



Pregunta 1. Rendimiento (PEP 1 – 2019-01)

Parte 1

Se tiene un procesador con una tasa de reloj 2 GHz. Un programa P contiene un total de 400 mil millones de instrucciones que en este procesador se ejecutan en 13 min y 20 s (tiempo de CPU despreciando tiempos de acceso a memoria, buses y otros).

- A) (0,2 pts.) Calcule el CPI para el programa P .

$$\begin{aligned} \text{CPI} &= \# \text{ciclos} / \# \text{instrucciones} = \text{tiempo CPU} * \text{tasa reloj} / \# \text{instrucciones} \\ &= 800 \text{ s} * 2\text{E}9 \text{ s}^{-1} / 400\text{E}9 = 4 \end{aligned}$$

- B) (0,2 pts.) Calcule el tiempo de CPU para un programa Q que tiene 20% más instrucciones que P , y para el que se obtuvo igual CPI.

$$\text{tiempo CPU} = \text{CPI} * \# \text{instrucciones} * \text{tiempo ciclo reloj} = 4 * 1,2 * 400\text{E}9 * (1/2) \text{ s} = 960 \text{ s} = 16 \text{ min}$$

- C) (0,3 pts.) Si otro procesador obtiene un CPI de 3,5 para P , ¿qué procesador es más rápido para este programa y por cuánto?

$$\text{tiempo_CPU2} / \text{tiempo_CPU1} = \text{CPI}_1 / \text{CPI}_2 = 4 / 3,5 = 1,14. \text{ Es decir, el segundo procesador es } 1,14 \text{ veces más rápido. Notar que se ha asumido que implementan el mismo ISA y corren a igual tasa de reloj.}$$

- A) (0,3 pts.) Considere un nuevo procesador que implementa un conjunto de instrucciones distinto y que tiene una frecuencia de reloj de 3,2 GHz. En este nuevo procesador, P compila en muchas menos instrucciones de lenguaje ensamblador, y el tiempo de ejecución baja a 10 min. Con estos datos, ¿se puede calcular el número de instrucciones de P para este procesador? Calcule el CPI en estas condiciones (si requiere el número de instrucciones, “N”, deje expresado el CPI en términos de “N”).

$$\text{CPI} = \# \text{ciclos} / \# \text{instrucciones}. \text{ Con tiempo y tasa de reloj se puede calcular } \# \text{ciclos}, \text{ pero faltan datos para determinar el número de instrucciones; no se indica qué características tiene el nuevo ISA. Entonces CPI} = 600 \text{ s} * 3,2\text{E}9 \text{ s}^{-1} / N$$

Parte 2

- A) (1 pt.) Considere el siguiente programa escrito en lenguaje C. Considere que todas las instrucciones toman, en promedio, 1,5 ciclos en ejecutarse, excepto las de acceso a memoria que demoran, en promedio, 5 ciclos. Calcule el CPI para este programa.

```
int i;
for (i=0; i<1000000; i=i+1)
    A[i] = B[i] + 2
```

Notar que el programa por cada iteración requiere al menos: 1 beq, 1 addi y 1 jump para actualización de índice y manejo del bucle, 1 lw para cargar B[i], 1 add para la suma y 1 sw para guardar resultado en A[i]. O sea, por cada iteración hay 2 instrucciones de acceso a memoria y 4 de otro tipo. Entonces
 $\text{CPI} = (1,5*4 + 5*2)/6 = 16/6$. (El número total de iteraciones se factoriza y cancela de numerador y denominador).



Pregunta 2. Rendimiento (PEP 1 – 2018-02)

La empresa de procesadores LETNI ha diseñado un nuevo procesador, el LETNI 8080, que promete revolucionar el mercado. Actualmente, si bien la empresa tiene diseñadores prolíficos de hardware, estos desconocen las nociones básicas del rendimiento de los procesadores y no saben cómo obtener sus especificaciones técnicas. En busca de alguien capacitado para ayudarles, se dirigen a la Universidad de Santiago de Chile, pues en esta institución los alumnos aprenden todo lo que LETNI necesita acerca del rendimiento de procesadores. Por recomendación del profesor, usted ha sido seleccionado para obtener las especificaciones técnicas de rendimiento del procesador.

El LETNI 8080 tiene un tiempo de ciclo de reloj de $2/3$ ns. Un programa P contiene un total de 300 mil millones de instrucciones que en el LETNI 8080 se ejecutan en 16 min y 40 s (tiempo de CPU despreciando tiempos de acceso a memoria, buses y otros).

- A) (0,3 pts) Calcule el CPI del LETNI 8080 para el programa P .

$$\begin{aligned} \text{CPI} &= \# \text{ciclos} / \# \text{instrucciones} = \text{tiempo CPU} / \text{tiempo ciclo reloj} / \# \text{instrucciones} \\ &= 1000 \text{ s} / 2/3\text{E-9 s} / 3\text{E}11 = 5 \end{aligned}$$

- B) (0,3 pts) Calcule el tiempo de CPU para un programa Q que tiene 18% más instrucciones que P , y para el que se obtuvo igual CPI.

$$\text{tiempo CPU} = \text{CPI} \times \# \text{instrucciones} \times \text{tiempo ciclo reloj} = 5 \times 118/100 \times 3\text{E}11 \times 2/3\text{E-9 s} = 1180 \text{ s}$$

- C) (0,3 pts) La competencia del LETNI 8080 es el DMA Nolhta, que implementa el mismo conjunto de instrucciones (ISA). El DMA Nolhta corre a la misma tasa de reloj que el LETNI 8080, y entrega un CPI de 6 para P . ¿Qué procesador es más rápido para este programa y por cuánto?

$$\begin{aligned} \text{tiempo_LETNI} / \text{tiempo_DMA} &= \text{CPI_LETNI} \times \# \text{inst} \times \text{tiempo reloj_LETNI} / \text{CPI_DMA} \times \# \text{inst} \times \text{tiempo reloj_DMA} \\ &= \text{CPI_LETNI} / \text{CPI_DMA} = 5 / 6 < 1 \end{aligned}$$

Es decir, el LETNI es $6/5$ veces más rápido que el DMA.

- D) (0,3 pts) Considere el procesador LETNI 8080X, idéntico al 8080 (*i.e.*, mismo ISA), pero con una frecuencia de reloj de 1,7 GHz. Se corre un programa R que tiene un 15% menos de instrucciones que P , y cuyo tiempo de ejecución en el LETNI 8080X es 11 min. Calcule el CPI del 8080X en estas condiciones.

$$\text{CPI} = 660 \text{ s} \times 17\text{E}8 \text{ Hz} / (85/100 \times 3\text{E}11) = 4,4$$

Debido a su excelente desempeño en el trabajo anterior, LETNI le solicita ayuda para comparar tres procesadores de su línea Quantium: el Quantium A, el Quantium B y el Quantium C. Estos tienen frecuencias de reloj de 3, 2,2 y 3,6 GHz, respectivamente.

- E) (0,3 pts) Para un programa T , los Quantium A, B y C arrojaron un CPI de 1,2, 1 y 1,8, respectivamente. En estas condiciones, ¿qué procesador tiene el mejor rendimiento en millones de instrucciones por segundo?

$$\text{MIPS} = \# \text{instrucciones} / (\text{tiempoCPU} \times 1\text{E}6) = \text{tasa reloj} / (\text{CPI} \times 1\text{E}6)$$

$$\text{MIPS_A} = 3\text{E}9 / 1,2\text{E}6 = 2500$$

$$\text{MIPS_B} = 2,2\text{E}9 / 1\text{E}6 = 2200$$

$$\text{MIPS_C} = 3,6\text{E}9 / 1,8\text{E}6 = 2000$$

¿Es el Quantium A mejor?

- F) (0,3 pts) T ejecuta un total de 18 mil millones de instrucciones en el Quantium A. Sin embargo, para los Quantium B y C se utilizaron distintas extensiones multimedia del ISA de modo que T



ejecuta 13 mil doscientas millones de instrucciones en el Quantum B y 14 mil millones de instrucciones en el Quantum C. Calcule los tiempos de CPU para T en cada procesador.

$$\text{tiempoCPU} = \text{CPI} \times \# \text{instrucciones} / \text{tasa reloj}$$

$$\text{tiempoCPU_A} = 1,2 \times 18\text{E}9 / 3\text{E}9 = 7,2 \text{ s}$$

$$\text{tiempoCPU_B} = 1,0 \times 13,2\text{E}9 / 2,2\text{E}9 = 6 \text{ s}$$

$$\text{tiempoCPU_C} = 1,8 \times 14\text{E}9 / 3,6\text{E}9 = 7 \text{ s}$$

G) (0,2 pts) ¿En qué orden recomendaría estos tres procesadores? ¿Por qué?

