Profesores: Néstor González – Leo Medina
Fecha: 27 de abril de 2018
Duración: 90 minutos

Pregunta 1

Considere el siguiente código MIPS. Considere que el bucle se ejecuta muchas veces antes de salir. Para determinar el rendimiento bajo esta condición basta con calcular el rendimiento en estado permanente (o estado estable), lo que significa que la influencia de las primeras y de las últimas iteraciones se puede despreciar. Asuma que se dispone de *forwarding* completo y que la predicción de saltos es perfecta.

Código MIPS:

```
I1:    Label:    addi   r4, r4, 4
I2:              lw     r3, 0(r4)
I3:              lw     r2, 4(r4)
I4:              add    r2, r2, r3
I5:              lw     r3, 8(r4)
I6:              add    r2, r2, r3
I7:              sw     r2, –4(r4)
I8:              slt    r1, r4, zero
I9:              bne    r1, zero, Label
```

A) El código se ejecuta en un *pipeline* de 5 etapas. Determine los CPI para las condiciones dadas. (0,7pts)

En estado permanente, el pipeline completa una instrucción por ciclo de reloj, pero un lw seguido de una instrucción de tipo R genera una espera inevitable. Hay dos de estas dependencias en el código: I4(I3) y I6(I5). Como no hay otras esperas, entonces CPI= (9 + 2)/9 = 1.22

B) ¿Se puede reordenar el código para mejorar el rendimiento? Si su respuesta es "No", explique detalladamente porqué. En caso de responder "Sí", entonces proponga un reordenamiento para este código y determine ahora los CPI. Calcule también la aceleración lograda. (0,6pts)

Sí se puede reordenar. La dependencia I4(I3) se puede resolver si se ubica I8 entre medio. Se ahorra una espera. La dependencia I6(I5) no se puede evitar. Luego CPI=(9+1)/9 = 1.11. Aceleración = 9%.

C) Suponga ahora que no se dispone de *forwarding* de ningún tipo. Para el código obtenido en B), determine los CPI. ¿En cuánto se ve afectado el rendimiento? (0,7pts)

código obtenido en b.

```
I1:    Label: addi   r4,r4,4
I2:           lw     r3,0(r4)
I3:           lw     r2,4(r4)
I8:           slt    r1,r4,zero
I4:           add    r2,r2,r3
I5:           lw     r3,8(r4)
I6:           add    r2,r2,r3
I7:           sw     r2,–4(r4)
I9:           bne    r1,zero,Label
```

Las dependencias sin forwarding son:

I2(I1) lw requiere r4 espera addi 1 ciclo
I3(I1) lw requiere r4 espera addi 1 ciclo

I4(I3) add requiere r2 (pero I8 resuelve la espera) 0 ciclo
I6(I5) add requiere r3 espera lw 1 ciclo
I7(I6) sw requiere r2 espera add 2 ciclos

En total se agregan 5 ciclos, 4 debidos a no disponer de forwarding, 1 por el lw en I5.

CPI = (9 + 5)/9 = 1,55

El rendimiento se afecta en un 44%

Pregunta 2

Considere el procesador MIPS monociclo de la **Error! Reference source not found.**. El conjunto de instrucciones (ISA) de este procesador es modificado para agregar una nueva instrucción tipo-I:

