Valido	UltAsig
1	1	v3	v3	1	0
2	1	v2	v2	1	0
3	1	v4	v4	1	1
4	1	v3	v3 ₂	1	1
5	1	v2	v2 ₂	1	1

FIN

¿Cuánto tardarían en ejecutarse todas las instrucciones si add y sub tardan 2 ciclos y mult 5 ciclos y se pueden emitir hasta 3 instrucciones por ciclo?
(emisión ordenada y no alineada)

#i	Inst.	JF	ID ISS	EX	ROB	WB	
1	add r3,r2,r4	1	2	3-4	5	6	
2	sub r2,r3,r2	1	2	5-6	7	8	
3	add r4,r2,r3	1	2	7-8	9	10	Tardaría 13 ciclos considerando la etapa de WB.
4	mult r3,r5,r3	2	3	7-11	12	13	
5	sub r2,r4,r6	2	3	9-10	11	13	

4) LIBRO

Considerese que el fragmento de código siguiente:

lw r3,0x10a
 addi r2,r0,#128
 add r2,r0,0x0a
lw r4,0(r1)
lw r5,-8(r2)
 mult r6,r5,r3
 add r5,r6,r3
add r6,r4,r3
 sw 0(r4),r6
 sw -8(r1),r5
sub r2,r2,#16

se ejecuta en un procesador super-escalador que es capaz de
 Captar 4 inst/ciclo
 Decod 2 inst/ciclo
 Emittir 2 inst/ciclo (emisión NO ALINEADA)
 Escribir 2 ins/ciclo en los buffers correspondientes
 Relajar 2 ins/ciclo

Indicar el número de ciclos que tardaría en ejecutarse el
 programa suponiendo:
 a) Emisión ordenada y ejecución desordenada
 b) Emisión desordenada y ejecución desordenada

Nota: 1 UF LOAD Latencia 2 1 UF MULT Latencia 6
 1 UF STORE Latencia 1
 3 UF ADD/SUB Latencia 1
 No hay limitaciones de líneas.

a) Emisión ordenada y ejecución desordenada

#	Inst	IF	ID/ISS	EX	ROB	WB
1	lw r3,0x...	1	2	3-4	5	6
2	addi r2,r0,#128	1	2	3	4	6 } Finalizó solo 2i
3	add r1,r0,0x...	1	3	4	5	7
4	lw r4,0(1)	1	3	5-6	7	8
5	lw r5,-8(1)	2	4	7-8 (ocupado)	9	10
6	mult r6,r5,r3	2	4	9-14	15	16 } 1 Término 3UF, 100
7	add r5,r6,r3	2	5	15*1	16	17
8	add r6,r4,r3	2	5	15	16	17
9	sw o(1),r6	3	6	16	17	18
10	sw -8(1),r5	3	6	17	18	19
11	sub r2,r2,#16	3	7	17	18	19
→ Solo tenemos 1UF STORE						

b) Emisión desordenada y ejecución desordenada

! Lo escrito en este color es del ejercicio siguiente!

#	Inst	IF	ID/ISS	EX	ROB	WB
1	lw r3,0x...	1	2	3-4	5	6
2	addi r2,r0,#128	1	2	3	4	6
3	add r1,r0,0x...	1	3	4	5	7
4	lw r4,0(1)	1	3	5-6	7	8
5	lw r5,-8(1)	2	4	7-8	9	10
6	mult r6,r5,r3	2	4	9-14	15	16
7	add r5,r6,r3	2	5	15	16	17
8	add r6,r4,r3	2	5	7	8	17
9	sw o(1),r6	3	6	8	9	18
10	sw -8(1),r5	3	6	16	17	18
11	sub r2,r2,#16	3	7	8	10	19

6) LIBRO

En el caso descrito en el problema 4 (el mejor caso), supongamos que se usa un buffer de reorden (ROB) para permitir la ejecución desordenada y la finalización ordenada. Indicar cómo evolucionaría dicho buffer en el mejor de los casos.

ciclo	#	codop	nº inst	Reg Dest	UF	Res	ok	marca
2	1	lw	1	r3	LOAD		0	i
	2	addi	2	r2	ADD1		0	i
3	1	lw	1	r3	LOAD		0	X load
	2	addi	2	r2	ADD1		0	X load
	3	add	3	r1	ADD1		0	i
	4	lw	4	r4	LOAD		0	i
4	1	lw	1	r3	LOAD		0	i
	2	addi	2	r2	ADD1	r2+r0	0	X ready
	3	add	3	r1	ADD1		0	finacabida
	4	lw	4	r4	LOAD		0	X load
	5	lw	5	r5	LOAD		0	i
	6	mulld	6	r6	MUL		0	i
5	1	lw	1	r3		r3	1	f
	2	addi	2	r2		r2+r0	1	f
	3	add	3	r1		r0+0x...	1	f
	4	lw	4	r4			0	X
	5	lw	5	r5			0	i
	6	mulld	6	r6			0	i
	7	add	7	r5			0	i
	8	add	8	r6			0	i

6 → Igual que 5 pero sin las líneas 1y2. Además 4 está en su 2º ciclo y tenemos qy/o

7	4	lw		r4	1	f
5	lw			0		
6	mulld		11	0	X	
7	add			0	i	
8	add			0	X	
9	sw (6)		r6	0	i	
10	sw (6)		15	0	i	
11	sub		r2	0	i	

Ciclo	#_cod	RegDest	UF	Res	ok	marca
8	5	lw r5			0	X
	6	mult r6			0	i
	7	add r5			0	i
	8	add r6		r4+r3	1	f
	9	sw ← r6			0	X
	10	sw ← r5			0	i
	11	sub r2			0	X

9	5	lw r5		r5	2	f
	6	mult r6			0	X
	7	add r5			0	i
	8	add r6		r4+r3	1	f
	9	sw ← r6		← r6	2	f
	10	sw ← r5			0	i

PREGUNTAR QUE PASA

Se dispone de un procesador superscalar con la siguiente configuración

- La estación de reserva RS1 para las sumas y restas.
- La estación de reserva RS2 para las mults y divs.
- Un buffer de reorden ROB.
- 2 UF de sumas/restas con una latencia de 2 ciclos
- 1 UF de mult con latencia de 5 ciclos
- 2 UF de divisiones con latencia 40 ciclos.

El procesador es capaz de : Emitir (dos) instrucciones cada ciclo, las estaciones de reserva pueden realizar ENTRADAS NO ALINEADAS Y DESORDENADAS a las UF y pueden recibir hasta 2 inst/ciclo.

addd f_3, f_1, f_2

addd f_2, f_3, f_1

multd f_1, f_3, f_2

divd f_5, f_2, f_1

subd f_1, f_3, f_1

Los registros f_1 y f_2 tienen inicialmente los valores 10 y 5
¿qué valores y en qué orden se escribirán en los registros de la arquitectura?

Ciclo i

[RS1]	opc	dest	opl	ok1	opz	ok2	[RS2] VACIA
addd	①		10	1	5	1	
addd	2	$[f_3]$	0	5	0		

[ROB]	#	opc	Dest	UD	Valor	Ok	Marcia
	①	addd	f_2	add	0	0	i
	②	addd	f_2	add	0	0	i

Ciclo i+1

[RS1]	opc	dest	opl	ok1	opz	ok2
	addd	2	$[f_3]$	0	5	1

[RS2]	opc	dest	opl	ok1	opz	ok2
	multd	3	$[f_3]$	0	$[f_1]$	1
	divd	4	$[f_5]$	0	10	1

[ROB]	#	OPC	Dest	UD	Valor	ok	Marcia
	1	addd	f_3	ADD	0	x	(i+1)ciclo)
	2	addd	f_2	ADD	0	i	
	3	multd	f_4	MUL	0	i	
	4	divd	f_5	DIV	0	i	

ciclo i+2

[RS1]

opc	dest	op1	ok1	op2	ok2
addd	2	[f ₃]	0	5	1
subd	5	[f ₃]	0	10	1

[RS2]

opc	dest	op1	ok1	op2	ok2
-----	------	-----	-----	-----	-----

Igual que en (i+1)

[ROB]

#	Opc	Dest	UD	Valor	ok	marca
1	addd	f ₃	A00		0	X (2º ciclo)
2	addd	f ₂	A00		0	i
3	multd	f ₄	MUL		0	i
4	divd	f ₅	DIV		0	i
5	subd	f ₂	A00		0	i

ciclo i+3

[RS1]

VACIO

[RS2]

Igual que en (i+2)

[ROB]

#	Opc	Dest	UD	Valor	ok	marca
1	addd	f ₃	A00	15	1	f
2	addd	f ₂	A00	*	0	X (1º ciclo)
3	multd	f ₄	M..		0	i
4	divd	f ₅	D..		0	i
5	subd	f ₂	A00		0	X (1º ciclo)

ciclo i+4

[ROB]

#	Opc	Dest	UD	Valor	ok	marca
1	addd	f ₃	A..	15	1	→ Se saca y pasa a WB (Ya no está dentro de ROB)
2	addd	f ₂	A..		0	X (2º ciclo)
3	multd	f ₄	M..		0	i
4	divd	f ₅	D..		0	i
5	subd	f ₂	A..	5	1	X (2º ciclo)

[RS1]

VACIA

RS2

Ciclo i+5

[ROB]

#	opc	Dest	UD	Valor	Ok	Marcas
---	-----	------	----	-------	----	--------

2 addd f_2 A.. 20 1 f

3 multd f_4 M.. 0 ~~X~~ (1^{er} ciclo) acaba en i+9

4 divd f_5 D.. 0 ~~X~~ i ciclo

5 subd f_2 A.. 5 1 f \leftarrow No se puede retirar hasta que salgan los anteriores.

[RS1]

VACIA

[RS2]

opc	dest	opl	ok1	op2	ok2
-----	------	-----	-----	-----	-----

VACIA, se han pasado a ejecutar

Ciclo i+6 Se saca la linea 2 y la multd

Permanece así hasta i+10 que se escribe en ROB el resultado de la multd

Ciclo i+10

[ROB]

#	opc	Dest	UD	Valor	Ok	Marcas
---	-----	------	----	-------	----	--------

3 multd f_4 M.. 300 1 f

4 divd f_5 D.. 0 ~~X~~ (es en 6^{er} ciclo)

5 subd f_2 A.. 5 1 f \hookrightarrow escribirlo en ROB en i+45

Ciclo i+11 Se retira linea multd

Ciclo i+45

#	opc	Dest	UD	Valor	ok	Marcas
---	-----	------	----	-------	----	--------

4 divd f_5 D.. 2 2 f

5 subd f_2 A.. 5 1 f

Ciclo i+46 Puedo retirar los dos

ya están todos los registros actualizados.

¿Otra manera de hacerlo? Duda

#i	ID17SS	EX	ROB	WB	
1 addd f_3, f_1, f_2	i	$(i+1) - (i+2)$	$i+3$	$i+4$	\rightarrow Escribe en $f_3 \leftarrow 10+5$ 15
2 addd f_2, f_3, f_2	i	$(i+3) - (i+4)$	$i+5$	$i+6$	\rightarrow " " $f_2 \leftarrow 15+5$ 20
3 multd f_4, f_3, f_2	$i+1$	$(i+5) - (i+9)$	$i+10$	$i+11$	\rightarrow " " $f_4 \leftarrow 15 \cdot 20$ 300
4 divd f_5, f_2, f_1	$i+2$	$(i+5) - (i+4)$	$i+45$	$i+46$	\rightarrow " " $f_5 \leftarrow 20 : 10$ 2
5 subd f_2, f_2, f_1	$i+2$	$(i+3) - (i+4)$	$i+5$	$i+46$	\rightarrow " " $f_2 \leftarrow 15 - 10$ 5
6					

Juan Ignacio Alberola Colomo
 jignacio.al@gmail.com
<http://sitio.de/jignacio>

Hojas 9)

Suponga un procesador superscalar en que se captan 4 ins/ciclo, se decodifican 3 ins/ciclo, se emiten tres instrucciones por ciclo como máximo, se reciben 3 ins/ciclo.

La emisión y la ejecución son desordenadas y las instrucciones una vez decodificadas se introducen en un ROB que permite la finalización ordenada de instrucciones.

a) Indicar las dependencias entre instrucciones y cómo evoluciona el ROB hasta que se hayan ejecutado todas las instrucciones de la secuencia

add_d f₁, f₁, f₄

Suma y resta tardan 1 ciclo.

mult_d f₃, f₁, f₂

Mult tarda 4 ciclos.

add_d f₆, f₁, f₄

Sin límites en cuanto a líneas del buffer

sub_d f₄, f₁, f₆

ni en VF. Se supone que f₁, f₂ y f₄ tienen valores previos.

#	OP	IF	ID/kiss	EX	ROB	WB
1	add _d	f ₁ , f ₁ , f ₄	1	2	3	4
2	mult _d	f ₃ , f ₁ , f ₂	1	2	4 - 6	8
3	add _d	f ₆ , f ₁ , f ₄	1	2	4	5
4	sub _d	f ₄ , f ₁ , f ₆	1	3	5	6

ROB

ciclo 2	#	opc	UF	Res	ok	Marca	dest
	1	add _d	A	0	i	f ₁	
	2	mult _d	M	0	i	f ₃	
	3	add _d	A	0	i	f ₆	

ciclo 5	#	opc	UF	Res	ok	Marca	Dest
	2	mult _d	M	0	X	f ₃	
	3	add _d	A	f ₁ , f ₆	1	f	f ₆
	4	sub _d	A	0	X	f ₄	

ciclo 3	#	opc	UF	Res	ok	Marca	dest
	1	add _d	A	0	X	f ₁	
	2	mult _d	M	0	i	f ₃	
	3	add _d	A	0	i	f ₆	
	4	sub _d	A	0	i	f ₄	

ciclo 6	#	UF	Res	ok	Marca	Dest
	2	"		0	X	f ₂
	3	"	f ₁ , f ₆	1	f	f ₆
	4	"	f ₁ , f ₄	1	f	f ₄

ciclo 7... último ciclo de ejecución de la mult

ciclo 8... cambia linea 2 a f

ciclo 9... se vacía el buffer ROB.

ciclo 4	#	opc	UF	Res	ok	Marca	dest
	1	add _d	A	f ₁ , f ₄	2	f	f ₁
	2	mult _d	M	0	X	f ₃	
	3	add _d	A	0	X	f ₆	
	4	sub _d	A	0	i	f ₄	

9) LIBRO

2UF Suma / Resta \rightarrow 2 ciclos
 1 MULT \rightarrow 4 ciclos

Considerese que el fragmento de código siguiente :

- 1 subd f₂, f₂, f₁
- 2 addd f₄, f₂, f₃
- 3 subd f₅, f₂, f₃
- 4 multd f₆, f₂, f₃
- 5 subd f₁, f₂, f₅
- 6 subd f₇, f₄, f₆

Se ejecuta en un procesador superscalar capaz de captar, decodificar y omitir (ventana centralizada) 3 instrucciones cada ciclo y retirar hasta 2 inst./ciclo.

- a) Si disponemos de un ROB que permite la ejecución desordenada y la finalización ordenada.

- Tiempo emisión ordenada (no alineada)
- Tiempo emisión desordenada (no alineada)
- Tiempo emisión desordenada (no

- Emisión ordenada no alineada

H ⁱ	Instrucción	IF	ID/ISS	EX	ROB	WB
1	subd f ₂ , f ₂ , f ₁	1	2	3	4	5
2	addd f ₄ , f ₂ , f ₃	1	2	4	5	6
3	subd f ₅ , f ₂ , f ₃	1	2	(4)	5	6
4	multd f ₆ , f ₂ , f ₃	2	3	5-8	9	10
5	subd f ₁ , f ₂ , f ₅	2	3	5	6	10
6	subd f ₇ , f ₄ , f ₆	2	3	9	10	11

La emisión desordenada no se nota debido a las dependencias de dato, entre las instrucciones.

- Emisión desordenada no alineada

H ⁱ	Instrucción	IF	ID/ISS	EX	ROB	WB
1	subd f ₂ , f ₂ , f ₁	1	2	3	4	5
2	addd f ₄ , f ₂ , f ₃	1	2	4	5	6
3	subd f ₅ , f ₂ , f ₃	2	2	4	5	6
4	multd f ₆ , f ₂ , f ₃	2	3	5-8	9	10
5	subd f ₁ , f ₂ , f ₅	2	3	5	6	10
6	subd f ₇ , f ₄ , f ₆	2	3	9	10	11

Igualas

EJERCICIOS DE SUPERESCALARES

GESTIÓN DE SALTOS

14) LIBRO

7) HOJAS

En un programa, una instrucción de salto condicional (a una dirección de anterior) dada tiene el siguiente comportamiento en una ejecución de dicho programa.