Dados los tiempos de CPU, recomendaría primero B, luego C y finalmente A. Notar que la utilización de extensiones multimedia permitió reducir considerablemente el número de instrucciones y con esto el tiempo total de ejecución.



Pregunta 1. Rendimiento (1,5 pts.)

Parte 1

Considere 3 procesadores distintos, P1, P2 y P3, que implementan el mismo conjunto de instrucciones (ISA). P1, P2 y P3 tienen tasas de reloj de 3 GHz, 2,5 GHz y 4 GHz, respectivamente, y CPIs de 1,5, 1,0 y 2,2, respectivamente.

- A) (0,3 pts.) ¿Qué procesador tiene el mejor rendimiento, medido en instrucciones por segundo?
- B) (0,3 pts.) Si cada procesador ejecuta un programa (no necesariamente el mismo) en 10 segundos, encuentre el número de instrucciones y el número de ciclos.
- C) (0,3 pts.) Para cada procesador, encuentre la tasa de reloj que permite reducir el tiempo de ejecución en un 30% a expensas de un incremento de 20% del CPI.

Parte 2

Considere el siguiente programa que muestra un bucle que se repite infinitamente. Se pide comparar el desempeño de un procesador MIPS monociclo con tasa de reloj de 750 MHz y un procesador MIPS con *pipeline* de 5 etapas, como el discutido en clases. Asuma que el procesador con *pipeline* dispone de adelantamiento (*forwarding*) y que la dirección de destino de un salto incondicional (j) está disponible en la etapa IF.

```
ilovebeatles:    lw $t0, 0($s0)
                  add $t1, $t0, $t2
                  add $t1, $t0, $t2
                  add $t1, $t0, $t2
                  j ilovebeatles
```

- D) (0,3 pts.) Determine el CPI para ambos procesadores.
- E) (0,3 pts.) Determine la tasa de reloj del procesador con *pipeline* de modo que el tiempo promedio de ejecución de una iteración del bucle sea el mismo para ambos procesadores.

P1

$$A) CPI = \frac{\# \text{ciclos}}{\# \text{instr.}} = \frac{\text{tiempo CPU} \times \text{tasa reloj}}{\# \text{instr.}} \rightarrow \frac{\# \text{instr.}}{\text{tiempo CPU}} = \frac{\text{tasa reloj}}{CPI}$$

$\frac{\# \text{ciclos}}{\# \text{instr.}} = 190 \leftarrow$ dato y el sistema opera mejor

$$IPS_{P_1} = 2 \times 10^9 \text{ instr/s}$$

$$IPS_{P_2} = 2,5 \times 10^9 \text{ instr/s}$$

$$IPS_{P_3} = 1,82 \times 10^9 \text{ instr/s}$$

P_2 tiene mejor rendimiento

$$B) \# \text{instr.} = \frac{\text{tiempo CPU} \times \text{tasa reloj}}{CPI} = I \quad \# \text{ciclos} = CPI \times \# \text{instr.} = C$$

Notar que se está asumiendo mismo CPI de arriba, pero esto podría no ser cierto, dependiendo del programa

$$T_{P_1} = 20 \times 10^9 \quad CPI_{P_1} = 30 \times 10^9$$

$$T_{P_2} = 25 \times 10^9 \quad CPI_{P_2} = 25 \times 10^9$$

$$T_{P_3} = 18,2 \times 10^9 \quad CPI_{P_3} = 40 \times 10^9$$

$$C) \frac{\text{tiempo CPU nuevo}}{\text{tiempo CPU viejo}} = 0,7 = \frac{\frac{\# \text{instr.} \times CPI_{\text{nuevo}}}{\text{tasa reloj nueva}}}{\frac{\# \text{instr.} \times CPI_{\text{vieja}}}{\text{tasa reloj vieja}}} = \frac{\text{tasa reloj nueva}}{\text{tasa reloj vieja}} \times 1,2$$

$$\text{tasa reloj nueva} = \text{tasa reloj vieja} \times \frac{1,2}{0,7}$$

→ Para P_1 : 5,14 GHz

→ Para P_2 : 4,29 GHz

→ Para P_3 : 6,86 GHz

D) monociclo: $CPI = 1$

Pipeline: 1 burbuja entre lw, add $\rightarrow CPI = \frac{6}{5}$

$$E) t_{\text{mono}} = \frac{5}{\text{tasa reloj mono}} = t_{\text{pipeline}} = \frac{6}{\text{tasa reloj pipeline}}$$

$$\rightarrow \text{tasa reloj pipeline} = \frac{6}{5} \times 750 \text{ MHz} = 900 \text{ MHz}$$



Pregunta 2. Procesador Monociclo (2,5 pts.)

Considere el procesador MIPS monociclo visto en clases (Figura 1). Suponga que en determinado ciclo de reloj se está ejecutando una instrucción, In , que escrita en binario es: '00100001100011011000000000010100'. In está almacenada en la dirección 0x000011B4 de la memoria de instrucciones. Asuma que los registros del procesador tienen los siguientes valores, escritos en decimal, al inicio del ciclo en que se realiza "fetch" de esta instrucción:

#Registro	8	9	10	11	12	13	14	15	16	17	18	19
Valor almacenado	11	-1	0	10	10	15	20	300	100	-21	22	21

- A) (0,4 pts.) ¿Cuáles son los valores de *todas* las entradas del archivo de registros?
- B) (0,3 pts.) ¿Cuáles son los valores de *todas* las salidas del archivo de registros?
- C) (0,3 pts.) ¿Cuál es el valor de la salida del "Add" de más a la derecha en la Figura 1?
- D) (0,2 pts.) ¿Cuál es el nuevo valor del contador de programa al terminar la ejecución de esta instrucción?
- E) (0,3 pts.) Para cada multiplexor, indique el valor de la señal de control que lo comanda. HINT: Note que el dibujo de cada multiplexor en la Figura 1 indica qué línea se selecciona dependiendo del valor de la señal de control.
- F) (0,4 pts.) Suponga que la fabricación de este procesador resultó fallida, de modo que el bit 2 (número de bit se cuenta de derecha a izquierda, partiendo de 0) de la entrada "Read Register 1" del archivo de registros es siempre cero, independiente de la instrucción en ejecución. Este error se denomina "*stuck-at-0*". ¿Cuál es el resultado de la ALU al ejecutar In en el procesador con este error de fabricación?
- G) (0,6 pts) Suponga que, para probar el procesador, se pueden llenar el PC, los registros y las memorias de datos e instrucciones con valores a elección, luego ejecutar una instrucción particular y finalmente leer el PC, registros y memorias, los que son examinados para determinar la presencia de una falla particular. ¿Cómo detectaría el error "*stuck-at-0*" de la parte F)?

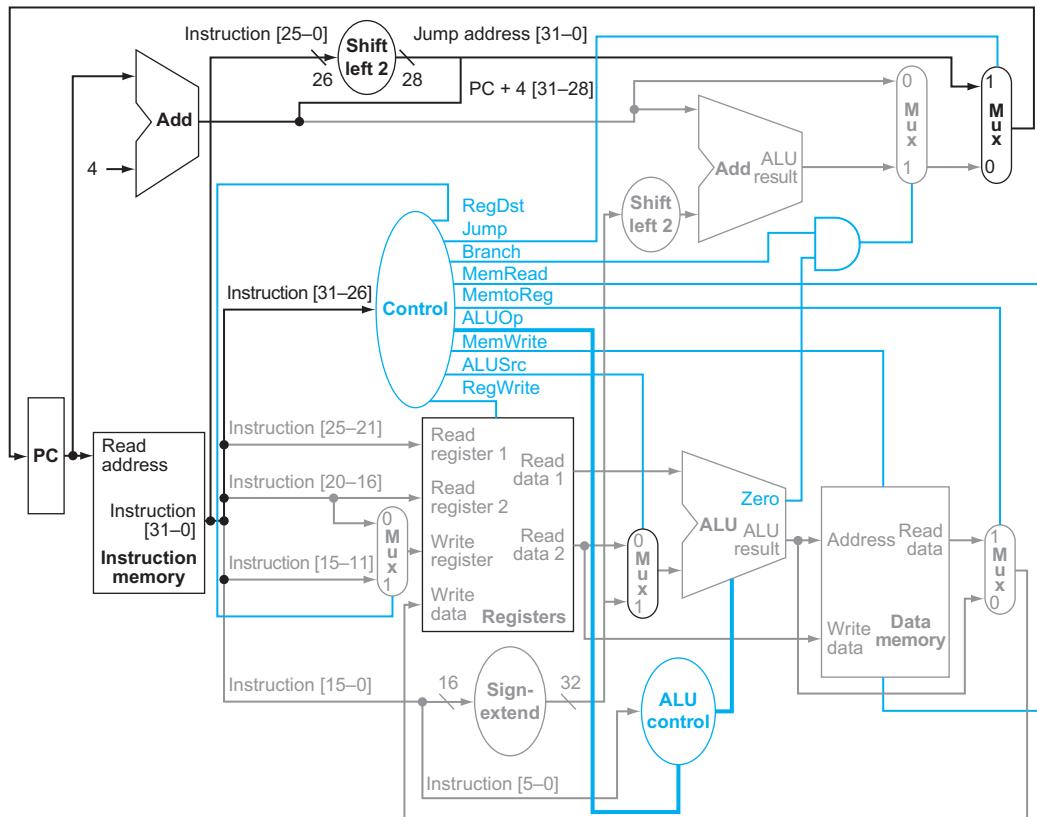


Figura 1. Diagrama de un procesador monociclo. Se ha marcado el camino de datos de la instrucción “jump”.