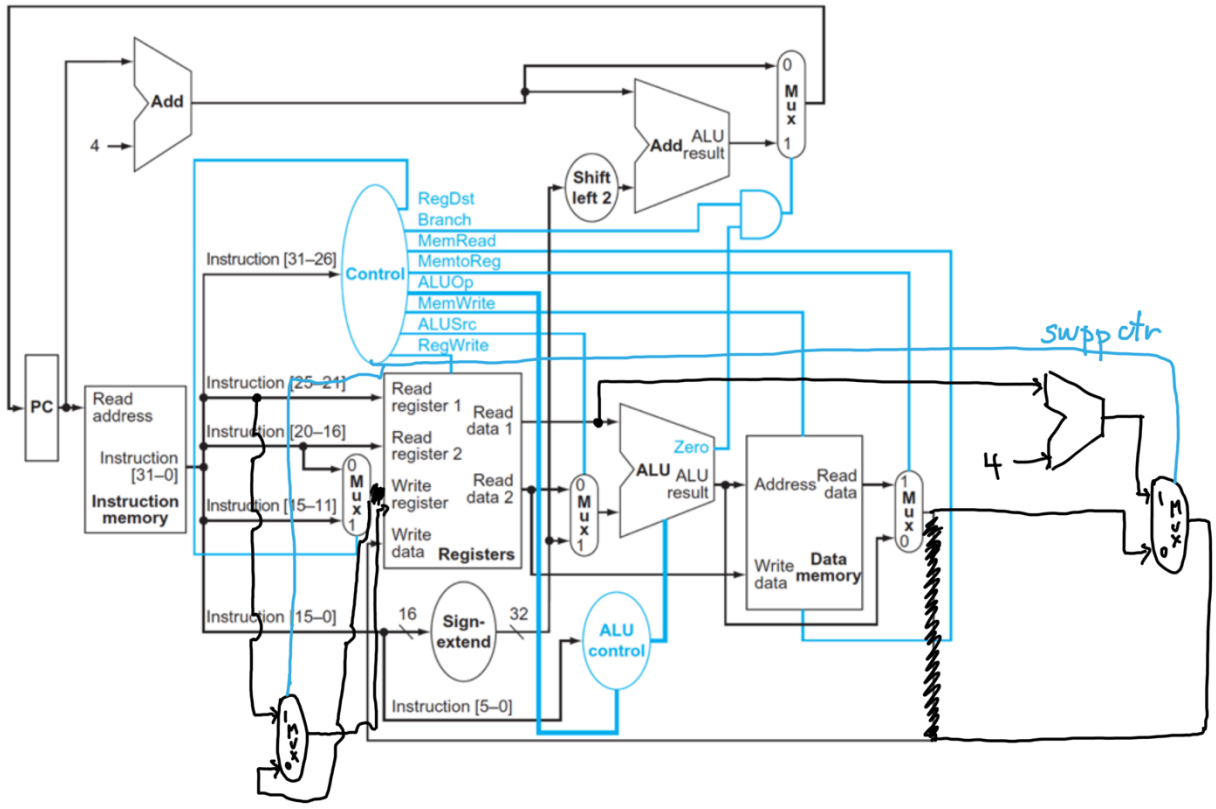
```
swpp $rt, imm($rs)
```

donde `$rt` es el registro que contiene el valor a almacenar, `$rs` el registro de dirección base y `imm` es un valor *offset* de 16 bits. La instrucción almacena en memoria el contenido de `$rt` de igual manera como lo hace `sw`, pero además incrementa el registro de dirección `$rs` de manera de apuntar de manera inmediata a la siguiente palabra en memoria. Esta instrucción es útil si necesitamos almacenar varios elementos en un arreglo de manera rápida. Notar que la instrucción `swpp` es equivalente a dos instrucciones MIPS:

```
sw   $rt, imm($rs)
addi $rs, $rs, 4
```