S S N N N S S N S N S N S / S S S S N

S = Se produce salto

N = No se produce salto. Indicar la penalización efectiva que se introduce si se utiliza:

- Predictión fija (always not taken).
- Predictión estática (si negativo se toma, si positivo no).
- Predictión dinámica con 2 bits inicialmente en 11.
- Predictión dinámica de 3 bits, inicialmente en 111.

a) Predictión fija (always not taken) Se observa que el salto se produce 11 veces por lo tanto de los 18 saltos se falla 11.

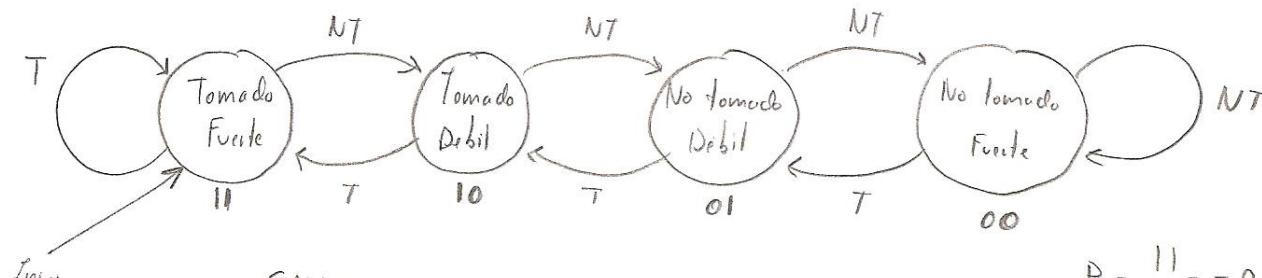
$$P = \frac{11}{18} = 0'61 \quad 61\% \text{ falla}$$

b) Predictión estática (si negativo se toma, si positivo no).

El salto siempre es hacia abajo (negativo). Se preverá siempre por saltarse.

No salta en 7 saltos , $\frac{7}{18} = 0'38 \quad 38\% \text{ falla}$

c) Predictión dinámica con 2 bits inicialmente en 11



SALTO S S N N N S S N S N S N S / S S S S N

PREDIC S S S S N N N S N S N S N S S S S

PEN 0011011111110001

$$P = \frac{11}{18} = 0'61$$

61% de fallos

d) Predicción dinámica de 3 bits. Inicialmente en 111.

Estado	Fallo	Predictión	Penalización
111	S	S	0
111	S	S	0
111	N	S	1
011	N	S	1
001	N	N	0
000	S	N	1
100	S	N	1
110	N	S	1
010	S	S	0
101	N	S	1
010	S	N	1
101	N	S	1
010	S	N	1
101	S	S	0
110	S	S	0
111	S	S	0
111	S	S	0
111	N	S	1

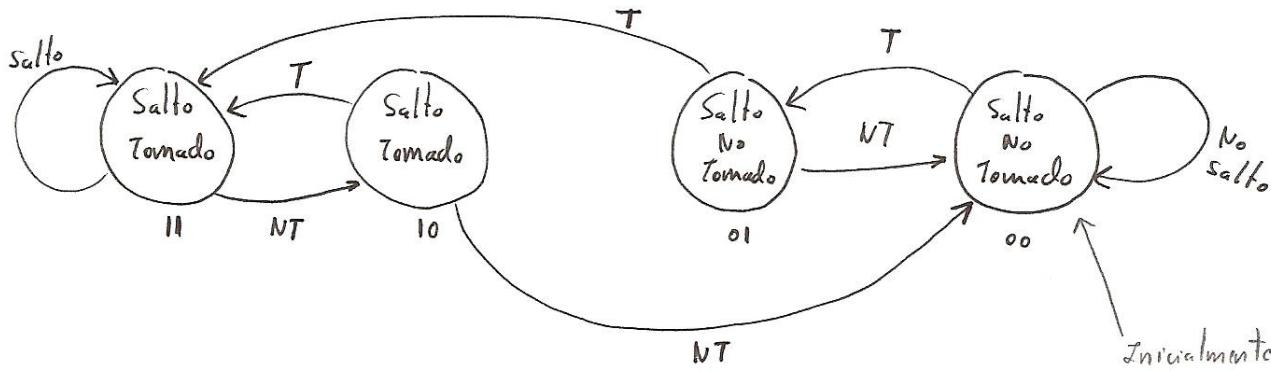
10) LIBRO

13) HOJAS

En el siguiente código : bucle :

$ld \quad f4, a ;$ valor constante en $f4f5$
 $ld \quad f0, 0(v);$ carga en $f0f1$ de 64 bits
 $add \quad f2, f4, f0; \quad f2f3 = f4f5 + f0f1$
 $sd \quad f2, 0(v);$ almacena los 64 bits de la suma
 $addi \quad r1, r1, #8;$ incrementa el puntero
 $subi \quad r2, r2, #1;$ decrementa el nro de datos
 $bnez \quad r2, bucle;$ saltar si $r2 \neq 0$

Si el procesador usa predicción dinámica de saltos con 2 bits de historia, según el diagrama de estados que se indica a continuación. ¿Cuántos fallos de predicción se producen al ejecutar una vez el código si los bits de historia se inicializan a 00?



Los saltos están en función del valor de $r2$, dependiendo de su valor cada ejecución será diferente.

$r2 = 0$ SALTO S S S S S S S S S ... S bucle infinito.
 PRED N N S S S
 PEN 1 1 0 0 0 ...

2 fallos de penalización

$r2 = 1$

SALTO N
 PRED N Sin fallos
 PEN 0

$r2 = 2$

SALTO S N
 PRED N N 1 fallo de predicción
 PEN 1 0

$r2 = 3$

SALTO S S N
 PRED N N S 3 fallos
 PEN 1 1 1

$r2 = 4$

SALTO S S S N
 PRED N N S S 3 fallos
 PEN 1 1 0 1

Para $r2 > 3$ siempre se producen los mismos fallos

EJERCICIOS VLIW. TEMA 3.

Ejercicios de ejemplo... antes de los exámenes...

- 1) En un procesador VLIW cuyas instrucciones pueden codificar dos operaciones (dos campos o slots en cada instrucción VLIW), todas las operaciones pueden predicarse. Para establecer los valores de los predicados se usan instrucciones de comparación (cmp) con el formato (p) p1,[p2] cmp.cnd x,y donde cnd es la condición que se comprueba entre x e y (lt, ge, eq, ne,...). Si la condición es verdadera p1 = 1 y p2 = 0 y si es falsa p1 = 0 y p2 = 1. La operación solo se ejecuta si el prediccado p = 1. Indique como se escribiría la sentencia.

```

for i=1 to 2 do
    if ((X[i]/4)==2) then X[i]=8*X[i];
    else
        if (X[0]<2) then X[i]=2;
    finIf
finFor

```

en lenguaje ensamblador sin ninguna operación de salto y con el número mínimo de instrucciones VLIW teniendo en cuenta que las operaciones de comparación solo pueden aparecer en el primer campo o slot de la instrucción VLIW.

p1 cmp.eq r0,r0 ; p1=1
 p2 cmp.ne r0,r0 ; p2=0
 p3 cmp.ne r0,r0 ; p3=0
 p4 cmp.ne r0,r0 ; p4=0

Iteración 1

- (p1) lw r1,X(r0); r1 = X[0]
 (p1) lw r2,#2 ; r2 = 2
 (p1) lw r3,X+4(r0); r3 = X[1]
 (p1) sllr r4,r3,#2 ; r4 = X[1]/4 → desplazamos r3 a la derecha para dividir
 (p1) p2,p3 cmp.eq r4,r2;
 (p2) slli r4,r3,#3 ; r4 = X[1]*8
 (p2) sw X+4(r0),r4
 (p3) p4 cmp.lt r1,r2 ;
 (p4) sw X+4(r0),r2 ; X[1]=2

Iteración 2

p5 cmp.eq r0,r0
 p6 cmp.ne r0,r0
 p7 cmp.ne r0,r0
 p8 cmp.ne r0,r0

(p5) lw r5, X+8(r0); r5 = X[2]
 (p5) sllr r6, r5, #2; r6 = X[2]/4
 (p5) p6, p7 cmp.eq r6, r2
 (p6) slli r6, r5, #3
 (p6) sw X+8(r0), r6
 (p7) p8 cmp.lt r1, r2
 (p8) sw X+8(r0), r2

Ahora hay que meterlo en las instrucciones VLIW.

SLOT 1

p1 cmp.eq r0,r0
 p2 cmp.ne r0,r0
 p3 cmp.ne r0,r0
 p4 cmp.ne r0,r0
 p5 cmp.eq r0,r0
 (p1) p2, p3 cmp.eq r4, r2
 p6 cmp.ne r0,r0
 p7 cmp.ne r0,r0
 (p3) p4 cmp.lt r1, r2
 p8 cmp.ne r0,r0

(p5) p6, p7 cmp.eq r6, r2
 (p7) p8 cmp.lt r1, r2
 (p8) sw X+8(r0), r2

SLOT 2

(p1) lw r1, X(r0)
 (p1) lw r2, #2
 (p1) lw r3, X+4(r0)
 (p1) sllr r4, r3, #2
 (p5) lw r5, X+8(r0)
 (p2) slli r4, r3, #3
 (p2) sw X+4(r0), r4
 (p5) sllr r6, r5, #2
 (p4) sw X+8(r0), r2

(p6) slli r6, r5, #3
 (p6) sw X+8(r0), r6

Hacer lo mismo para el siguiente código

```
if ((X[0] mod 2) == 0) then
    for i=1 to 2 do
        if ((X[i] mod 2) == 0) then X[i] = 2 * X[i]
    else
        if (X[0] < 0) then X[i] = 0
```

1^ª iteración

p1 cmp.eq r0,r0
 p2 cmp.ne r0,r0
 p3 cmp.ne r0,r0
 p4 cmp.ne r0,r0
 p5 cmp.ne r0,r0
 (p1) lw r1, X(r0)
 (p1) andi r2, r1, #1; en r2 todo 0 si r1 es par
 (p1) p2 cmp.eq r2, r0
 (p2) lw r3, X+4(r0)
 (p2) andi r4, r3, #1
 (p2) p3, p4 cmp.eq r4, r0
 (p3) slli r5, r3, #1
 (p3) sw X+4(r0), r5
 (p4) p5 cmp.lt r1, r0
 (p5) sw X+4(r0), r0

2^ª iteración

p6 cmp.ne r0,r0
 p7 cmp.ne r0,r0
 p8 cmp.ne r0,r0
 (p2) lw r6, X+8(r0)
 (p2) andi r7, r6, #1
 (p2) p6, p7 cmp r7, r0
 (p6) slli r8, r6, #1
 (p6) sw X+8(r0), r8
 (p7) p8 cmp.lt r1, r0
 (p8) sw X+8(r0), r0

SLOT 1

$p_1 \text{ cmp.eq } r0, r0$
 $p_2 \text{ cmp.ne } r0, r0$
 $p_3 \text{ cmp.no } r0, r0$
 $p_4 \text{ cmp.ne } r0, r0$
 $(p_1) p_2 \text{ cmp.eq } r2, r0$
 ~~$p_5 \text{ cmp.no } r0, r0$~~
 $p_6 \text{ cmp.no } r0, r0$
 $(p_2) p_3, p_4 \text{ cmp.eq } r4, r0$
 $p_7 \text{ cmp.no } r0, r0$
 $p_8 \text{ cmp.ne } r0, r0$
 $(p_2) p_6, p_7 \text{ cmp } r7, r6, \#1$
 $(p_7) p_8 \text{ cmp.lt } r1, r0$
 $(p_6) sw X+\delta(r0), r8$

SLOT 2

$-$
 $(p_1) lw r1, X(r0)$
 $(p_1) andi r2, r1, \#1$
 $-$
 $-$
 $(p_2) lw r3, X+r4(r0)$
 $(p_2) andi r4, r3, \#1$
 $(p_2) lw r6, X+\delta(r0)$
 $(p_2) andi r7, r6, \#1$
 $(p_3) slli rs, r3, \#1$
 $(p_3) sw X+r4(r0), rs$
 $(p_6) slli r8, r6, \#1$
 $- \sw \circlearrowleft \circ \text{ riesgo estructural}$

Preguntar

Ejercicios VLIW de exámenes.

a) Describe un ejemplo de aplicación práctica de la arquitectura VLIW aplicado a cualquier rama de la ciencia que requiera de las características de este tipo de arquitectura

b) En un procesador VLIW cuyas instrucciones pueden codificar dos operaciones, todas las operaciones deben prediseñarse. Para establecer los valores de los prediseños se usan operaciones de comparación (cmp) con el formato (p) $p1[p_1, p_2]$ cmp.cnd x,y donde cnd es la condición que se comprueba entre x e y. Si la condición es verdadera $p1=1$ y $p2=0$ y si es falsa $p1=0$ y $p2=1$. La operación solo se ejecuta si $p=1$.

Pasar el siguiente código a ensamblador. Las cmp solo en 1er slot. Sin ninguna ins de sc/flo.

for i=1 to 2 do

```

    X[i+1] = Y[i]/2;
    if (X[i]<0) then Y[i] = X[i+1]/2;
    else
        if (X[i]==2) then X[i+1] = Y[i+2];
        else
            if (Y[i]==1) then X[i] = Y[0];
            else X[i] = X[i+1] / Y[i]
}

```

fin for

1^a iteración

$p1 \text{ cmp.eq } r0,r0$	(p1) lw r1, Y+4(r0)
$p2 \text{ cmp.ne } r0,r0$	(p1) sllr r2, r2, #1 → (p1) sw X+8(r0), r2
$p3 \text{ cmp.ne } r0,r0$	(p1) lw r3, X+4(r0)
$p4 \text{ cmp.ne } r0,r0$	(p1) p2,p3 cmp.lt r3,r0 ; r3<0?
$p5 \text{ cmp.ne } r0,r0$	(p2) sllr r4, r2, #1
$p6 \text{ cmp.no } r0,r0$	(p2) sw Y+4(r0), r4
$p7 \text{ cmp.ne } r0,r0$	(p3) p4,p5 cmp r3, #2
	(p4) lw r5, Y+12(r0)
	(p4) sw X+8(r0), r5
	(p5) p6,p7 cmp.eq r2, #1
	(p6) lw r6, Y(r0)
	(p6) sw X+4(r0), r6
	(p7) div r7, r2, r2
	(p7) sw X+8(r0), r7

2^o iteración

p8 cmp.ne r0,r0	(p1) lw r8, Y+8(r0)
p9 cmp.ne r0,r0	(p1) sllr r9, r8, #1
p10 cmp.ne r0,r0	(p1) sw X+12(r0), r9
p11 cmp.ne r0,r0	(p1) p8, p9 cmp. lt r2, r0
p12 cmp.ne r0,r0	(p8) sllr r10, r9, #1
p13 cmp.ne r0,r0	(p8) sw Y+8(r0), r10
	(p9) p10, p11 cmp. eq r2, #2
	(p10) lw r11, Y+16(r0)
	(p10) sw X+12(r0), r11
	(p11) p12, p13 cmp.eq r8, #1
	(p12) lw r12, Y(r0)
	(p12) sw X+8(r0), r12
	(p13) div r13, r9, r8
	(p13) sw X+8(r0), r13

SLOT 1

p1 cmp.eq r0,r0	
p2 cmp.no r0,r0	
p3 cmp.ne r0,r0	
p4 cmp.ne r0,r0	
p5 cmp.ne r0,r0	
(p1) p2, p3 cmp.lt r3, r0	
p6 cmp.ne r0,r0	(p1) lw r1, Y+4(r0)
p7 cmp.ne r0,r0	(p1) sllr r2, r1, #1
(p3) p4, p5 cmp.r3, #2	(p1) sw X+8(r0), r2
p8 cmp.no r0,r0	(p1) lw r3, X+4(r0)
p9 cmp.ne r0,r0	(p1) lw r8, Y+8(r0)
(p5) p6, p7 cmp.eq r1, #1	(p2) sllr r4, r2, #1
p10 cmp.ne r0,r0	(p2) sw Y+4(r0), r4
p11 cmp cmp.no r0,r0	(p1) sllr r9, r8, #1
(p1) p8, p9 cmp.lt r2, r0	(p4) lw r5, Y+12(r0)
(p9) p10, p11 cmp.eq r2, r0	(p4) sw X+8(r0), r5
(p11) p12, p13 cmp.eq r8, #1	(p1) sw X+12(r0), r9
(p8) sw Y+8(r0), r10	(p6) lw r6, Y(r0)
(p10) sw X+12(r0), r11	(p6) sw X+4(r0), r6
(p12) sw X+8(r0), r12	(p7) div r7, r2, r1

SLOT 2

(p1) sllr r2, r1, #1	(p1) sw X+8(r0), r13
(p1) sw X+8(r0), r14	
(p1) sllr r9, r8, #1	
(p4) lw r5, Y+12(r0)	
(p4) sw X+8(r0), r5	
(p1) sw X+12(r0), r9	
(p6) lw r6, Y(r0)	
(p6) sw X+4(r0), r6	
(p7) div r7, r2, r1	
(p7) sw X+8(r0), r7	
(p8) sllr r10, r9, #1	
(p10) lw r11, Y+16(r0)	
(p12) lw r12, Y(r0)	
(p13) div r13, r9, r8	

• Lo mismo que los anteriores ejecutados pero con alguna salvedad.