Tabla 1. Opcode de algunas instrucciones MIPS

Instrucción	Opcode (decimal)
slt	0
add	0
sub	0
j	2
beq	4
bne	5
addi	8
lw	35
sw	43

P2 / Notar opcode 001000_{bin} = 8_{dec} → add i

A) Read Reg 1 : 0110 0_{bin} = 12

Read Reg 2 : 0110 1_{bin} = 13

Write Reg 1 : 0110 1_{bin} = 13

Write Data : $\underbrace{10}_{\text{contenido reg. 12}} - \underbrace{20}_{\text{"immediate"}} = -10$

Reg Write : 1

B) Read Data 1 : 10_{dec}

Read Data 2 : 15_{dec} (no usado)

C) $(PC + 4) + \text{"immediate"} \times 4 = 11B8_{hex} + -20_{dec} \times 4 = 4456_{dec}$
 $= 1168_{hex}$

D) PC + 4 = 11B8_{hex}

E) Reg Dst : 0

ALUSrc : 1

MemtoReg : 0

Jump : 0

Branch ANDZero 01:40

F) Se lee registro 8 en vez de 12 → Suma ALU : 11 + -20 = -9

G) Varias opciones. Se puede por ejemplo llenar \$4 con un 1_{dec}

) luego hacer add \$5, \$4, \$0 e. Si hay error \$5 guardará un 0 en vez de 1!

Pregunta 3. Pipeline (2 pts.)

Parte 1

Considere las siguientes secuencias de instrucciones, S1 y S2.

S1	S2
add \$t1, \$t2, \$t1	lw \$t1, 0(\$t1)
lw \$t2, 0(\$t1)	and \$t1, \$t1, \$t2
lw \$t1, 4(\$t1)	lw \$t2, 0(\$t1)
and \$t3, \$t1, \$t2	lw \$t1, 0(\$t3)
or \$t3, \$t3, \$t3	
sw \$t3, 0(\$t1)	
or \$t4, \$t3, \$t2	

- A) (0,4 pts.) Encuentre todos los riesgos de datos en ambas secuencias de instrucciones para un *pipeline* de cinco etapas como el visto en clases (*i.e.*, IF-ID-EX-MEM-WB). Indique cuáles riesgos se resuelven completamente con adelantamiento (*forwarding*) y cuáles no.
- B) (0,6 pts.) Para ambas secuencias, dibuje un diagrama de ciclos de reloj del *pipeline* (ciclos avanzan de izquierda a derecha). Considere el caso **sin** adelantamiento.
- C) (0,2 pts.) Para reducir el tiempo del ciclo de reloj, se considera la posibilidad de dividir la etapa MEM en dos. Repita A) para este *pipeline* de seis etapas (*i.e.*, IF-ID-EX-MEM1-MEM2-WB).
- D) (0,3 pts.) Asuma que, antes de ejecutar las secuencias de instrucciones S1 y S2, todos los valores en la memoria de datos valen 0 y los valores en los registros \$t0 a \$t3 para S1 y S2 son los siguientes, respectivamente:

	\$t0	\$t1	\$t2	\$t3
S1	0	1	31	1000
S2	0	-2	63	2500

¿Cuál es el primer valor que se adelanta y a qué valor reemplaza?

Parte 2

- E) (0,3 pts.) Considere un procesador con *pipeline* de N etapas en el que, para una bifurcación (beq), la comparación y el cálculo de la dirección de destino se realizan en la etapa i -ésima. Asuma un esquema de predicción “branch not taken” o “bifurcación no tomada”. Entregue una expresión del CPI para un programa de X instrucciones sin riesgos de datos ($X >> N$), con una proporción b de instrucciones de bifurcación, y en el que hay probabilidad p de que las bifurcaciones se tomen. Asuma que, al finalizar el programa, una fracción e del total de instrucciones se ejecutaron, y esta fracción incluye todas las instrucciones de bifurcación.
- F) (0,2 pts.) Considere el particular caso en que, al finalizar el programa, sólo instrucciones de bifurcación fueron ejecutadas y en todas ellas hubo un salto. En este caso, ¿en qué condiciones el CPI podría ser 2?

P3

A) Para S1		Para S2	
use resolve con forwarding?		use resolve on forwarding?	
I1 → I2	Si	I1 → I2	No
I1 → I3	Si	I1 → I3	Si
I2 → I4	Si	I2 → I3	Si
I3 → I4	No	I1 = 10110 : 1st str	
I4 → I5	Si	I2 = 01010 : 2nd str	
I5 → I6	Si	I3 = 01 ~ 01 : short str	
I5 → I7	Si	"forwarding" elimination SI. (P)	

B)

Para S1

add	IF ID EX M WB	IF ID EX M WB	(B)
lw → nop	IF ID B B B	(between or) and SI : S. short str	
lw → nop	ID B B B		
lw JZ P P	= P B B B	= P "division" + (P + 39)	(D)
lw	IF ID EX M WB		
and → nop	IF ID B B B		
and → nop	ID B B B	xed S811 = P + 29	(E)
and			
or → nop			
or → nop			
or			
sw → nop			
sw → nop			
or			

P - Para S2: UFA word ← SI do SW as & enter for set SI (F)

lw	IF ID EX M WB	IF ID EX M WB	(G)
and → nop	IF ID B B B	depends reg during R2 longword write	
and → nop	ID B B B	, P, Z & R2 need to reg	(H)
and	IF ID EX M WB	& on constraint Z &	
lw → nop	IF ID B B B		
lw → nop	ID B B B		
lw			
lw			

C) Para S1:
 mismos riesgos \rightarrow se agrega:
 $I_1 \rightarrow I_4$
 $I_3 \rightarrow I_6$
 $I_4 \rightarrow I_7$ } se resuelven con adelantamiento

Para S2:
 mismos riesgos

D) Para S1:
 se adelanta $1 + 3 = 32$, \rightarrow reemplaza 1

Para S2:
 se adelanta 0 (memoria de datos), reemplaza -2

$$E) CPI = \frac{N-1 + X \cdot b \cdot p \cdot (i-1) + eX}{eX} \approx \frac{bp(i-1) + e}{e}$$

\uparrow
 $X \gg N$

F) En este caso $b = e$, $p = 1$

$$\Rightarrow CPI = i$$

Se requeriría que la comparación de los saltos condicionales se realizará en la segunda etapa del pipeline.



Pregunta 3 (PEP 1 – 2018-01)

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 4. Procesador Monociclo (PEP 1 – 2019-01)

La implementación básica del procesador MIPS monociclo visto en clases (Figura 1), puede ejecutar sólo ciertas instrucciones. Es posible incluir nuevas instrucciones al conjunto de instrucciones (ISA), pero la decisión de hacerlo depende de, entre otras cosas, el costo y la complejidad que se introduce al camino de datos y el control del procesador. Considere las siguientes instrucciones nuevas:

lwi rt, rd(rs) → Interpretación: $\text{Reg}[rt] = \text{Mem}[\text{Reg}[rd] + \text{Reg}[rs]]$
bne rt, rs, BranchAddr → Interpretación: $\text{if}(\text{Reg}[rt] \neq \text{Reg}[rs]) \text{ PC} = \text{PC} + 4 + \text{BranchAddr}$

A) (0,6 pts.) ¿Qué bloques existentes se pueden utilizar para incluir lwi? ¿Y para incluir bne?

Para lwi se puede utilizar memoria de instrucciones, archivo de registros con puertos de lectura y escritura, ALU y memoria de datos. Para bne, además de los bloques anteriores, se puede utilizar el sumador para calcular la dirección de salto.