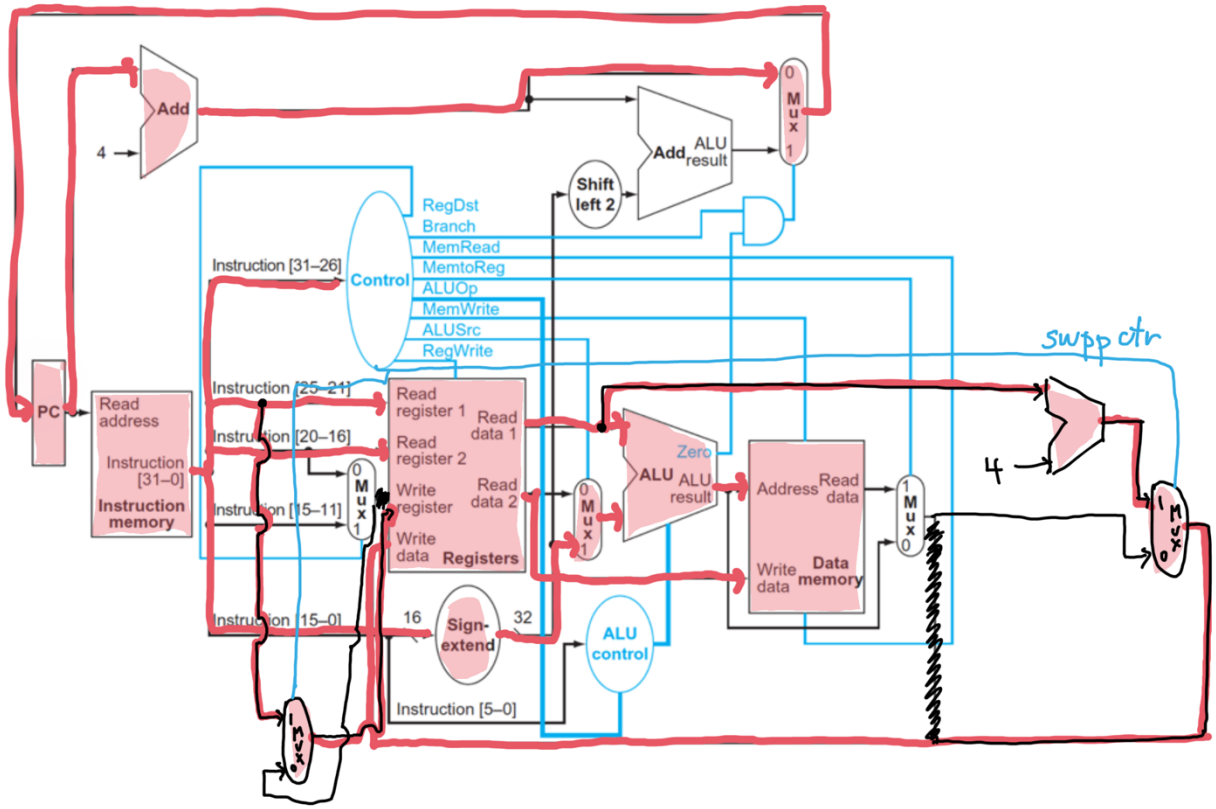
A) Muestre los mínimos cambios, adiciones o eliminaciones de componentes de hardware (*e.g.* multiplexores) e interconexiones que se requieren hacer al camino de datos MIPS monociclo para permitir esta modificación. Incluya las líneas de control que pueda requerir. (0,5pts)

Se requiere una unidad aritmética para sumar 4 a $rs, y luego escribir este resultado en el archivo de registros. Dado que hay otras instrucciones del ISA de MIPS que escriben en el archivo de registros, debemos agregar un multiplexor que determine qué dato se escribirá, i.e., seleccionar entre salida del multiplexor MemtoReg y línea que estamos incorporando para la suma. Además, la dirección del registro a escribir en el archivo de registros, debe poder seleccionarse (multiplexor) entre la línea para otras instrucciones y $rs, que es donde pondremos el resultado de la suma $rs + 4. Finalmente, los dos muliplexores añadidos deben controlarse con una nueva línea de control "swppctr".



B) Marque claramente en su diagrama de la parte A) el camino de datos para `swpp`. (0,5pts)

En el siguiente diagrama, el camino de datos para `swpp` se ha marcado en rojo.



C) Defina los valores de todas las líneas de control para la correcta ejecución de la instrucción `swpp`. Incluya el(los) valor(es) de la(s) línea(s) que haya agregado. Si el valor de una línea es indiferente para esta instrucción, márquelo con una X ("don't care"). (0,5pts)

Similar a `sw`, las líneas de control deben tomar los siguientes valores:

RegDst	X
ALUSrc	1
MemtoReg	X
RegWrite	1
MemRead	0
MemWrite	1
Branch	0
ALUOp	00 (notar que son dos bits)

Además, agregamos una nueva línea de control que debe definirse en 1:

Swppctr	1

D) Suponga que los bloques lógicos del procesador monociclo tienen las siguientes latencias en ps:

I-MEM	Add	Mux	ALU	Regs	D-MEM	Sign-Extend	Shift-Left-2	ALU Ctr
200	70	20	90	90	250	15	10	30