NOTA . X se supone inicializado en dirX y a en dirA

Todas las instrucciones tienen de latencia 1 ciclo menos la mult que tiene 3.

1^a iteración

```

p1 cmp.eq r0,r0
p2 cmp.ne r0,r0
p3 cmp.ne r0,r0
(p1) lw r1, dirX(r0)
(p1) sllr r2, r1, #1
(p1) lw r3, dirA(r0)
(p1) mult r4, r3, r2
(p1) p2,p3 cmp.eq r4, #100
(p2) lw r5, #12
(p2) sw dirA(r0), r5
(p3) lw r6, #6
(p3) sw dirA(r0), r6
    
```

2^a Iteración

```

p4 cmp.eq r0,r0
p5 cmp.ne r0,r0
p6 cmp.ne r0,r0
(p4) lw r6, dirX+r4(r0)
(p4) sllr r7, r6, #1
(p4) lw r8, dirA(r0)
(p4) mult r9, r8, r7
(p4) p5,p6 cmp.eq r9, #100
(p5) sw dirA(r0), r5
(p6) sw dirA(r0), r6
    
```

3^a Iteración

```

p7 cmp.eq r0,r0
p8 cmp.ne r0,r0
p9 cmp.ne r0,r0
(p7) lw r10, dirX+r8(r0)
(p7) sllr r11, r10, #1
(p7) lw r12, dirA(r0)
(p7) mult r13, r11, r12
(p7) p8,p9 cmp.eq r13, #100
(p8) sw dirA(r0), r5
(p9) sw dirA(r0), r6
    
```

for (i=0 ; i<2; i++)

}

if ((X[i]/2) · a == 100) a = 12
 else a = 6

}

SLOT1

```

p1 cmp.eq r0,r0
p2 cmp.ne r0,r0
p3 cmp.ne r0,r0
p4 cmp.eq r0,r0
p5 cmp.ne r0,r0
p6 cmp.ne r0,r0
p7 cmp.eq r0,r0
p8 cmp.ne r0,r0
(p1) p2,p3 cmp.eq r4, #100
(p2) p9 cmp.ne r0,r0
(p7) sllr r11, r10, #1
(p8) sw dirA(r0), r5
(p4) p5,p6 cmp.eq r9, #100
(p5) sw dirA(r0), r5
(p3) sw dirA(r0), r6
(p6) sw dirA(r0), r6
(p7) p8,p9 cmp.eq r13, #100
(p8) sw dirA(r0), r5
    
```

SLOT2

```

(p1) lw r1, dirX(r0)
(p1) sllr r2, r1, #1
(p1) lw r3, dirA(r0)
(p1) mult r4, r3, r2
(p4) lw r6, dirX+r4(r0)
(p4) sllr r7, r6, #1
(p4) lw r8, dirA(r0)
(p4) mult r9, r8, r7
(p7) lw r10, dirX+r8(r0)
(p2) lw r5, #12
(p7) lw r12, dirA(r0)
(p7) mult r13, r11, r12
(p3) lw r16, #6
(p9) sw dirA(r0), r6
    
```

En un procesador VLIW cuyas instrucciones pueden codificar dos operaciones todas las operaciones pueden prediseñarse. Para establecer el valor de los predicados, ya sabemos, (p) p1, [p2] cmp.eq r0,r0 x,y

Resumir instrucciones VLIW lo siguiente. Ten en cuenta que las operaciones cmp solo pueden estar en el primer slot.

for i=1 to 2 do

```

    X[i] = Y[i+1];
    if (X[i] < X[i+1]) then Y[i] = X[i-1];
    else
    {
        if (X[i] == Y[i]/2) then X[i+1] = Y[i+2]
        else X[i+1] = Y[0] * 16;
    }
    X[i] = X[i+1] * Y[i-1];

```

finFor

1^a iteración

```

p1 cmp.eq r0,r0
p2 cmp.ne r0,r0
p3 cmp.ne r0,r0
p4 cmp.ne r0,r0
p5 cmp.ne r0,r0
(p1) lw r1, Y+8(r0); r1 = Y[2]
(p1) sw X+8(r0), r1; X[1] = Y[2]
(p1) lw r2, X+8(r0); r2 = X[2]
(p1) p2, p3 cmp.lt r1, r2
(p2) lw r3, X(r0); r3 = X[0]
(p2) sw Y+8(r0) = r3; Y[1] = X[0]
(p3) lw r4, Y+8(r0)
(p3) slli r5, r4, #1
(p3) p4, p5 cmp.eq r1, r5
(p4) lw r6, Y+12(r0)
(p4) sw X+8(r0), r6
(p5) lw r7, Y(r0)
(p5) slli r8, r7, #1
(p5) sw X+8(r0), r8
(p1) lw r9, Y(r0) → (p1) lw r20, X+8(r0)
(p1) mult r10, r9, r20
(p1) sw X+8(r0), r10

```

2^a iteración

p6 cmp.eq r0,r0	(p6) lw r12, Y+12(r0)
p7 cmp.ne r0,r0	(p6) sw X+8(r0), r11
p8 cmp.ne r0,r0	(p6) lw r12, X+12(r0)
p9 cmp.ne r0,r0	(p6) p7, p8 cmp.lt r11, r12
p10 cmp.ne r0,r0	podría → (p7) sw Y+8(r0), r10
	ahorrarme (p8) lw r13, Y+8(r0)
	j. lw (p7) slli r14, r13, #1
	(p8) p9, p10 cmp.eq r11, r14
	(p9) lw r15, Y+16(r0)
	(p9) sw X+12(r0) ≠ r15
	(p10) lw r16, Y(r0)
	(p10) slli r17, r16, #1
	(p10) sw X+12(r0), r17
	(p6) lw r18, Y+4(r0)
	(p6) mult r19, r21, r18
	(p6) sw X+8(r0), r19
	(p6) lw r21, X+12(r0)

Slot1

$p_1 \text{ cmp.eq } r0, r0$
 $p_2 \text{ cmp.ne } r0, r0$
 $p_3 \text{ cmp.ne } r0, r0$
 $p_4 \text{ cmp.ne } r0, r0$
 $(p_1) p_2, p_3 \text{ cmp.lt } r2, r2$
 $p_5 \text{ cmp.ne } r0, r0$
 $p_6 \text{ cmp.eq } r0, r0$
 $p_7 \text{ cmp.ne } r0, r0$
 $p_8 \text{ cmp.ne } r0, r0$
 $(p_3) p_4, p_5 \text{ cmp.eq } r2, r2$
 $p_9 \text{ cmp.ne } r0, r0$
 $p_{10} \text{ cmp.ne } r0, r0$
 $(p_5) \text{ lw } r7, Y(r0)$
 $(p_5) \text{ slli } r8, r7, \#4$
 $(p_5) \text{ sw } X+8(r0), r8$
 $(p_1) \text{ mult } r10, r9, r20$
 $(p_1) \text{ sw } X+4(r0), r10$
 $(p_6) p_7, p_8 \text{ cmp.lt } r11, r12$
 $(p_7) \text{ sw } Y+8(r0), r10$
 $(p_8) p_9, p_{10} \text{ cmp.eq } r11, r14$
 $(p_9) \text{ lw } r15, Y+16(r0)$
 $(p_9) \text{ sw } X+12(r0), r15$
 $(p_{10}) \text{ sw } X+12(r0), r17$
 $(p_6) \text{ mult } r19, r21, r18$
 $(p_6) \text{ sw } X+8(r0), r19$

Slot2

$-$
 $(p_1) \text{ lw } r2, Y+8(r0)$
 $(p_1) \text{ sw } X+4(r0), r1$
 $(p_1) \text{ lw } r2, X+8(r0)$
 $(p_2) \text{ lw } r3, X(r0)$
 $(p_2) \text{ sw } Y+4(r0), r3$
 $(p_3) \text{ lw } r4, Y+4(r0)$
 $(p_3) \text{ sllr } r5, r4, \#1$
 $(p_6) \text{ lw } r11, Y+12(r0)$
 $(p_4) \text{ lw } r6, Y+12(r0)$
 $(p_4) \text{ sw } X+8(r0), r6$
 $(p_1) \text{ lw } r9, Y(r0)$
 $-$
 $(p_1) \text{ lw } r20, X+8(r0) \longrightarrow ? \text{ Duda}$
 $(p_6) \text{ sw } X+8(r0), r11$
 $(p_6) \text{ lw } r12, X+12(r0)$
 $-$
 $(p_8) \text{ sllr } r14, r13, \#1$
 $(p_6) \text{ lw } r18, Y+4(r0)$
 $(p_{10}) \text{ lw } r16, Y(r0)$
 $(p_{10}) \text{ slli } r17, r16, \#4$
 $(p_6) \text{ lw } r21, X+12(r0) \longrightarrow ? \text{ Duda}$
 $-$
 $-$

Uffff...

- En los procesadores VLIW una técnica para el aprovechamiento del rendimiento utilizada comúnmente por los compiladores es la segmentación software. Describa en qué consiste esta técnica y expliquela con el siguiente código de programa.

```

loop : ld f0, 0(r1)
       add f4, f0, f2
       sd f4, 0(r1)
       addd r1, r1, #8
       addd r3, r3, #1
       breq r3, #100, loop
    
```

dependientes

Consiste en una reorganización de los bucles de forma que cada iteración del código transformado contiene instrucciones tomadas de distintas iteraciones del bucle original.

De esta manera se separan las instrucciones dependientes en el bucle original entre diferentes iteraciones del bucle nuevo

ld f0, 0(r1)	add f4, f0, f2	ld f0, 8(r1)
sd f4, 0(r1)	add f4, f0, f2	ld f0, 8(r1)
sd f4, 8(r1)	add f4, f0, f2	ld f0, 16(r1)
sd f4, 16(r1)	add f4, f0, f2	
	sd f4, 8(r1)	add f4, f0, f2
	sd f4, 16(r1)	

```

loop : sd f4, 0(r1)
       add f4, f0, f2
       ld f4, 16(r1)
       add r1, r1, #8
       add r3, r3, #1
       breq r3, #100, loop
    
```

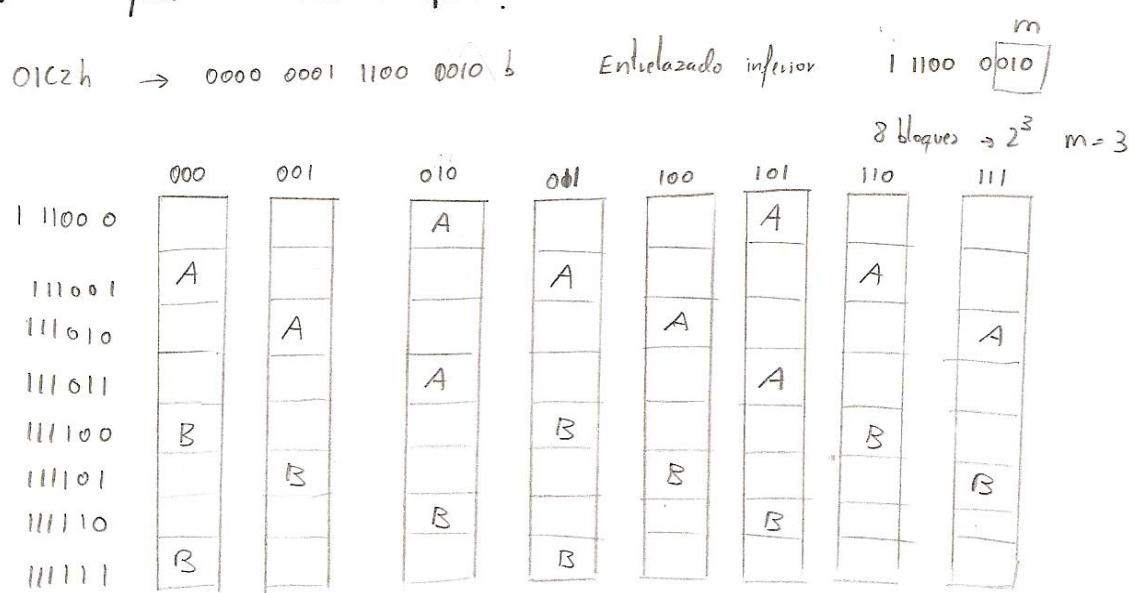
PROBLEMAS TEMA 4. ARQUITECTURAS VECTORIALES

Ejercicio de examen.

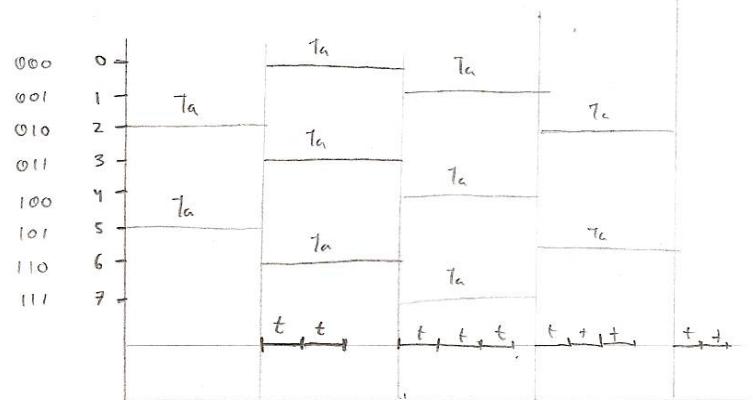
Se dispone de una arquitectura vectorial especializada en suma de vectores, dicha arquitectura se caracteriza por usar un acceso a memoria de tipo S.

La memoria se estructura en 8 bloques y sigue un enlazado de orden inferior. Se desea sumar 2 vectores de 10 elementos que se encuentran almacenados con un stride de 3 a partir de la dirección 01C2h

- ¿Cuántos accesos deberá hacer el procesador para recuperar el contenido de los dos vectores?
- ¿Y si fuese un acceso de tipo C?



Accesos con TIPO S

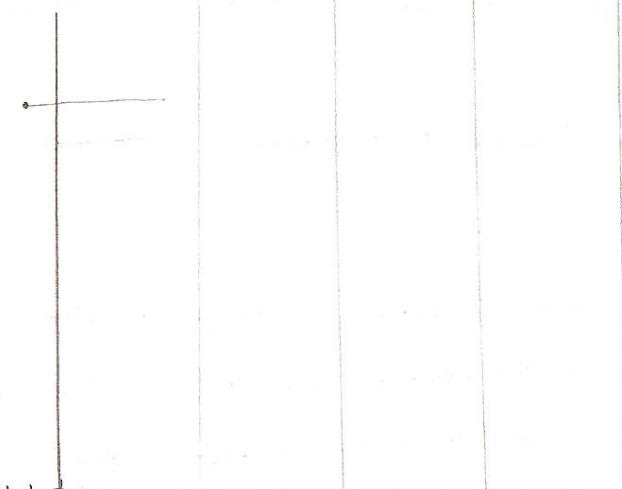
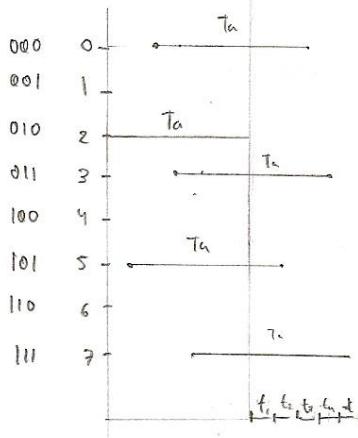


Se puede deducir que como el acceso es simultáneo el tiempo total será de $8T_a + 2t$
 ↳ 8 accesos a memoria.

$$T = 4T_a + 2t \text{ para el primer vector}$$

Con acceso de tipo C.

Suponemos que hasta aquí tenemos 16 elementos con $2Ta + 8t$



Si tuviésemos 8 elementos en módulos diferentes tendríamos $1Ta + 8t$.
con 16 elementos serían $2Ta + 8t$

Para 20 elementos tendríamos que son $3Ta + 4t$

Juan Ignacio Alberola Colomo
jignacio.al@gmail.com
<http://sitio.de/jignacio>

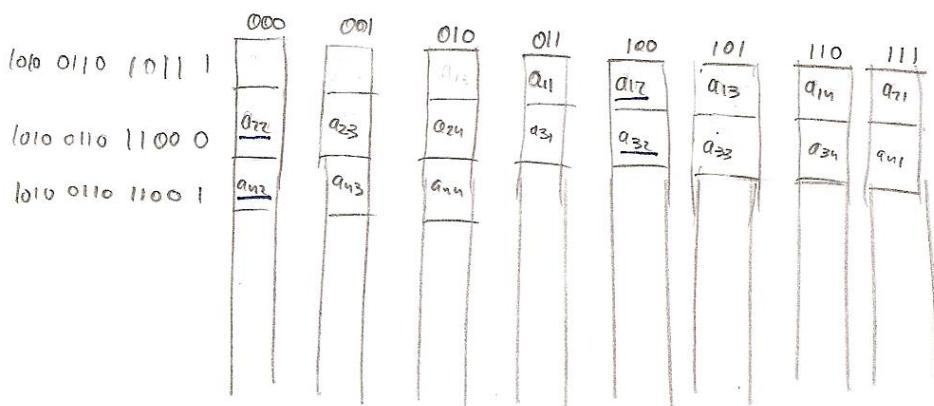
2) La memoria principal de un procesador vectorial con capacidad de 16 M palabras se encuentra distribuida entre 8 módulos usando 8 módulos y entrelazado de orden inferior y acceso de tipo S.