B) (0,6 pts.) ¿Qué nuevos bloques funcionales, si se requieren, necesitamos para lwi? ¿Y para bne?

Para lwi no se requiere ningún bloque adicional, se puede implementar con los bloques existentes. Para bne se requiere mayor lógica para determinar si los operandos son distintos, y una opción es incluir un circuito inversor de la señal “zero” de la ALU y un multiplexor que escoja entre “zero” y la señal invertida para alimentar el AND con la señal de control “branch”.

C) (0,6 pts.) ¿Qué nuevas señales de control, si se requieren, necesitamos para dar soporte a lwi? ¿Y para bne?

Para lwi no se requieren nuevas señales de control, basta con actualizar la lógica de control. Para bne sí se requiere una señal nueva que permita distinguir entre beq y bne, y una posibilidad es que controle el multiplexor mencionado en B).

D) (0,2 pts.) Después de incluir estas dos nuevas instrucciones, ¿cambia la tasa de reloj de este procesador monociclo? Explique.

La instrucción lwi tiene igual latencia total que lw, y bne sólo agrega un multiplexor y un inversor que no deberían ser parte de la latencia crítica. Además, la instrucción con mayor latencia es lw. Entonces, la tasa de reloj, dada por la mayor latencia entre todas las instrucciones, no cambia.

Pregunta 5 (PEP 1 – 2018-01)

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:

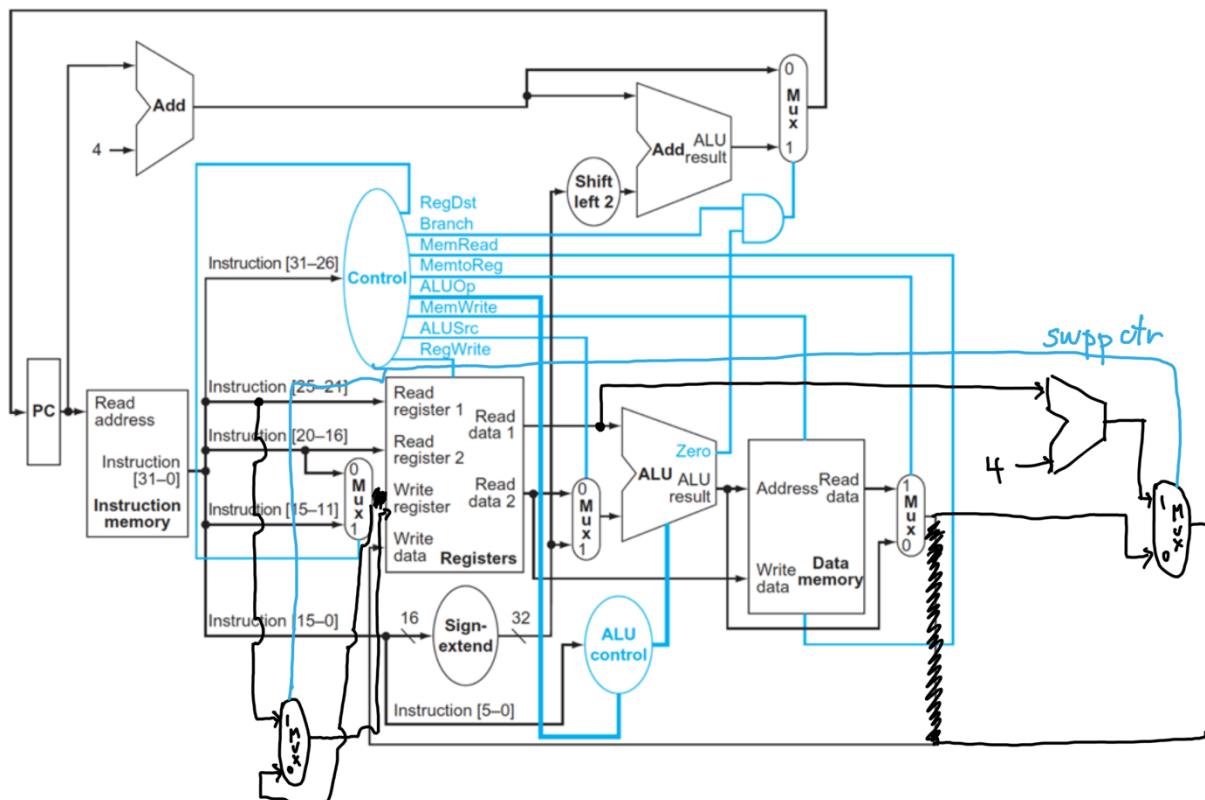
```
swpp $rt, imm($rs)
```

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:

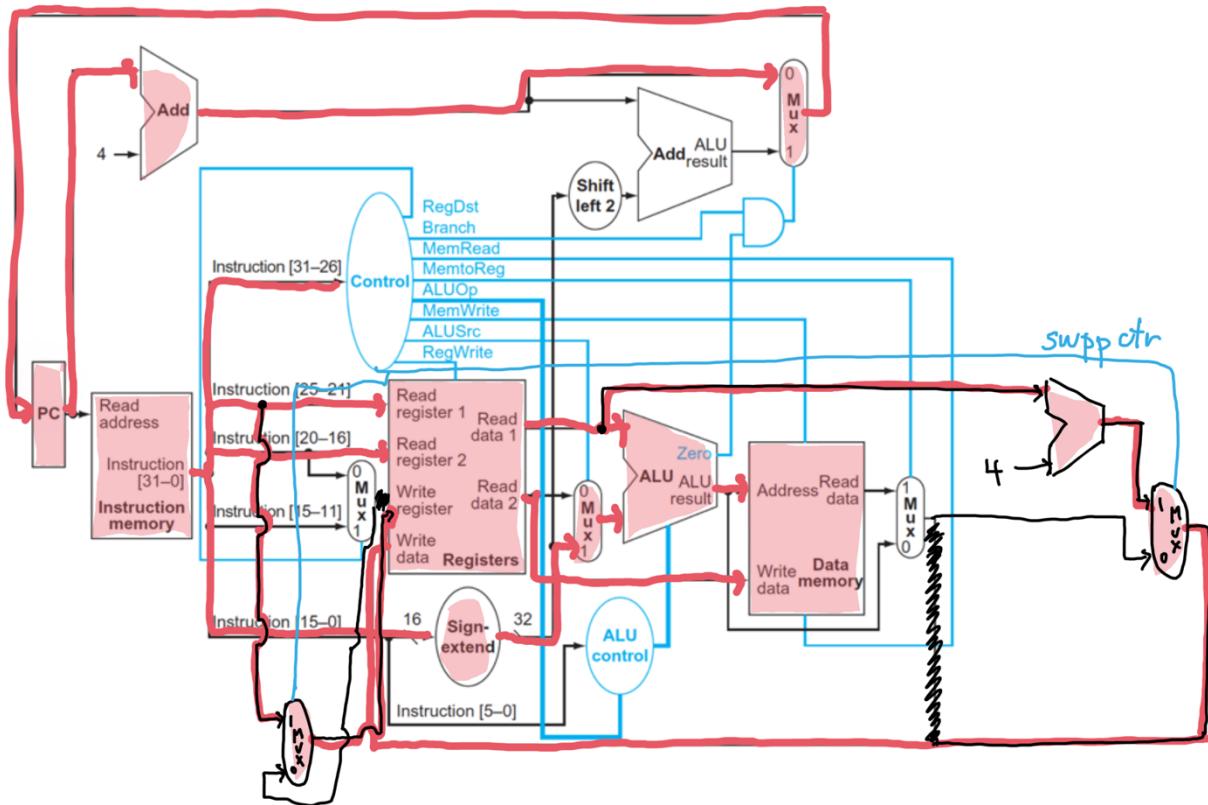
```
sw $rt, imm($rs)
addi $rs, $rs, 4
```

- 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 multiplexores 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(es) 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 6. Procesador Monociclo (PEP 1 – 2018-02)

En el diseño de procesadores, para considerar una posible mejora en el camino de datos del procesador, la decisión muchas veces depende del compromiso entre costo y desempeño. Considere el procesador monociclo de la Figura y las siguientes latencias, en ps, y costos, en \$, de los bloques que se indican:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Control
Latencia (ps)	400	100	30	120	200	350	100
Costo (\$)	1000	30	10	100	200	2000	500

Considere la adición de un multiplicador a la ALU. Esta potencial mejora agrega 300 ps a la latencia de la ALU, e incrementará su costo en \$600. Agregar el multiplicador significará una reducción del 25% del número total de instrucciones, pues ya no será necesario emular la instrucción MUL.