Determine las latencias de las instrucciones `sw`, `addi` y `swpp`. Considerando solo estas 3 instrucciones, ¿cuál sería la tasa de reloj máxima de este procesador? (0,5pts)

`sw`: I-MEM+Regs+Mux+ALU+D-MEM+Mux= 650 ps

`addi`: I-MEM+Mux+Regs+Mux+ALU+Mux = 440 ps

`swpp`: I-MEM+Mux+Regs+Mux+ALU+Add+Mux+D-MEM= 740 ps

Si consideramos solo estas 3 instrucciones, la latencia crítica está dada por los 740 ps de swpp. Entonces la tasa de reloj máxima sería 1000/740 GHz

Pregunta 3

Parte 3.1

Considere el siguiente código de instrucciones en MIPS. Los valores iniciales de los registros son `$t0=6`, `$sp=0x1313`, `$t1=1` y `$t3=9`.

```
INICIO:    bne  $t0, $zero, CODE
           j FINAL
CODE:      subi $t0, $t0, 2
           sub  $t0, $t0, $t1
           addi $t3, $t3, -8
           sw   $t3, 0($sp)
           lw   $t1, 0($sp)
           j INICIO
FINAL:     addi $t1, $t0, 1
```

A) Encuentre todas las dependencias de datos en el código. (0,4pts)

B) Dibuje como se ejecuta el código por un *pipeline* de 5 etapas (IF, ID, EX, MEM, WB) que no posee *forwarding* y utiliza *branch* "NOT TAKEN" (es decir, se predice que el *branch* o bifurcación no va a saltar). (0,4pts)

C) Agregue los NOP necesarios a esta secuencia de instrucciones para resolver los posibles riesgos de datos. (0,3pts)

D) Marque en su dibujo del *pipeline* en B) todos los *hazards* que ocurren en la ejecución. (0,3pts)

ciclo	if	id	ex	mem	wb
1	bne $t0, $zero, CODE				
2	j FINAL	bne $t0, $zero, CODE			
3	subi $t0, $t0, 2	j FINAL	bne $t0, $zero, CODE		
4	addi $t1, $t0, 1	NOP	j FINAL	bne $t0, $zero, CODE	
5	subi $t0, $t0, 2	NOP	NOP	NOP	bne $t0, $zero, CODE
6	sub $t0, $t0, $t1	subi $t0, $t0, 2	NOP	NOP	NOP
7	addi $t3, $t3, -8	sub $t0, $t0, $t1	subi $t0, $t0, 2	NOP	NOP
8	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	subi $t0, $t0, 2	NOP
9	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	NOP	subi $t0, $t0, 2
10	sw $t3, 0($sp)	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	NOP
11	lw $t1, 0($sp)	sw $t3, 0($sp)	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP
12	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	addi $t3, $t3, -8	sub $t0, $t0, $t1
13	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	NOP	addi $t3, $t3, -8
14	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	NOP
15	addi $t1, $t0, 1	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP
16	bne $t0, $zero, CODE	NOP	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)
17	j FINAL	bne $t0, $zero, CODE	NOP	j INICIO	lw $t1, 0($sp)
18	subi $t0, $t0, 2	j FINAL	bne $t0, $zero, CODE	NOP	j INICIO
19	addi $t1, $t0, 1	NOP	j FINAL	bne $t0, $zero, CODE	NOP
20	subi $t0, $t0, 2	NOP	NOP	NOP	bne $t0, $zero, CODE
21	sub $t0, $t0, $t1	subi $t0, $t0, 2	NOP	NOP	NOP
22	addi $t3, $t3, -8	sub $t0, $t0, $t1	subi $t0, $t0, 2	NOP	NOP
23	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	subi $t0, $t0, 2	NOP
24	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	NOP	subi $t0, $t0, 2
25	sw $t3, 0($sp)	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP	NOP
26	lw $t1, 0($sp)	sw $t3, 0($sp)	addi $t3, $t3, -8	sub $t0, $t0, $t1	NOP
27	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	addi $t3, $t3, -8	sub $t0, $t0, $t1
28	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	NOP	addi $t3, $t3, -8
29	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP	NOP
30	addi $t1, $t0, 1	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)	NOP
31	bne $t0, $zero, CODE	NOP	j INICIO	lw $t1, 0($sp)	sw $t3, 0($sp)
32	j FINAL	bne $t0, $zero, CODE	NOP	j INICIO	lw $t1, 0($sp)
33	subi $t0, $t0, 2	j FINAL	bne $t0, $zero, CODE	NOP	j INICIO
34	addi $t1, $t0, 1	NOP	j FINAL	bne $t0, $zero, CODE	NOP
35		addi $t1, $t0, 1	NOP	j FINAL	bne $t0, $zero, CODE
36			addi $t1, $t0, 1	NOP	j FINAL
37				addi $t1, $t0, 1	NOP
38					addi $t1, $t0, 1