Suponiendo que el compilador ha almacenado en memoria por filas una matriz de 4×4 , se pide:

a) Obtener las posiciones de memoria en las que se sitúa la segunda columna teniendo en cuenta que el almacenamiento empieza en A6BBh

b) Pide el procesador leer la segunda columna realizando un único acceso a memoria

(Comienza en 1010 0110 1011 1011 b Entrelazado inferior)



La segunda columna está en

$$\begin{aligned} A6BCh &\rightarrow a_{12} \\ A6C0h &\rightarrow a_{22} \\ A6C4h &\rightarrow a_{32} \\ A6C8h &\rightarrow a_{42} \end{aligned}$$

b) Con un solo acceso a memoria no se puede leer todo la segunda columna ya que con el acceso simultáneo en cada acceso se pierden las posiciones a_{21} y a_{31}

$n-m$ dentro de cada módulo y en este caso solo están en la misma posición a_{21} y a_{31}

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

LIBRO) Suponiendo que el tiempo de latencia de inicio T_{LI} para una multiplicación vectorial es de 10 ciclos y que después TPC es de 1 ciclo por reloj ¿cuál es el número de ciclos por resultado, CPR , para un vector de 64 componentes?

$$T_{CV_k} = T_{LI} + (k \cdot TPC)$$

$$CPR = \frac{T_{CV_k} \text{ en ciclos}}{k}$$

$$T_{CV_{64}} = 10 + (64 \cdot 1) = 10 + 64 = 74 \text{ ciclos}$$

$$CPR = \frac{74}{64} = 1'15625 \text{ ciclos de reloj por resultado}$$

LIBRO) ¿Cuál es el tiempo de ejecución T_n y el número e ciclos por resultado, CPR , para la operación vectorial $A = B \cdot s$ donde s es un escalar y los vectores tienen 200 componentes ($n=200$)?

- Registros vectoriales de 64 componentes.
- $T_{BASE} = 10$, $T_{BUCLE} = 15$ y $TPC = 3$
- $T_{LI\,L1S} = 12$ ciclos, $T_{LI\,\text{mult}} = 7$ ciclos, $T_{LI\,\text{mem}} = 4$ ciclos

Creo que esto es fatal...

$$T_n = T_{BASE} + \left[\frac{k}{MVL} \right] (T_{BUCLE} + T_{LI}) + k \cdot TPC$$

$$T_n = 10 + \left[\frac{200}{64} \right] (15 + 12) + 64 \cdot 3$$

$\left. \begin{matrix} low = 1 \\ VL = 200 \bmod 64 \end{matrix} \right\} T_{BASE}$

$$T_{200} = 10 + 3 + 38 + 192 = 243$$

```

do j=0, 200
  do i = low, low+VL-1
    A(i) = B(i) * s
  continue
  low = low+VL;
  VL = MVL
} TBUCLE
    
```

\rightarrow lv v1, rB;
 multov v2, r1, #s $12 + 7 + 4 = 23$
 sv rx, v2

2 clase) ¿Cuanto tarda el siguiente código vectorial dentro de un bucle de n iteraciones?

- a) Sin encadenamiento.
- b) Con encadenamiento.

Juan Ignacio Alberola Colomo
jignacio.al@gmail.com
<http://sitio.de/jignacio>

TLLs

lw r0, a(r0)	6
lw r1, b(r0)	6
multv r3, r1, r3	12
lw r2, c(r0)	6
multsv r4, r0, r3	12
addv r0, r2, r4	6
sw r0, d(r0)	7

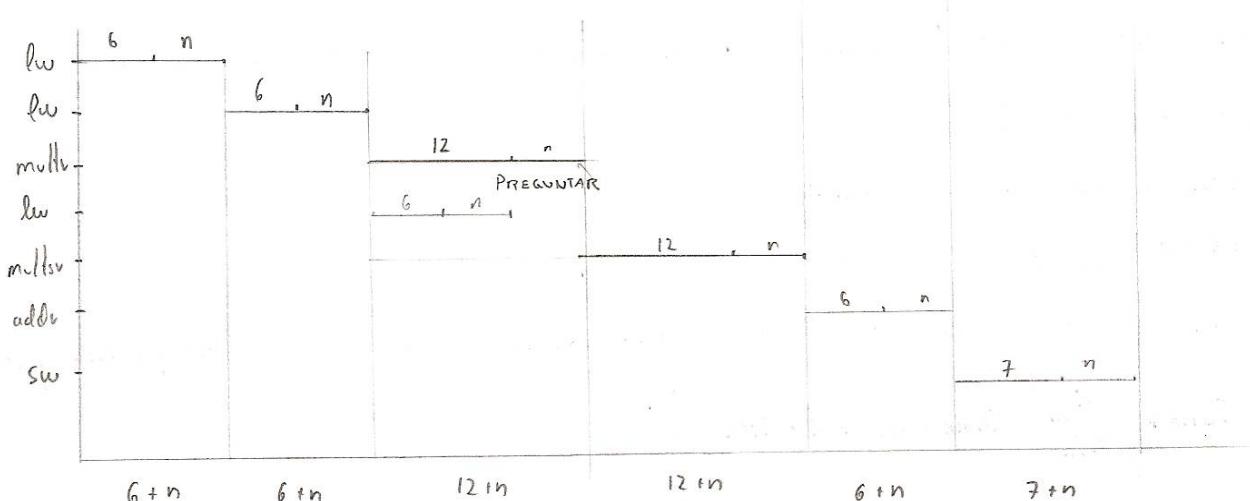
Este código está dentro de un bucle.

$$T_{\text{base}} = 25$$

$$T_{\text{buclle}} = 10$$

$$MVL = 64 \rightarrow$$

Los registros tienen como máxima capacidad 64.



$$3 \cdot 6 + 12 \cdot 2 + 7 + 6n = 18 + 24 + 7 + 6n = \underbrace{49}_{T_L} + \underbrace{6n}_{TPC}$$

$$T_n = T_{\text{BASE}} + \left\lceil \frac{n}{MVL} \right\rceil \cdot (T_{\text{BUCLLE}} + T_L) + n \cdot T_{\text{PC}}$$

$$T_n = 25 + \left\lceil \frac{n}{64} \right\rceil \cdot (10 + 49) + 6n = 25 + \left\lceil \frac{n}{64} \right\rceil \cdot 59 + 6n$$

$$= 25 + 59 \cdot \left(\frac{n}{64} + 1 \right) + 6n = 25 + \frac{59n}{64} + 59 + 6n = 84 + \frac{59n}{64} + 6n = 84 + \frac{443n}{64}$$

$$T_n = 84 + \frac{443n}{64}$$

$R_K = \text{Operaciones por unidad de tiempo para un vector de } K \text{ componentes.}$

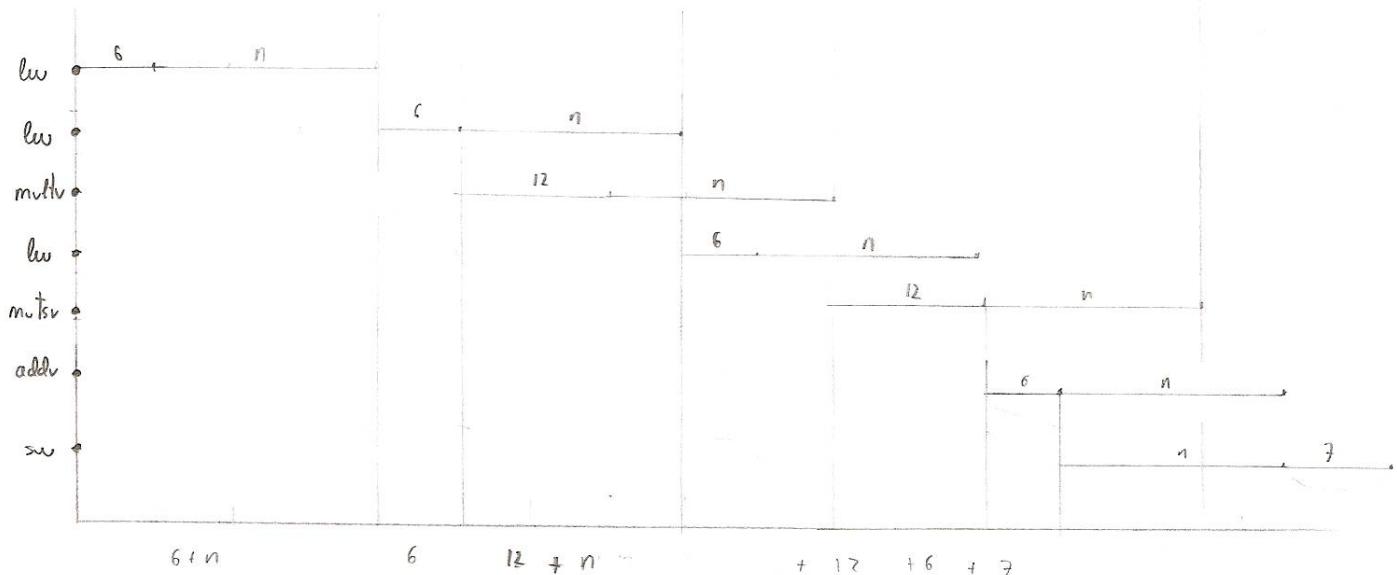
Juan Ignacio Alberola Colomo
jignacio.al@gmail.com
<http://sitio.de/jignacio>

$$R_K = \frac{\text{OpVecEj} \cdot K}{T_K}$$

$$R_{\infty} = \lim_{K \rightarrow \infty} R_K$$

$$R_{\infty} = \lim_{n \rightarrow \infty} \frac{3n}{84 + \frac{443n}{64}} = \lim_{n \rightarrow \infty} \frac{192n}{443n + \dots} = \frac{192}{443} \quad \circ 150 \text{ MHz} = 65.01 \text{ MFlops}$$

b) Con encadenamiento



$$6+n + 6 + 12+n + 12 + 6+n + 7 = \underbrace{49}_{TUI} + \underbrace{3n}_{TPC}$$

$$T_n = T_{\text{BASE}} + \left[\frac{n}{64} \right] \cdot (T_{\text{AVL}} + T_{\text{U}}) + T_{\text{PC}} n$$

$$T_n = 25 + \left[\frac{n}{64} \right] \cdot (10 + 49) + 3n = 25 + 59 \cdot \left[\frac{n}{64} + 1 \right] + 3n =$$

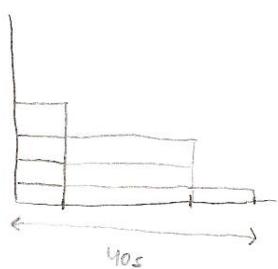
$$= 25 + 59 + \frac{59n}{64} + 3n = 84 + \frac{251n}{64}$$

$$R_{\infty} = \lim_{n \rightarrow \infty} \frac{3n}{84 + \frac{251n}{64}} = \lim_{n \rightarrow \infty} \frac{192n}{251n + \dots} = \frac{192}{251} \quad \circ 150 \text{ MHz} = 114 \text{ MFlops}$$

Ejercicios TEMAS. ARQUITECTURAS PARALELAS

1) Un programa tarda 40s en ejecutarse en un multiprocesador. Durante 70% de ese tiempo se ha ejecutado en 4 procesadores; durante 60% en 3; durante el 20% restante en un procesador. Despreciando sobrecarga y el trabajo se ha repartido equitativamente.

- a) ¿Cuánto tardaría en ejecutarse en 1 único procesador?
- b) ¿Cuál es la ganancia en velocidad obtenida con respecto al tiempo de ejecución secuencial?
- c) ¿Y la eficiencia?



$$T_S = 4 \cdot 0.7 \cdot 40 + 3 \cdot 0.6 \cdot 40 + 0.2 \cdot 40 = 32 + 72 + 8 = 112s$$

$$S(p) = \frac{112}{40} = 2.8 \quad 180\% \text{ más rápido el multiprocesador}$$

$$E(p) = \frac{2.8}{4} = 0.7$$

2) Un programa tarda 20s en ejecutarse en un procesador P1, y requiere 30s en otro procesador P2. Si se dispone de los dos procesadores para la ejecución del programa (despreciando sobrecarga).

a) ¿Qué tiempo tarda en ejecutarse el programa si la carga de trabajo se distribuye por igual entre los procesadores P1 y P2?

b) ¿Qué distribución de carga entre los 2 procesadores permite el menor tiempo de ejecución usando los dos procesadores en paralelo? ¿Cuál es este tiempo?

a) Si nos dicen que la carga se distribuye por igual $n=p$ Si paralleliza 2 tareas en 2 procesadores Reducimos la carga de cada procesador a la mitad.

el resto de carga

$$\left. \begin{array}{l} P_1 \rightarrow 10s \\ P_2 \rightarrow 15s \end{array} \right\} \text{Máximo } 15 \text{ segundos}$$

b) $20 \cdot \text{carga} = (1 - \text{carga}) \cdot 30$

$$20\text{carga} = 30 - 30\text{carga} ; 50\text{carga} = 30 ; \text{carga} = \frac{3}{5}$$

del total para P1 y $\frac{2}{5}$ para P2

3) ¿Cuál es la fracción de código ~~secuencial~~ paralelo de un programa secuencial que, ejecutado en paralelo en 8 procesadores tarda 100ns, durante 50ns usa 2 procesador y durante otros 50ns usa los 8 procesadores (distribución por igual) y se desprecia sobrecarga?

Hay parte secuencial y parte paralela.



50% del tiempo monoprocesador Nos piden ($1-f$)
50% del tiempo 8 procesadores.

$$T_s = 0'5 \cdot 100 + 8 \cdot 0'5 \cdot 100 = 50 + 400 = 450 \text{ de forma secuencial}$$

$$S(p) = \frac{450}{100} = 4'5$$

$$4'5 = \frac{8}{2+x(8-1)}$$

$$4'5 = \frac{8}{2+7x}; \quad 4'5 \cdot (1+7x) = 8; \\ 4'5 + 31'5x = 8;$$

$$31'5x = 8 - 4'5 \\ 31'5x = 3'5; \quad x = \frac{3'5}{31'5} = 0'11$$

La manera correcta es:

$$f = \frac{8 \cdot 50}{50 + 8 \cdot 50}$$

En un 11% la mejora no se usa

$$(1 - 0'11) = 0'89 \quad 89\% \text{ la mejora se usa}$$

4) Un 25% de un programa no se puede parallelizar, el resto se puede distribuir por igual entre cualquier número de procesadores (despreciamos sobrecarga)

¿Cuál es el máximo valor de la ganancia de velocidad que se podría conseguir al parallelizado?

¿A partir de cuál número de procesadores se podrían conseguir ganancias mayores o iguales que 2?

$$S(p) = \frac{P}{1+0'25(p-1)}$$

Máxima ganancia

$$2 \leq \frac{P}{1+0'25p-0'25}$$

$$2 \leq \frac{P}{0'25p+0'75}; \quad 2(0'25p+0'75) \leq P; \\ 0'5p + 1'5 \leq P;$$

$$1'5 \leq 0'5p$$

$$\frac{1'5}{0'5} \leq p$$

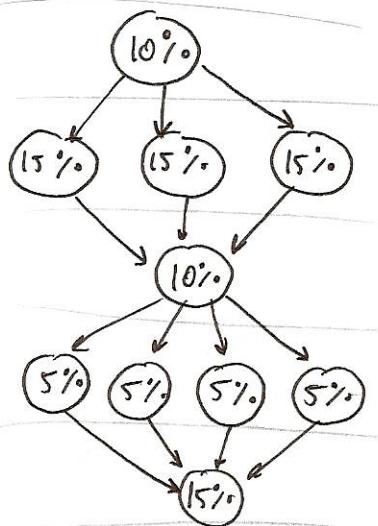
$$p = 3$$

5) En la figura se presenta el grafo de dependencia entre tareas para una aplicación.

La figura muestra la fracción de tiempo de ejecución secuencial que la aplicación tarda en ejecutar cada tarea.