- A) (0,5 pts.) Indique qué función cumple cada uno de los dos bloques sumadores etiquetados como “Add” en el diagrama de la **Error! Reference source not found.**. ¿Participan en el camino de datos crítico que determina la tasa de reloj de este procesador?

El sumador de la izquierda incrementa el contador de programa, PC, en 4 para obtener la instrucción siguiente, y el de la derecha, le suma al PC el offset de un salto. Estos sumadores no contribuyen al camino de datos crítico, pues sus latencias son menores que aquellas de Regs y ALU, y no interfieren en los cálculos aritméticos o de direcciones de las instrucciones que toman mas tiempo.

- B) (0,5 pts.) ¿Cuál es la tasa de reloj de este procesador monociclo con y sin esta mejora?

El camino crítico está dado por una instrucción de acceso a memoria: I-Mem + Regs (toma mas tiempo que Control) + Mux (seleccionar entrada a ALU) + ALU + D-Mem + Mux (seleccionar valor de memoria para ser escrito en registros). Entonces, sin mejora se tiene una latencia total de 1130 ps (0,88 GHz), y con la mejora, 1430 ps (0,7 GHz) (como la ALU está en el camino crítico, simplemente se suman los 300 ps adicionales de latencia de la ALU en esta condición).

- C) (0,5 pts.) ¿Cuál es la aceleración lograda al incorporar esta mejora?

Con la mejora, el reloj es mas lento, pero se requieren 25% menos ciclos en total para el programa. La aceleración es $(1/0,75)*(1130/1430) = 1,054$. Es decir, el procesador con la mejora sería 5,4% mas rápido.

- D) (0,5 pts.) Compare la razón costo/desempeño con y sin esta mejora, y concluya sobre la conveniencia de introducir esta mejora.

Del diagrama del procesador monociclo, y considerando todos los elementos (no solo aquellos del camino crítico), tenemos un costo total de \$3890 sin la mejora, y de \$4490 con la mejora. El costo relativo es entonces 1,15, y la razón costo/desempeño, $1,15/1,054 = 1,09$. Es decir, conviene introducir la mejora pues pagamos mas, pero obtenemos un desempeño comparativamente mejor por el costo adicional.



Pregunta 7. Pipeline (PEP 1 – 2018-01)

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



Pregunta 8. Pipeline (PEP 1 – 2018-02)

Parte 1 de 2

Considere la siguiente secuencia de instrucciones que se ejecutan en un procesador MIPS con *pipeline* de 5 etapas (Figura).

```
add $t0, $t0, $t1
add $t0, $t0, $t1
add $t0, $t0, $t2
```

Inicialmente, los registros $\$t0$, $\$t1$ y $\$t2$ almacenan, respectivamente, los valores 0, 10 y 20. El procesador tiene una unidad de adelantamiento (*forwarding unit*) cuya lógica de detección de riesgos y manejo de señales de control es la siguiente:

```
if(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA = 10
if(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB = 10
if(MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01
if(MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
```

- A) (0,5 pts) Explique qué problema tiene esta implementación de adelantamiento de datos y determine los valores que almacenan $\$t0$, $\$t1$, $\$t2$ después de completarse la ejecución de las tres instrucciones.

El problema es que para el *tercer “add”* se adelantará el resultado de la operación del *primer “add”* (desde el registro MEM/WB) pues en la lógica de la unidad de adelantamiento se detectó el riesgo de datos entre estas dos instrucciones. Así los valores almacenados al terminar la ejecución de los tres “add” serán: $\$t0=30$, $\$t1=10$ y $\$t2=20$.

- B) (0,3 pts) Explique brevemente cómo resolvería el problema descrito en A).

Para resolver el problema, habría que modificar la lógica de la unidad de adelantamiento para adelantar el resultado almacenado en el registro EX/MEM, que es el más reciente, en vez de aquel del registro MEM/WB. Concretamente, habría que agregar la siguiente condición al primer “if” (y condiciones similares a los demás):

```
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs))
```

*No es necesario explicitar esta condición en la respuesta, basta con una explicación descriptiva.



Parte 2 de 2

Considere la siguiente secuencia de instrucciones que se ejecuta en un procesador MIPS con *pipeline* de 5 etapas (IF, ID, EX, MEM y WB). A y B son constantes, y el valor inicial del registro \$t0 es 1. Asuma un esquema de predicción de bifurcaciones “branch not taken” o bifurcación no tomada, y que la decisión de bifurcación se toma en la etapa EX.

```

FOR:    beq    $t0, $zero, NEXT
        lw     $t1, A($s0)
        lw     $t2, B($s0)
        add   $t3, $t1, $t2
        sw     $t3, A($s0)
        addi  $t0, $t0, -1
        j     FOR
NEXT:   addi  $t0, $t0, 1
        sw     $t0, A($s0)

```

A) (0,3 pts) Muestre los riesgos (*hazards*) de datos y de control que existen en este programa.

Hay un riesgo de control en la línea 1 (rc1).

Riesgos de datos: entre líneas 2 y 4 (rd1), entre líneas 3 y 4 (rd2), entre líneas 4 y 5 (rd3), entre líneas 6 y 1 (rd4; después del salto) y entre líneas 8 y 9 (rd5).

B) (0,3 pts) Asumiendo un procesador sin adelantamiento (*forwarding*), muestre donde y cuantas esperas (burbujas) son necesarias agregar para eliminar los riesgos detectados.

Sin adelantamiento, rd1 y rd4 requieren una burbuja; rd2, rd3 y rd5, dos.

En caso de una predicción incorrecta, rc1 se resuelve con dos burbujas (y “flush” de las dos “lw” que venían detrás) pues la decisión de bifurcación se toma en etapa EX.

C) (0,4 pts) Considere ahora un procesador con adelantamiento. ¿Se requiere agregar esperas para resolver los riesgos de datos y/o control? Dibuje un diagrama de ciclos de reloj del *pipeline* (ciclos avanzan de izquierda a derecha) para esta secuencia de instrucciones.

Para una predicción incorrecta, el rc1 de todas maneras requiere burbujas, independiente de la existencia de adelantamiento.

Para los riesgos de datos, rd1, rd3, rd4 y rd5 pueden resolverse con adelantamiento sin necesidad de burbujas. Sin embargo, dado que “lw” y “add” son consecutivas y el dato de “lw” no está disponible sino hasta la etapa MEM, rd2 no puede resolverse sólo con adelantamiento y se requiere una burbuja.

Inst. \ CR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
beq	IF	ID	EX	ME	WB												
lw		IF	ID	EX	ME	WB											
lw			IF	ID	EX	ME	WB										
add → nop				IF	ID	***	***	***									
add					ID	EX	ME	WB									
sw					IF	ID	EX	ME	WB								
addi						IF	ID	EX	ME	WB							
j							IF	ID	EX	ME	WB						
beq								IF	ID	EX	ME	WB					
lw → nop									IF	ID	***	***	***				
lw → nop										IF	***	***	***	***			
addi											IF	ID	EX	ME	WB		



sw											IF	ID	EX	ME	WB
----	--	--	--	--	--	--	--	--	--	--	----	----	----	----	----

En este esquema, *** representa una burbuja. Notar que la burbuja se mueve por el pipeline, y por eso se repite horizontalmente a medida que avanzan los ciclos de reloj. Sin embargo, esto no significa que aparezca una nueva burbuja. En cambio, más burbujas aparecen en esta secuencia al cancelar la predicción del beq, y éstas también se propagan por el pipeline. En total, se producen tres burbujas para este programa cuando se utiliza adelantamiento.

D) (0,2 pts) ¿Cuánto mejora el rendimiento al incluir adelantamiento?

Con adelantamiento, se requieren 17 ciclos de reloj para completar esta secuencia de instrucciones, mientras que, sin adelantamiento, como habría siete burbujas más, se requieren 24 ciclos para la misma secuencia. El rendimiento mejora entonces en 1,41 veces.



Pregunta 9. Pipeline (PA – 2018-01)

Considere el siguiente bucle en MIPS para un procesador con un *pipeline* de 5 etapas:

```
loop:    lw      r1, 0(r1)
          and   r3, r1, r2
          lw      r1, 0(r1)
          lw      r1, 0(r1)
          beq   r1, r0, loop
```

Asuma que se tiene predicción de saltos perfecta, es decir, no hay esperas por *hazards* de control, y que el *pipeline* dispone de *forwarding* completo. Además, suponga que se ejecutan muchas iteraciones de este bucle antes de que el bucle termine.