Parte 3.2

Se tienen 3 procesadores *pipeline*, uno de 5 etapas, uno de 6 etapas y uno de 7 etapas. Dado el siguiente código y las latencias de las etapas de cada procesador entregadas en ps, responda lo siguiente.

```
lw   $t1, 0($sp)
add  $t0, $t0, $t2
addi $t2, $zero, 3
sub  $t3, $t2, $t1
sw   $t2, 0($sp)
addi $s0, $t2, 1
addi $t0, $t1, 3
lw $t0, 4($sp)
```

Tabla 1: Latencias del procesador de 5 etapas

IF	ID	EX	MEM	WB
40	60	150	180	90

Tabla 2: Latencias del procesador de 6 etapas

IF	ID	EX	MEM-1	MEM-2	WB
55	60	150	90	100	95

Tabla 3: Latencias del procesador de 7 etapas

IF	ID	EX-1	EX-2	MEM-1	MEM-2	WB
45	40	60	80	100	90	100

A) ¿Cuál procesador ejecutaría más rápido el código? (0,2pts)

B) ¿Cuál procesador se demoraría más en ejecutarlo? (0,2pts)

C) Si el procesador de 5 etapas fuera monociclo, ¿cuánto demoraría la ejecución del código? (0,2pts)

Ciclo	IF	ID	EX	MEM	WB
1	LW $t1,0($sp)	-	-	-	-
2	add $t0,$t0,$t2	LW $t1,0($sp)	-	-	-
3	addi $t2, $zero, 3	add $t0,$t0,$t2	LW $t1,0($sp)	-	-
4	addi $t2, $zero, 3	addi $t2, $zero, 3	add $t0,$t0,$t2	LW $t1,0($sp)	-
5	addi $t2, $zero, 3	addi $t2, $zero, 3	NOP	add $t0,$t0,$t2	LW $t1,0($sp)
6	sub $t3, $t2,$t1	addi $t2, $zero, 3	NOP	NOP	add $t0,$t0,$t2
7	sw $t2,0($sp)	sub $t3, $t2,$t1	addi $t2, $zero, 3	NOP	NOP
8	addi $t0,$t2,1	sw $t2,0($sp)	sub $t3, $t2,$t1	addi $t2, $zero, 3	NOP
9	addi $t0,$t1,3	addi $t0,$t2,1	sw $t2,0($sp)	sub $t3, $t2,$t1	addi $t2, $zero, 3
10	Lw $t0, 4($sp)	addi $t0,$t1,3	addi $t0,$t2,1	sw $t2,0($sp)	sub $t3, $t2,$t1
11	-	Lw $t0, 4($sp)	addi $t0,$t1,3	addi $t0,$t2,1	sw $t2,0($sp)
12	-	-	Lw $t0, 4($sp)	addi $t0,$t1,3	addi $t0,$t2,1
13	-	-	-	Lw $t0, 4($sp)	addi $t0,$t1,3
14	-	-	-	-	Lw $t0, 4($sp)

	Nro ciclos	Tiempo ciclo	Demora
5 fases	14	180	2520
6 fases	16	150	2400
7 fases	18	100	1800

Vemos que:

A) El que menos demora es el procesador de 7 fases

B) El que mas demora es el procesador de 5 fases

C) Si fuera monociclo, demoraría 520 ns en un ciclo, y 4160 ns en ejecutar el código