Supongamos un tiempo de ejecución secuencial de 60s, que las tareas no se pueden dividir en tareas de menor granularidad y que tiempo de comunicación es despreciable, obtener el tiempo de ejecución en paralelo y la ganancia en velocidad con:

- a) 4 procesadores ; b) 2 procesadores.



$$\text{a) } T_p(n) = 0'1 \cdot 60 + 0'15 \cdot 60 + 0'1 \cdot 60 + 0'05 \cdot 60 + 0'15 \cdot 60 = \\ = 6 + 9 + 6 + 3 + 9 = 33 \text{ s}$$

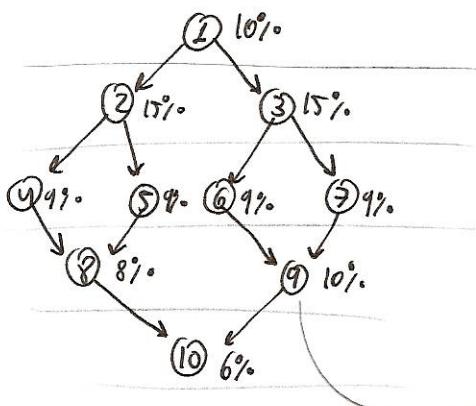
$$S(p,n) = \frac{60}{33} = 1'81 \quad 81\% \text{ más rápida en paralelo}$$

$$\text{b) } T_p(n) = 0'1 \cdot 60 + 0'15 \cdot 60 \cdot 2 + 0'1 \cdot 60 + 2 \cdot 0'05 \cdot 60 + 0'15 \cdot 60 = \\ = 6 + 18 + 6 + 6 + 9 = 45 \text{ s}$$

$$S(p,n) = \frac{60}{45} = 1\overline{33} \quad 33\overline{3}\% \text{ más rápido en paralelo}$$

6) Un programa se ha conseguido dividir en 10 tareas. El orden de precedencia entre las tareas se muestra en la figura. La ejecución de estas tareas supone 2s. El 10% debido a la tarea 2, el 15% a la 2, 15% a la 3, 4, 5, 6 y 7 suponen el 9%, la tarea 8 un 8%, la 9 un 10% y la 10 un 6%. Se disponen 8 procesadores

- a) ¿que tiempo tarda en ejecutarse el programa en paralelo?
- b) ¿que ganancia en velocidad se obtiene respecto a su ejecución secuencial?



$$\text{a) } T_p(n) = 0'17s + 0'157s + 0'097s + 0'17s + 0'067s = \\ = 0'57s = 1s$$

$$\text{b) } S(p,n) = \frac{2}{1} = 2 \quad 100\% \text{ más rápido}$$

Cogemos el que tarda más.

- Construcción de redes.
- Encaminamiento (directas e indirectas).

1) Se utiliza una malla abierta de 3 dimensiones que conecta 4096 nodos. Calcula el número de canales y commutadores para conectar el nodo 123 con el nodo 1200.

$$r = 16 \text{ para } N = 4096 \text{ nodos}$$

• Estructura cúbica.

$$\sqrt[3]{4096}$$

16 filas, 16 columnas, 16 profundidades

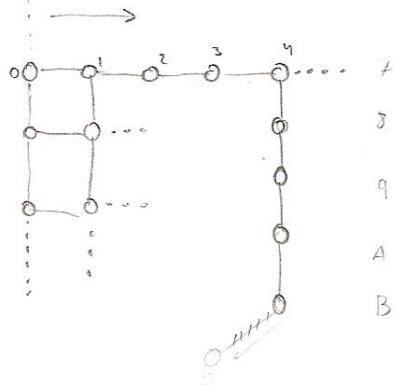
123
↓
1200

123 → 078h
1200 → 4B0h

Si el lado fuese alto habría que pasarlo a esa base.

↳ Tantos dígitos como dimensiones tiene la malla.

Cada dígito indica la posición en una dimensión.



$$\begin{aligned} \emptyset &\rightarrow 4 \quad (4 \text{ enlaces}) \\ 7 &\rightarrow B \quad (4 \text{ enlaces}) \\ B &\rightarrow \emptyset \quad (11 \text{ canales}) \end{aligned}$$

19 canales

Tener en cuenta el tipo de red con el que estás trabajando.

18 commu. sin contar inicio y final.
20 commu. contando inicio y final.

$$3(r-1)$$

Calcular la distancia mínima entre los nodos más alejados $\rightarrow 15 + 15 + 15 = \underline{45 \text{ canales}}$



DIÁMETRO

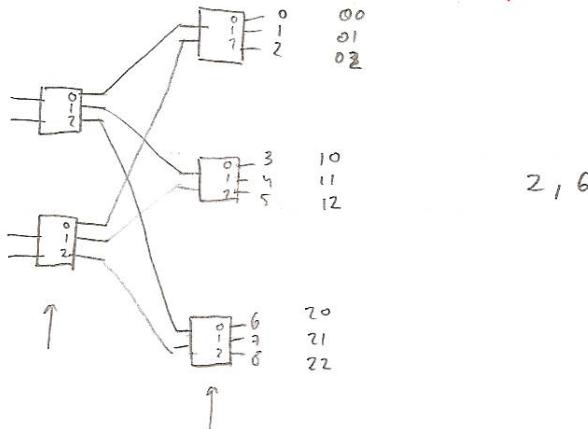
2) Construir red delta óptima que conecte 4 procesadores con 9 memorias y calcula las señales de control que hay que introducir por etapa para acceder a la memoria 6.

Juan Ignacio Alberola Colomo
jignacio.al@gmail.com
<http://sitio.de/jignacio>

$$\begin{array}{l} 4p \\ 9m \end{array} \quad \begin{array}{l} a^n \geq 4 \Rightarrow 2^2 \geq 4 \\ b^n \geq 9 \Rightarrow 3^2 \geq 9 \end{array}$$

n etapas

comutadores de 2×3



3) Comparar la técnica de conmutación wormhole y la Store and Forward en un toro bidimensional que usa canales bidireccionales y que unen 1024 nodos. Los canales tienen buffers de entrada y salida. y los tiempos de retraso son: $T_r = 6\text{ns}$, $T_{transf} = T_w = 7\text{ns}$, $T_s = 6\text{ns}$.

Suponemos un paquete de 100 phits + 2 phits extra de cabecera.

Utilizaremos la transferencia desde el nodo 522 al 56 por el camino más corto.

Red Toro 2D

$$\sqrt{1024} = 32$$

32 filas x 32 columnas

Base 32

$$\lceil \frac{24-15}{1} \rceil = 9 \text{ canales}$$

$$56_{32} = (1, 24)$$

$$522_{32} = (16, 15)$$

D = 24 canales

$$\lceil \frac{16-1}{1} \rceil = 15 \text{ canales}$$

↳ ¡OJO! El camino mínimo. Depende de la estructura de la red.

Wormhole:

$$t_v = D \cdot (T_r + T_s + T_w) + \max(T_s, T_w) \cdot \left\lceil \frac{L}{w} \right\rceil$$

$$t_v = 24 \left(6 + \frac{6}{2} + 7 \right) + 7 \cdot 100$$

||

1468ns

$$\frac{L}{w} \rightarrow \text{phit} = \frac{L}{w}$$

Store and Forward :

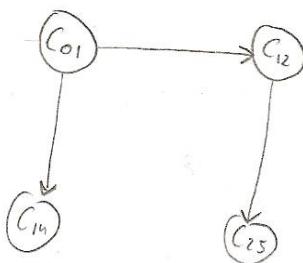
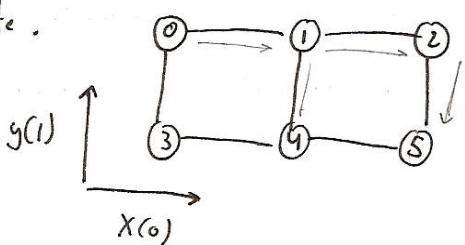
$$t_{AR} = D \cdot \left[t_v + (T_s + T_w) \left(\left\lceil \frac{L}{w} \right\rceil + 1 \right) \right]$$

$$t_{AR} = 24 \cdot \left[1 + (6+7) \cdot (102) \right]$$

Es mejor técnica que del Wormhole.

- Dibujar el grafo de dependencias entre canales para una malla 3×2 que sigue un encaminamiento 36

XY creciente.



Un grafo de dependencias solo partiendo desde 1 nodo

- La siguiente función de interconexión define una red barajador-intercambio.

$$B(h_{n-1}, h_{n-2}, \dots, h_1, h_0) = h_{n-2} h_{n-3} \dots h_1 h_0 h_{n-1}$$

$$L(\quad \quad \quad) = h_{n-1} h_{n-2} \dots h_1 h_0$$

siendo las h la representación en binario y el número de nodos $N=2^n$.

Dibujar la red para $N=8$.

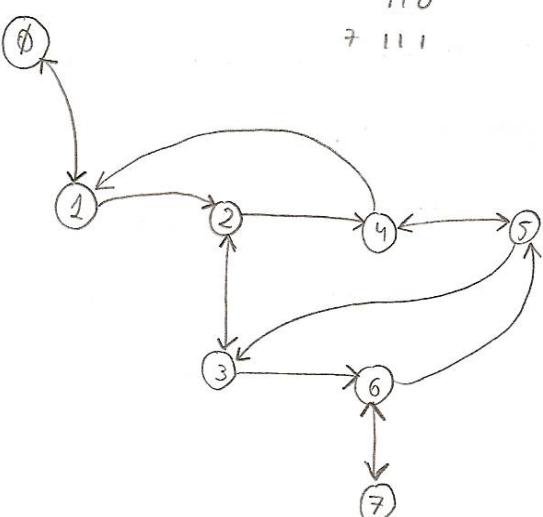
$$N=8 \quad n=3 \leftarrow 2^3$$

0 000
1 001

2 010
3 011

4 100
5 101

6 110
7 111



Una máquina NTP de 128 nodos usa un hipercubo como red de interconexión, la comunicación es de tipo DOR (empieza por la dimensión más alta). Se envía un paquete de p^{40} a p^{27} .

Indica el registro de encaminamiento del paquete.

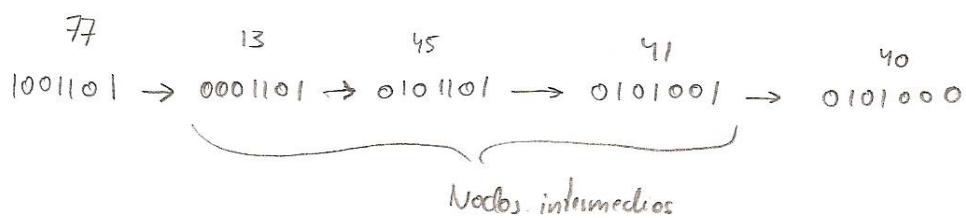
La distancia que recorrerà en la red.

Nodos intermedios por los que pasará.

$$N=128 \quad p77 = 1001101 \quad p40 = 0101000 \Rightarrow \begin{array}{r} 1001101 \\ 0101000 \\ \hline 1100101 \end{array} \xrightarrow{\text{XOR}} \text{registro de encaminamiento.}$$

La distancia es 4 (4 dimensiones).

Encaminamiento, negar la posición.



Un multicomputador con topología de cubo binaria y bidireccional y comunicación wormhole dispone de canales de 1 byte de ancho y $8 \cdot 10^8$ bits cada segundo de ancho de banda. Se dispone fibra óptica de un enrutador. La cabecera ocupa 1 bit, la cabecera son de 32 bits y los mensajes tienen 64 filas de datos.

Al sustituir el encaminador por otro el doble de rápido se observa que los tiempos de ~~transmisión~~ de la cabecera son los responsables del 14,5% de la latencia sufrida por los mensajes al recorrer una distancia igual al diámetro de la red con la red descargada.

¿Cuál es el retraso asociado al encaminador inicial?

Tiempo de Iilit

1 flit = 32 bits

$$f_w = \frac{32}{10.8} = 40 \text{ ns}$$

$$d \rightarrow \text{Diametro} \quad D = 3 \cdot \left[\frac{16}{2} \right] = 3 \cdot 8 = \underline{\underline{24}}$$

$$d = 24$$

$$\frac{L}{w} \cdot t_w \Rightarrow \frac{64}{40} = 2560 \text{ ns}$$

$$10^1 \cdot 2b \rightarrow 15$$

$$T_L = (x+40) \cdot 24 + 2560 = 24x + 3520$$

$$T_L = 24(f_{\eta_2}) + 35 \cancel{+} 20$$

$$\begin{array}{r} 7560 \\ 760 \\ \hline 3520 \end{array}$$

Responsable del 145% de la latencia.

$$0'145 = \frac{24\frac{1}{2}}{2} + 3.520$$

$$0.145 T_L = \frac{24 \pi}{3}$$

$$0'145 = \frac{12tr}{12tr + 3520} \Rightarrow 0'145(12tr + 3520) = 12tr \Rightarrow$$

Un determinado sistema de 256 procesadores utiliza como red de comunicación un toro de 2 dimensiones con enlaces de 10 Gbits/s. Se envía un paquete de L bytes desde el nodo p36 al p215. El tiempo de transmisión en la red se puede modelar como $T_{frames} = (d + L)/B$. Además de ello hay una serie de sobrecargas debidas al tráfico de 3ns por nodo, a los protocolos de generación y recogida de mensajes $T_{GES} = 100 + 0'1 \cdot L$ ns y a la cabecera + los datos de control son 8 bytes.

Calcular el tiempo total de la comunicación y el ancho de banda efectivo obtenido en estos dos casos.

- a) Se envían 8 bytes de datos.
- b) Se envían 256 bytes de datos.

Tenemos 256 nodos en 2 dimensiones. $\sqrt{256} = 16$

16 nodos de lado.

Renombramos los nodos en base 16

p36 $\frac{36}{16}$ Fila 2, Columna 4 (comenzando desde 0)

$\frac{0}{16}$ $\frac{2}{16}$ $\rightarrow 3$ (columna desde 0)

p215 $\frac{215}{16}$ Fila 13, Columna 6

$\frac{0}{16}$

(con $\frac{10 \text{ Gbits}}{4 \text{ s}}$ cuánto tarda en transferirse 1 byte

$\frac{11}{4 \text{ s}}$

8 bits

• Debemos calcular la distancia entre ambos

Seguimos esquema por reelección de dimensión.

$$13 - 2 = 11$$

Del 13 al 2 sin dar la vuelta son 11 enlaces que son mayores que $16/2$.

$$\text{Por lo que realmente } (16-13)+(2-0) = 3+2 = 5$$

$$7-4=3$$

La distancia total es 8

$$\left. \begin{array}{l} 10 \cdot 10^9 \rightarrow 1 \text{ s} \\ 8 \rightarrow x \end{array} \right\} = x = \frac{8}{10 \cdot 10^9} \cdot 4 \text{ s} = 0'8 \text{ ns}$$

$$\begin{aligned} \text{a)} \quad T_{TOTAL} &= T_{frames} + T_{Sob} + T_{GES} = 8 + (8+8) \cdot 0'8 \text{ ns} + 3 \cdot 8 + 100 + 0'1 \cdot (8+8) = \\ &= 8 + 16 \cdot 0'8 + 24 + 100 + 0'1 \cdot 16 = \\ &= 8 + 12'8 + 24 + 100 + 1'6 = 146'4 \text{ ns} \end{aligned}$$

$$\text{Ancho de banda efectivo} \Rightarrow Ba = \frac{8 \text{ bytes}}{146'4 \text{ ns}} = 0'05 \frac{\text{bytes}}{\text{ns}} \xrightarrow{\text{Ba} = \frac{64 \text{ bits}}{146'4 \cdot 10^9} = 4'37 \frac{\text{bits}}{\text{s}}}$$

b) $T_{TOTAL} = T_{fram} + T_{SOB} + T_{GES} = (8 + (8+256)) \cdot 0'8 \text{ ns} + 3 \cdot 8 + 100 + 0'1 \cdot (8+256) =$
 $(8 + 264) \cdot 0'8 + 24 + 100 + 0'1 \cdot 264 =$
 $217'6 + 24 + 100 + 26'4 = 360 \text{ ns}$

$B_a = \frac{256 \text{ bytes}}{360 \text{ ns}} = 0'711 \frac{\text{bytes}}{\text{ns}}$

Recordatorio que no tiene nada que ver pero que me he acordado de clase de esta mañana

La tabla de verdad de la puerta XOR es

$$\begin{array}{ccc} 0 & 0 & \rightarrow 0 \\ 0 & 1 & \rightarrow 1 \\ 1 & 0 & \rightarrow 1 \\ 1 & 1 & \rightarrow 0 \end{array}$$

Una aplicación se ejecuta en una máquina de 64 procesadores. El 80% de la misma se ejecuta en todos los procesadores. El 15% en 32 procesadores y el 5% restante en serie. El costo de comunicaciones se estima en un 5% del tiempo de ejecución ^{en paralelo} en cada caso. Tomando en cuenta únicamente estos aspectos, calcula el factor de aceleración.