- A) (0,5 pts.) Indique qué tipo de instrucción (R, I o J) es cada una de las instrucciones del bucle.
lw→I, *and*→R, *beq*→I
- B) (0,5 pts.) Muestre un diagrama de ejecución del *pipeline* para la tercera iteración de este bucle, desde el ciclo en el cual se puede hacer *fetch* de la primera instrucción de esa iteración, hasta (pero no incluyendo) el ciclo en el cual se puede hacer *fetch* de la primera instrucción de la iteración siguiente. Muestre todas las instrucciones que están en el *pipeline* durante estos ciclos (no solo aquellos desde la tercera iteración).

LW R1,0(R1)	WB
LW R1,0(R1)	EX MEM WB
BEQ R1,R0,Loop	ID *** EX MEM WB
LW R1,0(R1)	IF *** ID EX MEM WB
AND R1,R1,R2	IF ID *** EX MEM WB
LW R1,0(R1)	IF *** ID EX MEM
LW R1,0(R1)	IF ID ***
BEQ R1,R0,Loop	IF ***

- C) (0,5 pts.) ¿En qué porcentaje de los ciclos se tiene que las cinco etapas del *pipeline* están haciendo trabajo útil? (por ejemplo, la instrucción “add r3, r1, r2” no realiza trabajo útil durante la etapa MEM).

En el diagrama anterior, las etapas marcadas en celeste no realizan trabajo útil (notar que *beq* está haciendo trabajo útil en MEM pues está determinando el valor correcto de la siguiente instrucción en esa etapa). En todos los ciclos hay o bien etapas en espera o sin realizar trabajo útil y, por lo tanto, en el 0% de los ciclos se tienen todas las etapas del *pipeline* haciendo trabajo útil.

- D) (0,5 pts.) Determine el rendimiento medido en CPI para estas condiciones.
Las 5 instrucciones del bucle son ejecutadas en 8 ciclos de reloj. Entonces, CPI = 8/5.



Pregunta 10. Pipeline (PEP 1 – 2019-01)

Parte 1

Considere la siguiente secuencia de instrucciones que se ejecuta en un procesador MIPS con *pipeline* de 6 etapas, similar a MIPS con *pipeline* de 5 etapas, pero en el que IF se ha separado en dos etapas. Es decir, las etapas del *pipeline* de este procesador son: IF1, IF2, ID, EX, MEM y WB. Los valores iniciales de los registros \$t1 y \$t2 son 1 y 0, respectivamente. Asuma un esquema de predicción de bifurcaciones “branch not taken” o bifurcación no tomada, que la decisión de bifurcación se toma en la etapa ID, y que la dirección de destino de un salto incondicional está inmediatamente disponible en IF1.

```

1      FOR:  beq    $t1, $t2, NEXT
2          lw     $t3, 0($$0)
3          lw     $t4, 0($$0)
4          sub    $t5, $t4, $t3
5          addi   $t2, $t1, 1
6          j      FOR
7      NEXT: add    $t0, $t2, $t5

```

E) (0,3 pts.) Muestre los riesgos (*hazards*) de datos y de control que existen en este programa.

Hay un riesgo de control en la línea 1 (rc1).

Riesgos de datos: entre líneas 2 y 4 (rd1), entre líneas 3 y 4 (rd2) y entre líneas 5 y 1 (rd3).

F) (0,2 pts.) Asumiendo un procesador **sin** adelantamiento (*forwarding*), indique cuantas esperas (burbujas) son necesarias agregar para eliminar cada uno de los riesgos identificados. **No** es necesario mostrar un diagrama de ciclos de reloj en esta parte.

Sin adelantamiento, rd1 y rd3 requieren una burbuja, y rd2, dos.

En caso de una predicción incorrecta, rc1 se resuelve con dos burbujas (y “flush” de las dos “lw” que venían detrás) pues la decisión de bifurcación se toma en etapa ID.

G) (0,2 pts.) Considere ahora un procesador **con** adelantamiento. ¿Cambian sus respuestas en B)? Y en caso afirmativo, ¿en cuánto?

Sí, ahora sólo se requiere una burbuja para rd2, y ninguna para rd1 y rd3. Rc1 sigue igual.

H) (0,7 pts.) Dibuje un diagrama de ciclos de reloj del *pipeline* (ciclos avanzan de izquierda a derecha) para esta secuencia de instrucciones considerando adelantamiento.

Inst. \ CR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
beq	IF1	IF2	ID	EX	ME	WB										
lw		IF1	IF2	ID	EX	ME	WB									
lw			IF1	IF2	ID	EX	ME	WB								
sub → nop				IF1	IF2	ID	***	***	***							
sub						ID	EX	ME	WB							
addi					IF1	IF2	IF2	ID	EX	ME	WB					
j						IF1	IF1	IF2	ID	EX	ME	WB				
beq							IF1	IF2	ID	EX	ME	WB				
lw → nop								IF1	IF2	***	***	***	***	***		
lw → nop									IF1	***	***	***	***	***		
add										IF1	IF2	ID	EX	ME	WB	



Parte 2

- A) (0,6 pts.) Encuentre una expresión para calcular el rendimiento, medido en CPI, de un procesador con *pipeline* de “*N*” etapas (*N*>6) que ejecuta un programa de “*I*” instrucciones en el que se detectan “*B_k*” riesgos de datos que se resuelven con “*k*” burbujas. Considere $B_k = I/(2k)$ para $i = \{1, 2, 3, 4, 5, 6\}$ y $B_k = 0$ en cualquier otro caso.

$$\text{CPI} = \# \text{ciclos} / \# \text{instr} = (I + N-1 + \sum(B_k * k)) / I = (I + N-1 + 6*I/2) / I = 4 + (N-1)/I$$



ORGANIZACIÓN DE COMPUTADORES

SEMESTRE DE VERANO 2018

PEP1 – Solución Propuesta

Profesor: Néstor González Valenzuela

Fecha: 5 de enero de 2018

Tiempo Disponible: 100 minutos

Datos a usar:

Instrucciones tipo A, de cálculo:	1 ciclo
Instrucciones tipo B, de load/store:	2 ciclos
Instrucciones tipo C, de salto:	3 ciclos
Velocidad de procesador:	3 GHz

Código MIPS:

```
add    $r2, $r0, $r0
otra: beq    $r2, $r8, fin
      add    $r3, $r2, $r9
      lw     $r4, 0($r3)
      sw     $r4, 1($r3)
      addi   $r2, $r2, 2
      beq    $r0, $r0, otra
fin:
```

Pregunta 1.- Materia: Performance. (2,0 puntos en total)

Considere los siguientes datos iniciales para el código MIPS: r0 = 0 ; r8 = 6.

Si necesita otro valor para sus respuestas, defínalo usted mismo(a).

Con la información dada:

- Determine el tiempo de ciclo para este procesador. (0,2 ptos.)

R: La velocidad del procesador es de 3Ghz = 3×10^9 Hz. Como 1 Hz es 1 ciclo/seg, luego el ciclo de reloj o tiempo de ciclo es de $1/3 \times 10^{-9}$ seg, ó 0,33 ns [ns = nano segundos], ó 330 ps.

(Nota: el libro P&H expresa los tiempos en ps [pico segundos]



- b. Determine la cantidad de instrucciones de cada tipo A, B, C para la ejecución completa del código. (0,2 ptos.)

R: Con $r0 = 0$; $r8 = 6$ se ejecutan

Inicio y primer ciclo: $r0=0$	(r2=2)	add \$r2, \$r0, \$r0 otra: beq \$r2, \$r8, fin add \$r3, \$r2, \$r9 lw \$r4, 0(\$r3) sw \$r4, 1(\$r3) addi \$r2, \$r2, 2 beq \$r0, \$r0, otra fin: (r2=2)	A C A B B A C	add \$r2, \$r0, \$r0 otra: beq \$r2, \$r8, fin add \$r3, \$r2, \$r9 lw \$r4, 0(\$r3) sw \$r4, 1(\$r3) addi \$r2, \$r2, 2 beq \$r0, \$r0, otra fin: (r2=4)	C A B B B A C	add \$r2, \$r0, \$r0 otra: beq \$r2, \$r8, fin add \$r3, \$r2, \$r9 lw \$r4, 0(\$r3) sw \$r4, 1(\$r3) addi \$r2, \$r2, 2 beq \$r0, \$r0, otra: (r2=6)	C A B B B A C
-------------------------------	---------------	---	---------------------------------	---	---------------------------------	---	---------------------------------

Se ejecutan 7 instrucciones de tipo A, 6 instrucciones de tipo B y 7 instrucciones de tipo C. En total 20 instrucciones.

- c. Determine la frecuencia correspondiente a cada tipo de instrucción. (0,1 pto.)

R: Tipo A: $7/20 = 35\%$; Tipo B: $6/20 = 30\%$; Tipo C = $7/20 = 35\%$

- d. Calcule los CPI para cada tipo de instrucción. (0,2 ptos.)