Cuando hablamos de factor de aceleración se refiere a la ganancia. $S = \frac{T_{sec}}{T_{PAR}}$

$$T_{paralelo} = \left[0'8 \cdot \frac{T_s}{64} + 0'15 \cdot \frac{T_s}{32} \right] \cdot 1'05 + 0'05 T_s = S = \frac{T_s}{\text{↑}}$$

- La distancia entre los nodos 13 y 44 de un hipercubo de 6 dimensiones es: (calcular)

$$13_2 = 001101_2$$

$$44_2 = 101100_2$$

100001 → La distancia es 2

$$\begin{array}{r} 13_2 \\ \diagdown 6_2 \\ \diagdown 3_2 \\ \diagdown 1_2 \\ \diagdown 0 \end{array}$$

$$\begin{array}{r} 44_2 \\ \diagdown 22_2 \\ \diagdown 11_2 \\ \diagdown 5_2 \\ \diagdown 2_2 \\ \diagdown 1_2 \\ \diagdown 0 \end{array}$$

- El sistema de comunicación de un multicomputador emplea los siguientes tiempos en ns para enviar un paquete de $L = 128$ bytes a una distancia $d = 10$ siendo el ancho de banda de los enlaces $B = 1\text{Gbit/s}$. Si mejoramos la velocidad de transmisión de los enlaces hasta 4Gbit/s . ¿Cuántas veces más rápida será la comunicación emisor-receptor?

$$\text{Tiempos: } T_{\text{GEN}} = 200 + 10 \cdot L$$

$$T_{\text{TRANS}} = 5 \cdot d + \frac{L}{B}$$

$$T_{\text{RECEP}} = 200 + 10 \cdot L$$

Debemos calcular el tiempo con los 2 anchos de banda y calcular la ganancia.

$$B = \frac{1\text{Gbit}}{\text{s}} = \frac{10^9 \text{ bits}}{1\text{s}}$$

$$B = \frac{4\text{Gbit}}{\text{s}} = \frac{4 \cdot 10^9 \text{ bits}}{1\text{s}}$$

$$\left. \begin{array}{l} 10^9 \rightarrow 1\text{s} \\ 8 \text{ bits} \rightarrow x \end{array} \right\} x = \frac{8}{10^9} = 8\text{ns}$$

$$\left. \begin{array}{l} 4 \cdot 10^9 \text{ bits} \rightarrow 1\text{s} \\ 8 \text{ bits} \rightarrow x \end{array} \right\} x = \frac{8}{4 \cdot 10^9} = 2\text{ns}$$

$$\begin{aligned} a) T_{\text{TOTAL}} &= T_{\text{GEN}} \cdot 2 + T_{\text{TRANS}} = (200 + 10 \cdot 128) \cdot 2 + 5 \cdot 10 + 128 \cdot 8 = \\ &= 2960 + 50 + 1024 = \underline{4034\text{ns}} \end{aligned}$$

$$\begin{aligned} b) T_{\text{TOTAL}} &= T_{\text{GEN}} \cdot 2 + T_{\text{TRANS}} = (200 + 10 \cdot 128) \cdot 2 + 5 \cdot 10 + 128 \cdot 2 = \\ &= 2960 + 50 + 256 = \underline{3266\text{ns}} \end{aligned}$$

$$G = \frac{4034}{3266} = 1.235$$

23,5% más rápido con la mejora del ancho de banda.

- Un multicomputador usa una red con un $B = 16 \text{ bit/s}$ y la comunicación es store and forward. Manda un paquete de 32 bytes a $d=6$ cuesta 1'56μs. ¿Cuantas veces + rápida sería dicha transmisión si la comunicación fuese wormhole?

40

$$B = 16 \text{ bit/s} \quad 1 \text{ byte} \rightarrow 8 \text{ ns}$$

Hay que destacar que no siempre se tienen todos los datos

$$1'56\mu\text{s} = 1560 \text{ ns}$$

$$t_{AR} = 6 \cdot (t_r + 32 \cdot 8) = 6t_r + 1536 = 1560 ; t_r = 4$$

$$t_{WH} = \underbrace{6 \cdot (4 + 1 \cdot 8)}_{\text{cabecera}} + \underbrace{31 \cdot 8}_{\text{resto}} = 6 \cdot 12 + 31 \cdot 8 = 72 + 248 = 320 \text{ ns}$$

$$\frac{1560}{320} = 4'87 \text{ veces más rápido en WormHole}$$

- En un toro de 32×32 nodos. ¿Cuál es la distancia entre los nodos 900 y 257?

Lado 32

900 / 32
260 28
04 //

Fila 28, Columna 3

257 / 32
17 8
//
16

Fila 8, Columna 0

$$\text{Filas} \Rightarrow 28 - 8 = 20, \quad 20 > \frac{32}{2} \text{ por lo que hay que dar la vuelta}$$

$$\begin{array}{r} 0 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 21 \\ 22 \\ 23 \\ 24 \\ 25 \\ 26 \\ 27 \\ 28 \end{array}$$

$$\begin{array}{r} 8 - 0 = 8 \\ 31 - 28 = 3 \\ \hline 1 \\ 12 \end{array}$$

$$\text{Columnas} \Rightarrow 3 - 0 = 3$$

$$\boxed{\text{DISTANCIA TOTAL} = 12 + 3 = 15}$$

- La red de comunicación de un sistema masivamente paralelo es una malla de 16×16 en la que los nodos están conectados con 8 vecinos. Sobre dicha red se ejecuta una aplicación que envía mensajes entre los nodos a una distancia d con una probabilidad.

$$d = 1 \quad 3 \quad 7$$

$$\text{Prob} = 0.6 \quad 0.3 \quad 0.1$$

- ¿Cuál es el diámetro de la red?
- ¿Cuál es la distancia media de la aplicación?
- ¿Cuál es la distancia mínima que recorrerá un paquete enviado desde el nodo (3,5) al nodo (12,12)?
- El B de la biseción de la red es de 184 bit/s. ¿Cuál es el B de cada canal?

a) El diámetro es 15. En diagonal.

b) $0.6 \cdot 1 + 0.3 \cdot 3 + 0.1 \cdot 7 = 2.2$

c) Si redimensionamos en dimensión $12 - 3 = 9$ filas $12 - 5 = 7$ columnas Pero tenemos diagonales. Si avanzamos en diagonal 7 filas nos situamos en la misma columna. Solo faltan 2 más. La d mínima es 9

d) Biseción 46 enlaces $\frac{184}{46} = 4$ bits/s

- Un multiprocesador de memoria distribuida con red de interconexión toro bidimensional, canales bidireccionales y 1024 nodos terminales. Los conmutadores de la red tienen buffers asociados a las entradas y salidas. Esto hace que las técnicas de conmutación que aprovechan para la transferencia la segmentación en etapas del camino por parte de los buffers, se tenga que tener en cuenta, al obtener la latencia de transporte, que hay etapas que suponen atravesar un conmutador y etapas que implican atravesar el canal entre conmutadores.

Los tiempos que determinan el retardo de comunicación en el sistema son:

$$t_{\text{sobrecarga}} = 10 \mu\text{s} . \quad \text{Tiempo de sobrecarga en la fuente + tiempo de sub en el destino.}$$

$$t_r = 6 \text{ ns} . \quad \text{Tiempo de encaminamiento en el conmutador.}$$

$$t_w = 7 \text{ ns} . \quad \text{Tiempo de trans de 2 phits entre conmutadores.}$$

$$t_s = 6 \text{ ns} . \quad \text{Transferencia entre buffers.}$$

Supongamos un paquete de 100 phits + 2 phits de cabecera

Calcular la latencia que supone la transferencia de este paquete desde el nodo S27 al S6 por el camino más corto.

Primero calcularemos la distancia entre los dos nodos. $\sqrt{1024} = 32$ conmutadores de lado

S27 a S6

$$\begin{array}{r} 527 \longdiv{32} \\ 207 \quad 16 \longdiv{32} \\ \hline 15 \quad 16 \end{array}$$

$$\boxed{(16, 15)}$$

$$\begin{array}{r} 56 \longdiv{32} \\ 2 \quad 1 \longdiv{32} \\ \hline 0 \end{array}$$

$$\boxed{(1, 24)}$$

$$\begin{array}{r} 16 - 1 = 15 \\ 24 - 15 = 9 \\ \hline 24 \end{array}$$

$$D = 24$$

a) Si se utiliza conmutación vermicompare (consideramos que un flit consta de 2 phits)

$$t_r = t_{cabecera} + t_{resto} \Rightarrow t_r = D \cdot (t_r + t_w) + t_w \cdot \left\lceil \frac{L}{w} \right\rceil \quad \text{Buffer solo en las entradas}$$

$$\Rightarrow t_r = D \cdot (t_r + t_s + t_w) + \max(t_s, t_w) \cdot \left\lceil \frac{L}{w} \right\rceil$$

$$\begin{aligned} t_r &= 24 \cdot (6 + 6 \cdot 2 + 7 \cdot 2) + \max(6, 7) \cdot 100 = \\ &= 24 \cdot (6 + 12 + 14) + 7 \cdot 100 = 24 \cdot 32 + 700 = 1468 \text{ ns} \end{aligned}$$

b) Si se utiliza conmutación store and forward.

$$t_{AR} = D \cdot \left[t_r + t_w \cdot \left(\left\lceil \frac{L}{w} \right\rceil + 1 \right) \right] = D \cdot (t_r + t_w) + D \cdot t_w \cdot \left\lceil \frac{L}{w} \right\rceil$$

$$t_{AR} = 24 \cdot \left[6 + (7+6) \cdot 102 \right] = 24 \cdot \left[6 + 13 \cdot 102 \right] = 24 \cdot (6 + 1326) = 24 \cdot 1332 = 15984 \text{ ns}$$

c) Si se usa conmutación de circuitos con camino segmentado (1 flit = 2 phits y que la sonda y el reconocimiento son 1 flit)

$$t_{CC_{segm}} = \left\{ D \cdot (t_r + 2 \cdot t_w) + 2 \cdot t_w \right\} + \left\{ t_w + D \cdot t_w + t_w \cdot \left(\left\lceil \frac{L}{w} \right\rceil - 1 \right) \right\}$$

5) Dibujar una red baraje que conecte cuatro procesadores con nueve módulos de memoria
 ¿Se pueden realizar simultáneamente los siguientes accesos a memoria?
 (P_0, M_4) (P_1, M_5) (P_2, M_8) (P_3, M_6) .

¿Por qué?

¿Y se utiliza una red de bamas cruzadas?

Baraje o red delta

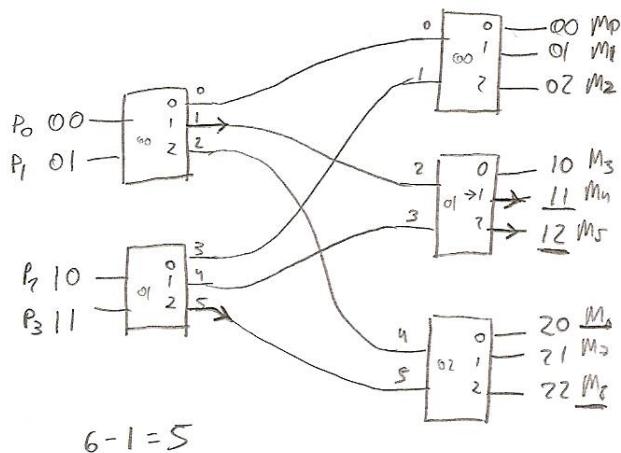
$$a^n \cdot b^n \Rightarrow 4 \cdot 9 \Rightarrow 2^2 \cdot 3^2 \quad \text{Commutadores de } 2 \times 3$$

n etapas de commutadores $\Rightarrow 2$ etapas de commutación

$$a^{n-1-i} \cdot b^i \text{ commutadores en cada etapa } C_i$$

$$C_0 \rightarrow 2^{2-2-0} \cdot 3^0 = 2 \cdot 1 = 2 \text{ comm}$$

$$C_1 = 2^{2-1-1} \cdot 3^1 = 1 \cdot 3 = 3 \text{ comm}$$



$$B_6^2(0) = 2 \cdot 0 = 0$$

$$B_6^2(4) = (2 \cdot 4) \bmod(5) = 3$$

$$B_6^2(1) = (2 \cdot 1) \bmod(5) = 2 \cdot 1 = 2$$

$$B_6^2(5) = 6-1=5$$

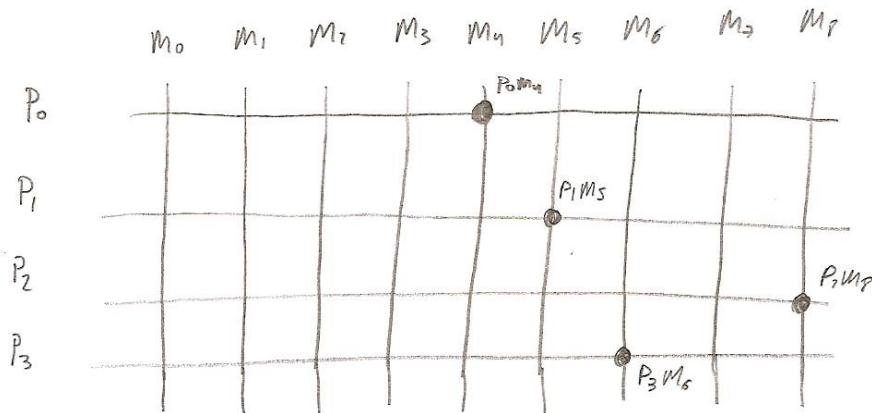
$$B_6^2(2) = (2 \cdot 2) \bmod(5) = 2 \cdot 2 = 4$$

Para el caso de P_0, M_4 y P_1, M_5 no se puede porque en el commutador 1 de C_1 tendrían que haber dos señales diferentes al mismo tiempo.

$$B_6^2(3) = (2 \cdot 3) \bmod(5) = 1$$

Para el caso de P_2, M_8 y P_3, M_6 tampoco, por la misma causa pero en 2 de C_1 .

Red de barras cruzadas



Sin problemas para acceder paralelamente.

6) Dibujar una red cubo dinámica multietapa que conecte 8 procesadores con 8 módulos de memoria.

Reposición de algunos conmutadores de la red para obtener una configuración de red multietapa Omega.

Red cubo

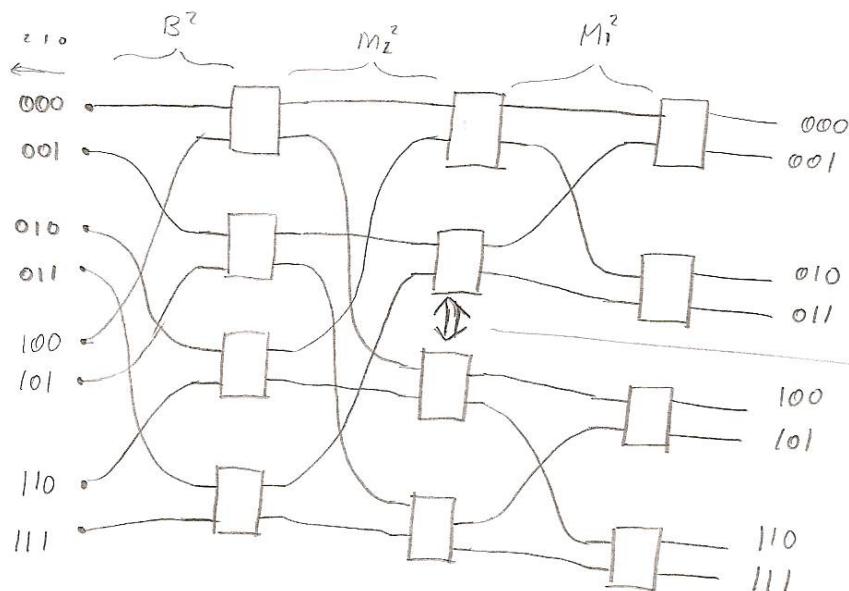
$$K^n \cdot K^n \Rightarrow 8 \cdot 8 \Rightarrow 2^3 \cdot 2^3$$

Conmutadores de $K \cdot K$ \Rightarrow conmutadores de $2 \cdot 2$

¿Reposición?

n etapas ($i \Rightarrow 3$ etapas, C_0, C_1, C_2)

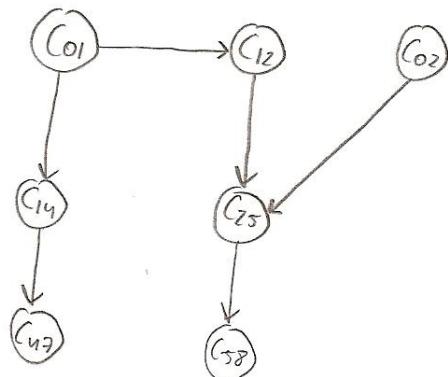
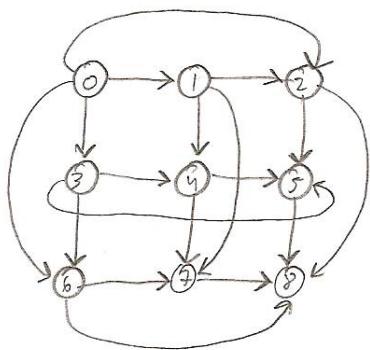
$$K^{n-1} \text{ conm por etapa} \Rightarrow 2^2 = 4$$



Si intercambiamos los nodos
obtenemos una red multietapa

OMEGA

- (b) En un toro bidimensional de base 3 y con canales unidireccionales (sin canales virtuales)
se utiliza un encaminamiento determinístico ordenado por dimensión (orden creciente)
- a) Dibujar el grafo de dependencias entre canales para la red.



EJERCICIOS DE EXAMENESREDES DE INTERCONEXIÓN

1 Punto.

Se desea implementar una red para conectar 64 procesadores. En su equipo de diseño se tienen dudas para decidir el tipo de red y se barajan dos posibilidades: Una red hiper cubo n-dimensional y una red delta con conmutadores crossbar 2x3.

- a) Explica si es posible usar cualquiera de las dos redes y argumenta cual sería la mejor basandote en el coste de componentes (a más conexiones, más coste) y la conectividad (distancia).

Red delta

$64 \times 64 \rightarrow 2^6 \times 3^6$ Habría 6 etapas. Si aunque quedarian conexiones libres en uno de los extremos

$$\begin{aligned} C_0 &\text{ tendría } 2^{6-1-0} \cdot 3^0 = 32 \text{ conmutadores} \\ C_1 &\text{ tendría } 2^{6-1-1} \cdot 3^1 = 48 \text{ conmutadores} \\ C_2 &\text{ tendría } 2^{6-1-2} \cdot 3^2 = 72 \text{ conmutadores} \\ C_3 &\text{ tendría } 2^{6-1-3} \cdot 3^3 = 108 \text{ conmutadores} \\ C_4 &\text{ tendría } 2^{6-1-4} \cdot 3^4 = 162 \text{ conmutadores} \\ C_5 &\text{ tendría } 2^{6-1-5} \cdot 3^5 = 243 \text{ conmutadores} \end{aligned}$$

Quedarian 665 conexiones libres en un extremo

Se tendrían conmutadores despredicados

Se tendría un total de

$$32 \cdot 3 + 48 \cdot 3 + 72 \cdot 3 + 108 \cdot 3 + 162 \cdot 3 = 1266 \text{ conexiones en las subredes.}$$

Diámetro sería $6+1 = 7$

Red hiper cubo

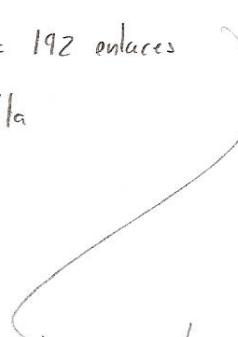
$\log_2 64 = 6$ Necesitaremos un hiper cubo de 6 dimensiones

$$\text{Enlaces necesitados} = \dim \cdot 2^{\dim - 1}$$

$$\text{Se necesitarian } 6 \cdot 2^5 \text{ enlaces} = 192 \text{ enlaces}$$

Bastantes menos que en la red delta

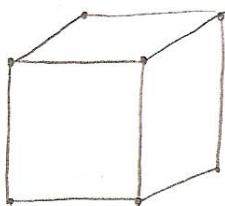
El diámetro = 6



Para este caso woria la red hiper cubo

En red hiper cubo

b) Pon un ejemplo del enrutamiento de 3 dimensiones. (hipercubo)



Ir del nodo 0 al 5

$$0 \rightarrow 000$$

$$5 \rightarrow 101$$

$$\begin{array}{r} 000 \\ 101 \\ \hline 101 \end{array} \text{ XOR}$$

$$000 \xrightarrow{0} 100 \xrightarrow{1} 101$$

1 Punto

Se dispone de una red toro 2D que interconecta 256 procesadores y que es capaz de proporcionar un ancho de banda de 8Mbits/s. Tras varias medidas se obtiene que el tiempo de enrutamiento $t_r = 2ns$.

Se tienen dudas sobre el mecanismo de conmutación a elegir entre Store and Forward y Wormhole. ¿Cuál recomendaría?

Basa tu recomendación en cálculos de latencia realizados para enviar un mensaje de 32 bytes

30 de datos + 2 de cabecera desde el nodo 6 hasta el nodo 200.

$$8 \text{ Mbits/s} = 8 \cdot 10^6 \frac{\text{bits}}{\text{s}}$$

$$\left. \begin{array}{l} 8 \cdot 10^6 \rightarrow 1s \\ 8 \rightarrow 1s \end{array} \right\} 1 \cdot 10^{-6} \text{ s}$$

$$10^{-6} \cdot 10^{-9} = 1000 \text{ ns}$$

$$256 \text{ procesadores} \rightarrow 2D \quad \sqrt{256} = 16 \quad r = 16$$

F C

$$6_{10} = (0, 6)$$

$$\left. \begin{array}{l} 0 \rightarrow 12 \text{ hay } 12 > 8 \text{ por lo tanto } 16 - 12 = 4 \\ 6 \rightarrow 8 \text{ hay } 2 < 8 \text{ por lo tanto } = 2 \end{array} \right\} D = 6$$

$$200_{10} = (12, 8)$$

$$\underline{\text{Store and Forward}} \quad t_{SF} = D \cdot \left[t_r + t_w \cdot \left(\left\lceil \frac{L}{w} \right\rceil + 1 \right) \right]$$

$$\begin{array}{c} 200 \underline{16} \\ 040 \quad 12 \underline{16} \\ 08 \end{array}$$

$$t_{SF} = 6 \cdot \left[2 + 1000 \cdot 32 \right] = 6 \cdot 32002 = 192012 \text{ ns}$$

$$\underline{\text{Wormhole}} \quad t_{WH} = \text{Cabecera} + T_{datos} = D \cdot (t_r + t_w) + t_w \cdot \left\lceil \frac{L}{w} \right\rceil$$

$$T_w = 6 \cdot (2 + 2 \cdot 1000) + 1000 \cdot 30 = 6 \cdot (2 + 2000) + 30000 =$$

$$= 6 \cdot 2002 + 30000 = \underline{\underline{42012}}$$

Recomendaría el Wormhole

1 punto

Un sistema multiprocesador utiliza una red de interconexión con un tiempo de enrutamiento = 6ns. La transmisión más común son paquetes de 128 bytes (1 flit = 1 byte). La cabecera son 2 bytes y corresponden a los 2 primeros de los 128.

Enlaces tienen velocidad de 1 Mbit/s

¿Cuánto tardaría el paquete en recorrer la distancia $D=6$?



$$T = 6 \cdot 6 + 8000 \cdot 128 \cdot 6 = 36 + 6144000 = 6144036 \text{ ns}$$

$$\frac{1 \text{ Mbit}}{\text{s}} = \frac{10^6 \text{ bits}}{\text{s}}$$

$$\left. \begin{array}{l} 10^6 \text{ bits} \rightarrow 1 \text{s} \\ 8 \text{ bits} \rightarrow x \end{array} \right\} 8 \cdot 10^{-6} \text{ s} = 9000 \text{ ns}$$

1 punto

Un sistema multiprocesador utiliza una red de interconexión hiperícuo de 6 dimensiones. Los enlaces tienen una velocidad de 1 Gbit/s. Para terminar de ajustar su diseño se desea conocer lo siguiente.

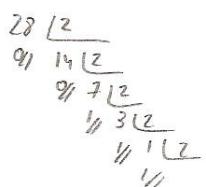
a) Número de enlaces que conectan el nodo 2 con el 28.

$$2_{10} = 000010_2$$

Registro de desplazamientos

$$28_{10} = 011100_2$$

$$\begin{array}{r} 000010 \\ 011100 \\ \hline 011110 \end{array}$$



$$000010_2 \rightarrow 010010 \rightarrow 011010 \rightarrow 011110 \rightarrow 011100 \quad 4 \text{ enlaces intermedios}$$

b) Como técnicas de conmutación se está decidendo entre escoger cut-through o store-and-forward. El tr es 4ns. La transmisión más común son 64 bytes (1 flit = 1 byte). La cabecera es 1 flit y corresponde al primero de los 64 bytes enviados. ¿Qué técnica sería la mejor para enviar datos del nodo 2 al 28?

$$\frac{1 \text{ Gbit}}{\text{s}} = \frac{10^9 \text{ bits}}{\text{s}}$$

$$\left. \begin{array}{l} 10^9 \text{ bits} \rightarrow 1 \text{s} \\ 8 \text{ bits} \rightarrow x \end{array} \right\} \frac{D}{10^9} = 8 \cdot 10^{-9} \text{ s}$$

$$T_{AR} = 4 \cdot [4 + 8 \cdot 64] = 4 \cdot [4 + 512] = 2064 \text{ ns}$$

$$\begin{aligned} T_H &= 4 \cdot (4+2) + 8 \cdot 63 = \\ &= 48 + 504 = 552 \end{aligned}$$

Mejor Cut-Through

8ns

1º)

Responde a las siguientes preguntas

- a) ¿Cuál sería el número de etapas necesaria para diseñar una red delta con commutadores 2×3 que conectara 6 procesadores con 10 módulos de memoria? ¿Cuántos módulos habría en cada etapa?

$$\begin{array}{l} \text{Red delta } a^n \times b^n \quad 2^n \times 3^n \\ n=3 \text{ etapas de commutadores} \\ \downarrow \quad \begin{array}{c} C_0 \quad C_1 \quad C_2 \\ \downarrow \\ 2^{3-2-1} \cdot 3^1 = 2^0 \cdot 3 = 6 \\ \downarrow \\ 2^{3-1-0} \cdot 3^0 = 2 \cdot 1 = 2 \end{array} \end{array}$$

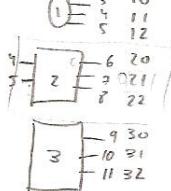
- b) En la red del apartado a), ¿cuál sería el control para ir desde el procesador 4 al módulo de memoria 8?

$$\text{Procesador } 4 \quad \text{Entradas } 6 \text{ procesadores} \quad \frac{4}{2} = \text{comutador } 2 \quad \frac{8}{3} = 2 \text{ comutador } 2$$

$$\text{En } C_1 \text{ hay } 6 \text{ entradas} = 6 \cdot 3 = 18 \text{ salidas } 0..17$$

concretamente

$$\text{En } C_2 \text{ hay } 9 \text{ entradas} = 9 \cdot 2 = 18 \text{ salidas } 0..17$$



$$4 = 2 \cdot x \bmod 17 \quad x = 2 \text{ salida } 2 \text{ de } C_1 \rightarrow 02$$

Salida 2 corresponde con el comutador 0 de C_1

En el comutador 0 de C_1 tenemos que las entradas son 0 y 1

$$0 = 2 \cdot x \bmod 11 \quad x = 0 \quad \text{por la salida } 0$$

$$C_0 \rightarrow 00$$

$$C_1 \rightarrow 02$$

$$C_2 \rightarrow 22$$

EJERCICIOS CONSISTENCIA DE CACHÉ (CLASE Y EXAMENES)

De momento, recordando el ejercicio de clase...

Disponemos de un multiprocesador con 4 procesadores con caché local y un procesador sin caché.

Los procesadores están todos conectados a través de un bus en el que se encarga de la coherencia de los datos un protocolo de sondeo o snoopy del tipo Illinois (MESI). Los cachés son de 2M palabras y el tamaño del bloque es 1 palabra. La caché es totalmente asociativa y la política de reemplazo es aleatoria.

Indicar cómo evolucionaría el estado de las memorias caché y de la mem principal.

ESTADO INICIAL

Caché1			Cache2			Cache3			Cache4			Mem Principal			
Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	@1	5	@2	9
@1	5	E	@1	-3	I	@2	9	S	@2	9	S	@3	10	@4	10
@2	9	S	@5	2	E	@3	6	M	@4	0	M	@5	2	@6	9
rd1(@1)															

No produce cambios ni mensajes en el bus. Es una lectura de un bloque que está en estado Exclusivo, quiere decir que es la única caché con una copia válida. La memoria lib tiene copia actualizada.

[wr1(@1,22)]

Acceso de escritura en bloque modificando o exclusivo

Como no hay otra copia válida no se genera paquete en el bus. Si esté en exclusivo pasa a modificando

@1 22 M	@1 -3 I	@2 9 S	@2 9 S	@1 5 @2 9
@2 9 S	@5 2 E	@3 6 M	@4 0 M	@3 10 @4 10 @5 2 @6 9

[wr3(@2,-11)]

Acceso de escritura en bloque compartido.

Escríbo en un bloque compartido (válido en esa caché, en mem y en al menos otra caché).

El estado cambia a modificando. Si otra caché tiene el bloque pasa a inválido.

El bloque no puede estar en ex o mod en otra caché

@1 22 M	@1 -3 I	@2 -11 M	@2 9 Z	@1 5 @2 9
@2 9 I	@5 2 E	@3 6 M	@4 0 M	@3 10 @4 10 @5 2 @6 9

[rd2(@4)]

La cache 2 no tiene el bloque @4 . Fallo de lectura.

En este caso hay que hacer un reemplazo . Reemplazamos el invalido . Pasamos a Compartida el bloque nuevo y los de los demás . Además se escribe en memoria.

@1 22 M	@4 0 S	@2 -11 M	@2 9 I	@1 5	@2 9
@2 9 I	@5 2 E	@3 6 M	@4 0 S	@3 10	@4 0

@5 2 @6 9

[rd3(@5)]

Caché 3 no tiene bloque @5 . Fallo de lectura

@1 22 M	@4 0 S	@2 -11 M	@2 9 I	@1 5	@2 9
@2 9 I	@5 2 S	@5 2 S	@4 0 S	@3 6	@4 0

@5 2 @6 9

[rd1(@2)]

Si que tiene @2 , pero como sino pq lo tiene en estado invalidado.

Lo leemos de otra caché con estado M o de mem

@1 22 M	@4 0 S	@2 -11 S	@2 -11 S	@1 5	@2 -11
@2 -11 S	@5 2 S	@5 2 S	@4 0 S	@3 6	@4 0

@5 2 @6 9

[rd4(@2)]

Acceso de lectura → wProc (@5,-9)

@1 22 m	@4 0 S	@2 -11 S	@2 -11 S	@1 5	@2 -11
@2 -11 S	@5 2 I	@5 2 I	@4 0 S	@3 6	@4 0

@5 -9 @6 9

Hacer lo mismo con el siguiente estado inicial. Aplicar MESI.

Cache1			Cache2			Cache3			Cache4			Mem Principal		
Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est
@6 3 M	@3 -3 I		@2 9 I	@4 7 M		@3 6 E	@1 6 S		@1 6 @2 -4	@3 106 @4 10		@5 2 @6 7		
@1 6 S	@5 2 I													

rd2(@1)

Cache2 no tiene @1. Fallo de lectura, se lanza una señal de petición de lectura, hay un reemplazo en Cache3 y el estado sera Shared.

@6 3 M	@1 6 S	@2 9 I	@4 7 M	@1 6 @2 -4
@1 6 S	@5 2 I	@3 6 E	@1 6 S	@3 106 @4 10 @5 2 @6 7

rd3(@6)

Cache3 no tiene @6. Fallo de lectura, se lanza señal de petición de lectura. Reemplazo en cache3 y estado sera Shared

@6 3 S	@1 6 S	@6 3 S	@4 7 M	@1 6 @2 -4
@1 6 S	@5 2 I	@3 6 E	@1 6 S	@3 106 @4 10 @5 2 @6 3

wr4(@1,2)

Acabó con estado compartido. Manda petición de lectura exclusiva e invalida los compartidos @1

@6 3 S	@1 6 I	@6 3 S	@4 7 M	@1 6 @2 -4
@1 6 I	@5 2 I	@3 6 E	@1 2 M	@3 106 @4 10 @5 2 @6 3

wr2(@4,-7)

Fallo de escritura. Se envía señal de lectura exclusiva, el nuevo estado pasa a estar M

@6 3 S	@4 -7 M	@6 3 S	@4 7 I	@1 6 @2 -4
@1 6 I	@5 2 I	@3 6 E	@1 2 M	@3 106 @4 7 @5 2 @6 3

wry (@1,15)

Acento de escritura con estodo Modificado

@6 3 S	@4 -7 M	@6 3 S	@4 7 M	@1 6 @2 -4
@1 6 I	@5 2 I	@3 6 E	@1 15 M	@3 106 @4 7
				@5 2 @6 3

wr3(@3,-11)

Acento de escritura

@6 3 S	@4 -7 M	@6 3 S	@4 7 M	@1 6 @2 -4
@1 6 I	@5 2 I	@3 -11 M	@1 15 M	@3 106 @4 7
				@5 2 @6 3

Preguntar ...

Más de lo mismo, no sé si bien o si mal...

Estado inicial

Cache1			Cache2			Cache3			Cache4			Mem Principal		
Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est	Dir	Dat	Est
@4 10 M	@3 -3 I		@2 9 I			@6 5 M			@1 9			@2 -3		
@1 9 S	@5 2 I		@3 6 E			@1 9 S			@5 -1			@4 10		

rd3(@2)

Fallo de lectura . Envía petición de lectura . Como ninguna otra cache tiene @2 se obtiene del mem princip. y se pone a modo Ex

@4 10 M	@3 -3 I	@2 -3 E	@6 5 M	@1 9	@2 -3
@1 9 S	@5 2 I	@3 6 E	@1 9 S	@3 6	@4 10

wrL(@4, x)

Aumento de escritura .