R: Los CPI de cada tipo de instrucción son los datos dados.

CPI Instrucciones Tipo A: 1; CPI Instrucciones CPI Tipo B: 2; CPI Instrucciones Tipo 3: 3

- e. Calcule los CPI para el programa completo. (0,2 ptos.)

$$CPI = \frac{7 \times 1 + 6 \times 2 + 7 \times 3}{20} = 2$$

o bien:

$$CPI = 1 \times 0,35 + 2 \times 0,3 + 3 \times 0,35 = 2$$

- f. Asuma que se pueden hacer dos mejoras. Mejora1: disminuir un ciclo de las instrucciones de tipo B; y Mejora2, disminuir un ciclo de las instrucciones de tipo C. Para las condiciones del código dado y usando ley de Amdahl ¿Cuál mejora recomienda? (0,5 ptos)

$$\text{Ley de Amdahl} \Rightarrow A = \frac{1}{(1 - Fm) + Fm/A_m}$$

De los datos dados:

Mejora 1: Fracción de mejora = $12/40 = 0,3$; mejora = 1 ciclo de 2 \Rightarrow mejora el doble = 2

$$A_1 = \frac{1}{(1 - 0,3) + 0,3/2} = \frac{1}{0,7 + 0,15} = \frac{1}{0,85} = 1,176$$

Luego, mejora1 = 17.6%



Mejora 2: Fracción de mejora=21/40 = 0,525; mejora = 1 ciclo de 3 => mejora = 3/2

$$A_1 = \frac{1}{(1 - 0,525) + \frac{0,525}{1,5}} = \frac{1}{0,475 + 0,35} = \frac{1}{0,825} = 1,212$$

Luego, mejora 2 = 21,2%

Confirmación calculando los CPI

$$CPI \text{ mejora 1} = 1x0,35 + 1x0,3 + 3x0,35 = 1,70$$

$$\text{Speedup} = 2/1,7 = 1,176 \Rightarrow 17,6\%$$

$$CPI \text{ mejora 2} = 1x0,35 + 2x0,3 + 2x0,35 = 1,65$$

$$\text{Speedup} = 2/1,65 = 1,212 \Rightarrow 21,2\%$$

Luego: se recomienda la mejora 2.-

- g. Suponga que cada tipo de mejora tiene un costo de implementación ¿Puede calcular cuál sería la relación de costo entre las mejoras que haría cambiar la decisión tomada en el punto anterior? Si puede, entonces haga el cálculo. Si no puede entonces explique porqué. (0,6 ptos).

R: Depende cómo se enfrente el problema por el alumno. Pero en general, se debe plantear una relación Coto/Beneficio. En principio, si la mejora 2, que es un 3,6% superior a la mejora 1 tiene un costo mayor, la decisión se podría cambiar. Si, para simplificar, se otorga el mismo peso al Costo que al Speedup logrado, se puede plantear una regla de 3. Si hubiera otras consideraciones, siempre sería posible plantear una ecuación para determinar la mejor decisión incluyendo los pesos definidos. Si el Costo de implementación y el Speedup tuvieran el mismo peso, entonces bastaría con que la mejora 2 tenga un Costo (C2) mayor que 1,031 veces el Costo (C1) de la mejora 1 para revertir la decisión.

Pregunta 2.- Pipeline y hazards. (2,0 puntos en total)

Considere el mismo código MIPS. Pero con datos iniciales r0 = 0 ; r8 = 4.

Considere que el procesador funciona con pipeline de 5 etapas: IF, ID, EX, MEM y REG

Considere que es la primera versión del procesador y por lo tanto no tiene implementada ninguna mejoría para resolver los estancamientos del pipeline.

(En principio, considere que el branch es taken)

- a. Determine todas las dependencias de datos. Indique aquellas que pueden producir estancamiento del pipeline. (0,5 ptos.)

add	\$r2, \$r0, \$r0	
otra:	beq \$r2, \$r8, fin	\$r2 se escribe en la instrucción previa: RAW, estancaría el pipeline
add	\$r3, \$r2, \$r9	\$r2 se lee y fue escrito en add previo: RAW, estancaría el pipeline
lw	\$r4, 0(\$r3)	\$r3 se lee y fue escrito en add previo: RAW, estancaría el pipeline
sw	\$r4, 1(\$r3)	\$r3 se lee y fue escrito en add previo: RAW, estancaría el pipeline
		\$r4 se lee y fue escrito en lw previo: RAW, estancaría el pipeline
addi	\$r2, \$r2, 2	Se escribe \$r2 que será usado en otra: RAW, estancaría el pipeline.
beq	\$r0, \$r0, otra	

fin:



- b. Desarrolle el comportamiento del pipeline del programa completo bajo las condiciones establecidas. (0,5 ptos.)

	CICLOS	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	29	30	31	32	33	34	35	36	37
add	\$r2, \$r0, \$r0	IF	ID	EX	ME	RE	se deben a \$r2																															
beq	\$r2, \$r8, fin	IF	ID	** **		EX	ME	RE																														
add	\$r3, \$r2, \$r9	IF	** **		** **		EX	ME	RE																													
lw	\$r4, 0(\$r3)						IF	ID	** **		EX	ME	RE																									
sw	\$r4, 1(\$r3)						IF	** **		** **		ID	EX	ME	RE																							
addi	\$r2, \$r2, 2										IF	** **		ID	EX	ME	RE																					
beq	\$r0, \$r0, otra											IF	ID	EX	ME	RE																						
beq	\$r2, \$r8, fin												IF	ID	EX	ME	RE																					
add	\$r3, \$r2, \$r9													IF	ID	EX	ME	RE																				

- c. Asuma que se ha mejorado el procesador para que tenga una unidad de predicción de saltos perfecta, es decir, predice con 100% de certeza ¿Disminuye el tiempo de ejecución del programa? Si su respuesta es positiva, muestre cómo fue que lo determinó haciendo uso de información extraída desde el pipeline y diga cuánto fue en términos de tiempo (no de ciclos). Si su respuesta es negativa, explique porqué. (0,5 ptos.)

R: Del pipeline de la respuesta b) se deduce que hay 6 ciclos perdidos debidos a los saltos. Entonces, el tiempo perdido sería = $330 \times 6 = 1.980\text{ps} = 1.98\text{ns}$

- d. Si la mejora propuesta en el punto previo no ha resuelto todos los estancamientos del pipeline. Agregue usted las mejoras necesarias para que se reduzcan al mínimo, o que se eliminen, todos los estancamientos. Con las mejoras propuestas, muestre en qué se modificó el comportamiento del pipeline. (0,5 ptos.)

R: Queda por eliminar los estancamientos debidos a las dependencia de datos de tipo RAW. Con mecanismos de Forwarding se pueden eliminar dichas esperas.

	CICLOS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21				
add	\$r2, \$r0, \$r0	IF	ID	EX	ME	RE																				
beq	\$r2, \$r8, fin	IF	ID	EX	ME	RE																				
add	\$r3, \$r2, \$r9	IF	ID	EX	ME	RE																				
lw	\$r4, 0(\$r3)						IF	ID	EX	ME	RE															
sw	\$r4, 1(\$r3)						IF	ID	EX	ME	RE															
addi	\$r2, \$r2, 2											IF	ID	EX	ME	RE										
beq	\$r0, \$r0, otra												IF	ID	EX	ME	RE									
beq	\$r2, \$r8, fin													IF	ID	EX	ME	RE								
add	\$r3, \$r2, \$r9														IF	ID	EX	ME	RE							
lw	\$r4, 0(\$r3)															IF	ID	EX	ME	RE						
sw	\$r4, 1(\$r3)																IF	ID	EX	ME	RE					
addi	\$r2, \$r2, 2																	IF	ID	EX	ME	RE				
beq	\$r0, \$r0, otra																		IF	ID	EX	ME	RE			
beq	\$r2, \$r8, fin																			IF	ID	EX	ME	RE		
add	\$r3, \$r2, \$r9																				IF	ID	EX	ME	RE	



Pregunta 3.- Trayectoria de Datos (2,0 puntos en total)

Considere la figura de la página 3.

Los bloques lógicos usados en la implementación tienen las siguientes latencias:

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

- a. Determine las latencias de las distintas posibles trayectorias de este procesador asumiendo que la Unidad de Control tiene latencia cero o es despreciable. (0,5 ptos.)