@4 X M	@3 -3 I	@2 -3 E	@6 5 M	@1 9	@2 -3
@1 9 S	@5 2 I	@3 6 E	@1 9 S	@3 6	@4 10

rd3(@6)

Fallo de lectura . Mandamos pet. lect y cambiamos el resto a S . Actualizamos mem

@4 X M	@3 -3 I	@6 5 S	@6 5 S	@1 9	@2 -3
@1 9 S	@5 2 I	@3 6 E	@1 9 S	@3 6	@4 10

wr4(@3, -7)

Fallo de escritura . Lanzamos pet de lectura exclusiva . Escribimos con modo M y los demás inválidos

@4 X M	@3 -3 I	@6 5 S	@3 -7 M	@1 9	@2 -3
@1 9 S	@5 2 I	@3 6 I	@1 9 S	@3 6	@4 10

rd3(@3)

Fallo de lectura . Mandamos señal al pet de lect . Obtenemos @3 de C1 y cambiamos a modo shared . Actualizamos memoria

@4 X M	@3 -3 I	@6 5 S	@3 -7 S	@1 9	@2 -3
@1 9 S	@5 2 I	@3 -7 S	@1 9 S	@3 -7	@4 10

rd3(@2)

Fallo de lectura . Hacemos Rd de lectura , radio tiene @2 , cogemos de mem principio y ponemos en modo exclusivo .

@4 x M	@3 -3 I	@6 S S	@3 -7 S	@1 9	@2 -3
@1 9 S	@5 2 I	@2 -3 E	@1 9 S	@3 -7	@4 10
				@5 -1	@6 S

wrl(@4,5)

@4 S M lo demás es igual

wrl(@5,-2)

@5 -2 M

Preguntar para corrección...

Ahora los otros ejercicios, de otro tipo. Calcular la memoria extra.

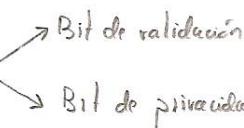
- Se está planificando la implantación de un sistema híbrido que combina los protocolos de sardes con protocolos basados en directorio P+1 centralizada. Lo implementamos usando clusters con 16 procesadores cada uno y usan coherencia de cache tipo snoopy.

Tenemos un total de 512 procesadores en 32 clusters. Con cada cluster se incorpora hardware adicional para mantener ... del directorio.

- Estimar la sobrecarga de memoria necesaria (bits extras) asumiendo 1 Gbyte de mem por cluster, 1 Mbyte de caché por procesador y bloques de cache de 128 bytes.

Memoria Extra

En cada cache hay que añadir 2 bits



$$1 \text{ MByte cada cache} = 2^{20} \cdot 2^3$$

$$\frac{2^{23}}{2^7} = 2^{16} = 64 \text{ K bloques}$$

$$1 \text{ bloque son 128 bytes} = 2^7$$

caché

$$512 \text{ procesadores} \Rightarrow 512 \text{ caches}$$

$$2 \text{ bits} \cdot 64 \text{ K} \cdot 512 = 2^1 \cdot 2^6 \cdot 2^{10} \cdot 2^9 = 2^{26} = 8 \text{ Mbytes adicional de caché}$$

Bits para directorio

$$P+1 \text{ bits} = 32 + 1 = 33$$

$$\frac{1 \text{ GByte}}{128 \text{ Bytes}} = \frac{2^{30} \cdot 2^3}{2^7} = \frac{2^{33}}{2^7} = 2^{26} = 64 \text{ M bloques}$$

caché

$$33 \cdot 64 \text{ M} \cdot 32 = 1056 \cdot 64 \cdot 2^1 = 67584 \text{ Mbytes}$$

↳ cantidad de directorios

PRUEBAS... DIVAGACIONES... ALGO...

$$\frac{32 \text{ GB}}{128 \text{ bytes}} = \frac{2^{30} \cdot 2^3}{2^7} = \frac{2^{33}}{2^7} = 2^{26} = 64 \text{ M bloques}$$

caché

$$33 \cdot 16 = 33 \cdot 2^4 \cdot 2^{10} = 67584 \text{ Mbytes}$$

67584 Mbytes COINCIDE

- Se quiere diseñar un MP escalable de memoria compartida con cachés privadas asociadas a cada procesador. La máquina usará un protocolo de coherencia basado en directores.

Tendremos 512 procesadores . 64 Gbytes de memoria central . 2Mbytes de cache por procesador y bloques de memoria caché de 256 bytes.

a) Cuanta memoria extra se necesita en las cachés ?

512 procesadores

$$\begin{array}{lcl} 2 \text{Mbytes} \text{ cada cache} & \rightarrow & 2^1 \cdot 2^{20} \cdot 2^3 = \frac{2^{24}}{2^{13}} = 2^{13} = 8 \text{k bloques} \\ 256 \text{ bytes de bloque} & \rightarrow & 2^8 \cdot 2^3 = 2^{11} \end{array}$$

$$2 \cdot 8k \cdot 512 = 2^1 \cdot 2^3 \cdot 2^{10} \cdot 2^9 = 2^{23} = 1 \text{Mbyte}$$

b) Y para el directorio?

$$p+1 = 512 + 1 = 513$$

$$\begin{array}{lcl} 64 \text{Gbytes} = 2^6 \cdot 2^{30} \cdot 2^3 = \frac{2^{39}}{2^{11}} = 2^{28} = 256 \text{M bloques} \\ 256 \text{ bytes} = 2^8 \cdot 2^3 = 2^{11} \end{array}$$

$$256 \text{M} \cdot 513 \text{ bits} = 131328 \text{ Mbits}$$

• Se quiere diseñar un multiprocesador escalable de memoria compartida con memorias cachés privadas asociadas a cada procesador. La máquina utilizará un protocolo de coherencia de invalidación basado en directorios. El multiprocesador tendrá 2048 procesadores, 1024 GBytes de memoria central, 4MBbytes de memoria caché por procesador y bloques de memoria caché de 512 bytes. Se pide:

a) Memoria extra en bits que se requiere para mantener la información en las cachés.

b) Memoria extra en bits para mantener la info del directorio.

$$a) \frac{4\text{MBbytes de caché}}{512\text{ bytes}} = \frac{2^2 \cdot 2^{30} \cdot 2^3}{2^9} = \frac{2^{25}}{2^9} = 2^{16} = 64\text{K bloques/caché}$$

$$2 \cdot 2048 \cdot 64K = 2^1 \cdot 2^{11} \cdot 2^6 \cdot 2^{10} = 2^{28} = 256\text{M bits}$$

$$b) \frac{1024\text{ GBytes de mem principal}}{512\text{ bytes}} = \frac{2^{10} \cdot 2^{30} \cdot 2^3}{2^9} = \frac{2^{43}}{2^9} = 2^{34} = 16\text{G bloques principal}$$

$$p+1 = 2048+1 \quad 16G \cdot 2049 = 32784\text{G bits}$$

A vuelta con lo de antes, ahora con protocolo MSI. (3 estados)

Estado Inicial

Caché1	Caché2	Caché3	Caché4	Memoria Principal
@3 35 M	@1 25 M	@2 6 I	@1 7 I	@1 5 @3 15
@4 20 S	@4 20 S	@4 20 I	@4 20 S	@2 10 @4 20

[rd1(@1)]

Fallo de lectura. Se difunde una petición de lectura. Pasarían a S los @1 y se actualiza en memoria

@3 35 M	@1 25 S	@2 6 I	@1 7 I	@1 25 @3 15
@1 25 S	@4 20 S	@4 20 I	@4 20 S	@2 10 @4 20

wrl(@3,5)

Ajusto de escritura.

@3 5 M	@1 25 S	@2 6 I	@1 7 I	@1 25 @3 15
@1 25 S	@4 20 S	@4 20 I	@4 20 S	@2 10 @4 20

[wrt(@2,3)]

Fallo de escritura. Petición de Lectura Exclusiva

@3 5 M	@1 25 S	@2 6 I	@1 7 I	@1 25 @3 15
@2 3 M	@4 20 S	@4 20 I	@4 20 S	@2 10 @4 20

[rd2(@2)]

Fallo de lectura. Petición de lectura

@3 5 M	@2 3 S	@2 6 I	@1 7 I	@1 25 @3 15
@2 3 S	@4 20 S	@4 20 I	@4 20 S	@2 10 @4 20

rd2(@3)

Fallo de lectura. Petición de lectura

@3 5 S	@2 3 S	@2 6 I	@1 7 I	@1 25 @3 5
@2 3 S	@3 5 S	@4 20 I	@4 20 S	@2 3 @4 10

• Se quiere diseñar un multiprocesador escalable de memoria compartida con memorias cachés privadas asociadas a cada procesador. La máquina usará un protocolo de coherencia de invalidación basado en directorios. El multiprocesador tendrá 512 procesadores, 2048 GBytes de memoria central, 32 Mbits de memoria caché por procesador y bloques de memoria caché de 2048 bits. Se pide:

- Memoria extra (en bits) que se requiere para mantener la información en las caches.
- Memoria extra (en bits) que se requiere para mantener la información del directorio siendo este de mapeado

a) 2 bits por bloque $\frac{32 \text{ Mbits de mem cache}}{2048 \text{ bits}} = \frac{2^5 \cdot 2^{20}}{2^{11}} = \frac{2^{25}}{2^{11}} = 2^4 = 16 \text{ Kbloques}$

$$2 \cdot 16k \cdot 512 = 16384 \text{ Kbits}$$

b) 2048 GBytes $\rightarrow \frac{2^{11} \cdot 2^{30} \cdot 2^3}{2^{11}} = \frac{2^{44}}{2^{11}} = 2^{33} = 8 \text{ Gbloques}$

$$p+1 = 513 \quad 8G \cdot 513 = 4104 \text{ Gbits}$$

Más ejercicios de exámenes

1. Se dispone de una arquitectura segmentada en 3 etapas que es capaz de realizar la función A en cuatro ciclos. Por otro lado se tiene otra arquitectura también segmentada en 3 etapas que es capaz de realizar la función B en 5 ciclos.

Las correspondientes tablas de latencias son

	t_0	t_1	t_2	t_3
S0	A			A
S1		A		
S2		A		

	t_0	t_1	t_2	t_3	t_4	B
S0	B					B
S1		B	B			
S2				B		

Se ha reunido un equipo de ingenieros con el encargo de obtener una arquitectura que realice las dos funciones. Han reducido las opciones a dos

1º Unir las etapas sin mezclarlas creando la operación C.

2º Crea multifuncional.

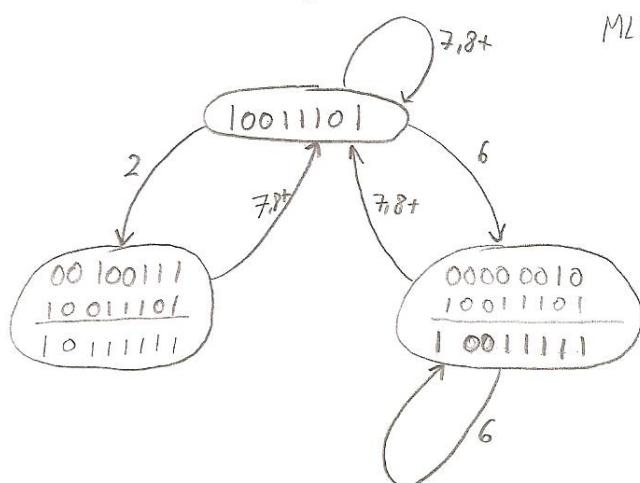
¿Cuál sería la mejor opción?

	S0	A		A	B		B
S1		A			B	B	
S2		A				B	

$$C = (10011101)$$

$$F = \{3, 4, 8, 1, 5\}$$

$$MLM_r = \frac{9}{2} = 45$$



Cause multifuncional

S_0	AB			A	B
S_1		AB	B		
S_2			A	B	

$$F_{AA} = \{3\}$$

$$F_{AB} = \{1, 4\}$$

$$F_{BA} = \{3\}$$

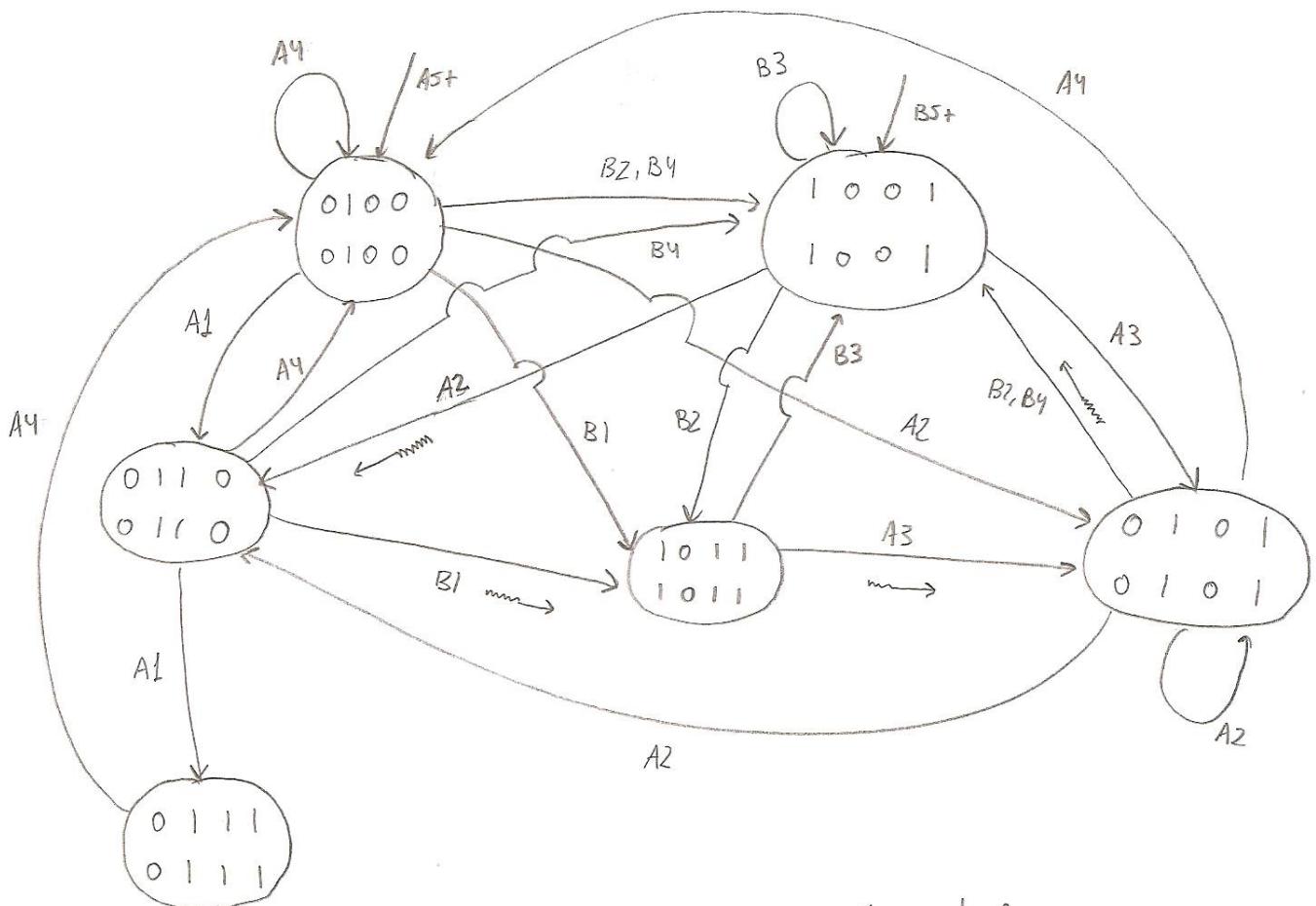
$$F_{BB} = \{2, 4\}$$

||

||

$$M_A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{array}{l} F_{AA} \\ F_{BA} \end{array}$$

$$M_B = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} F_{AB} \\ F_{BB} \end{array}$$



Secuencia: A B A B A B

Claramente la segunda opción es mejor que la primera

$$2 + 1 + 3 + 2 = \frac{8}{4} = 2$$

Eso es todo ;)

Ningún ejercicio está corregido, salvo algunos que están hechos en clase por lo tanto la solución será correcta.

Es bastante probable que algunos de los ejercicios contengan errores debido a que los he ido haciendo conforme estudiaba por lo tanto no los hacía todavía con mucha seguridad.

No obstante, y a pesar de los fallos (que si llegais a usar esto, y los encontrais, agradecería que los corrigieseis indicándolo en los comentarios del blog), espero que os ayude a aprobar esta asignatura en Julio, Diciembre, o ya, el año que viene.

Un saludo. Y suerte.