R: Consideremos tres tipos de instrucciones representativas:

Instrucción Load: I-Mem + Mux + Regs + Mux + ALU + D-Mem + Mux

$$200 + 20 + 90 + 20 + 90 + 250 + 20 = 690\text{ps}$$

Instrucción jump: I-Mem + Add + ShLf2 + Mux

$$200 + 10 + 20 = 230\text{ps}$$

Instrucción tipo R (add): I-Mem + Mux + Regs + Mux + ALU + Mux

$$200 + 20 + 90 + 20 + 90 + 20 = 440\text{ps}$$

- b. Determine la Trayectoria Crítica y cuál es el tipo de instrucción para la cual ocurre, indicando las unidades funcionales (bloques) que utiliza (0,5 ptos)

R: La Trayectoria Crítica es el caso de Load, con 690ps. Las unidades funcionales están indicadas en la respuesta anterior.

- c. Para evitar hacer aun más lenta la Trayectoria Crítica determinada en el punto anterior, se busca generar la señal MemWrite lo más rápida y tempranamente posible. ¿Cuánto tiempo debe tomar la Unidad de Control para generar MemWrite para cumplir esta premisa? Explique. (1,0 pto).

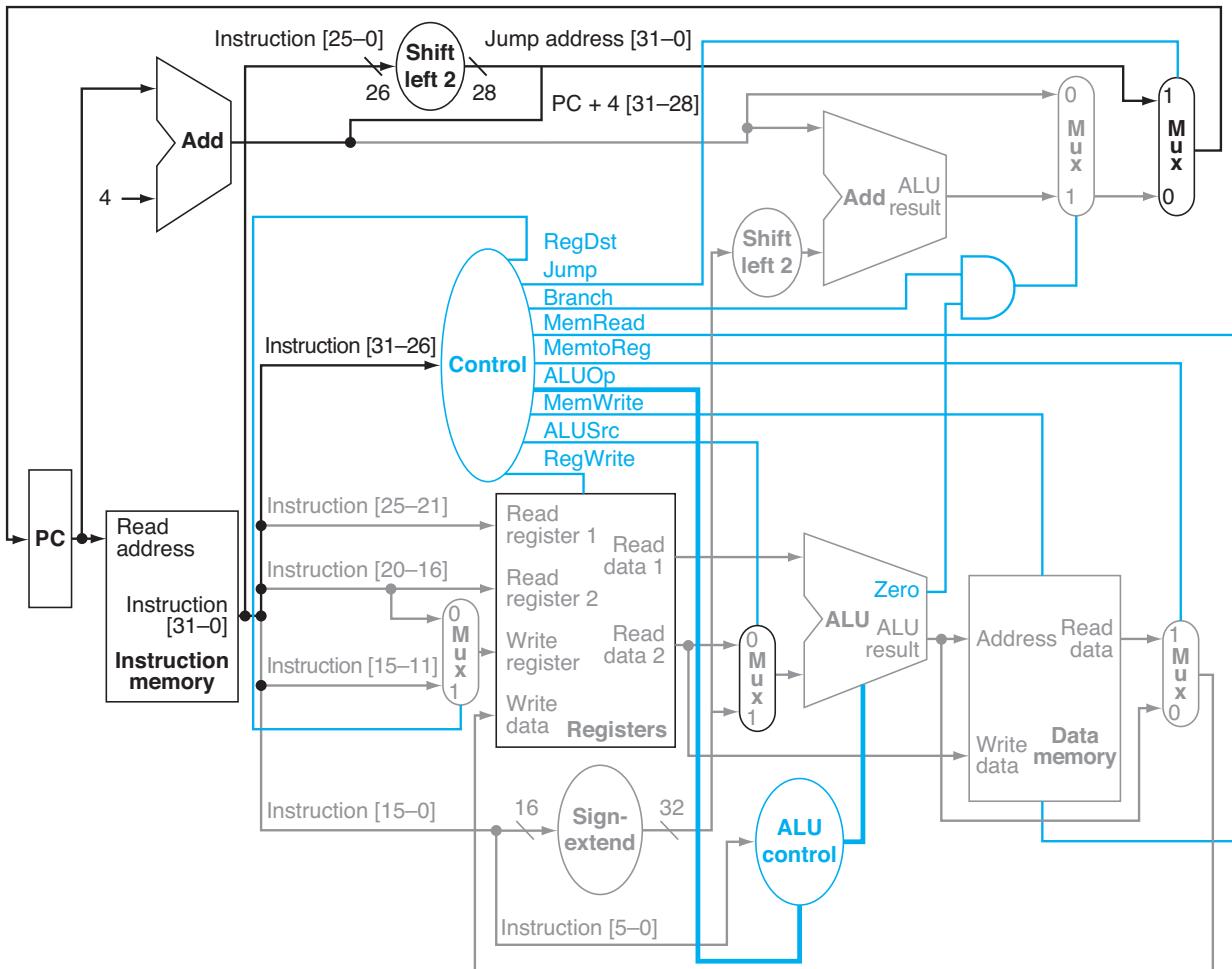
La Unidad de Control solo puede comenzar a generar MemWrite después de leer I-Mem. Debería terminar de generar esta señal antes del término de ciclo de reloj. MemWrite es una señal que habilita D-Mem (en rigor habilita los flip-flops de D-Mem). Si la escritura es habilitada por la transición del reloj, entonces la señal MemWrite debe llegar antes que ello ocurra, de lo contrario debe esperar un ciclo de reloj. Entonces, MemWrite debe ser generado en un ciclo de reloj menos el tiempo de acceso a I-Mem. Considerando la Trayectoria Critica de 690ps como la que define el ciclo de reloj, entonces MemWrite debería generarse en $690 - 200 = 490\text{ps}$.

Bonus Track: Ganará 1 punto adicional si escribe el código C que corresponde al código MIPS usado en la pregunta 1.

R: Código C

```
for (i=0; i=j; i+=2)
    a[i+1] = a[i];
```

Nota: En el código MIPS se tendría: i en \$r2; j en \$r8; a en \$r9



Fecha: 18 de octubre de 2019
 Duración: 90 minutos

Pregunta 1. Rendimiento (2 pts.)

Parte 1

Considere dos implementaciones distintas del mismo conjunto de instrucciones (ISA). Las instrucciones pueden ser divididas en cuatro clases de acuerdo con su CPI: clase A, B, C y D. La implementación P1 tiene una tasa de reloj de 2,5 GHz y CPIs de 1, 2, 3 y 3, y la implementación P2, una tasa de reloj de 3 GHz y CPIs de 2, 2, 2 y 2, para las clases A, B, C y D, respectivamente. Dado un programa con un total de un millón de instrucciones divididas en las clases como sigue: 10% clase A, 20% clase B, 50% clase C, y 20% clase D.

- A) (0,4 pts.) ¿Cuál es más rápida, P1 o P2?
- B) (0,4 pts.) ¿Cuál es el CPI global para P1 y P2?
- C) (0,4 pts.) Encuentre el número de ciclos de reloj que requieren P1 y a P2 para ejecutar el programa.

Parte 2

Considere el siguiente programa escrito en lenguaje C. Se pide comparar el desempeño de un procesador MIPS monociclo con tasa de reloj de 1 GHz y un procesador MIPS con *pipeline* de 5 etapas, como el discutido en clases, con tasa de reloj de 3 GHz. Asuma que el procesador con *pipeline* dispone de adelantamiento completo (*forwarding*), predicción perfecta de bifurcaciones y que las direcciones en saltos incondicionales están disponibles en la etapa IF.

```
int i;
for (i=0; i<10000; i=i+1)
    tangananica[i] = tanganana[i] - 10;
```

- D) (0,4 pts.) Determine el CPI para ambos procesadores.
- E) (0,4 pts.) Calcule el tiempo de ejecución para ambos procesadores.

$$\text{A) tiempo CPU} = \frac{\# \text{Instr.} \times \text{CPI}}{\text{tasa reloj}} = \frac{\sum \# \text{Instr}_i + \text{CPI}_i}{\text{tasa reloj}}$$

$$\text{tiempo P1} = \left(1 \times 10^5 \times 1 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 3 + 2 \times 10^5 \times 3 \right) / 2,5 \times 10^9$$

$$= 10,4 \times 10^{-4} \text{ s}$$

$$\text{tiempo P2} = \left(1 \times 10^5 \times 2 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 2 + 2 \times 10^5 \times 2 \right) / 3 \times 10^9$$

$$= 6,66 \times 10^{-4} \text{ s}$$

→ Es más rápida P2

$$\text{B) CPI} = \frac{\text{tiempo CPU} \times \text{tasa reloj}}{\# \text{Instr}}$$

$$\text{CPI(P1)} = \frac{10,4 \times 10^{-4} [\text{s}] \times 2,5 \times 10^9 [\text{Hz}]}{1 \times 10^6} = 2,6$$

$$\text{CPI(P2)} = \frac{6,66 \times 10^{-4} [\text{s}] \times 3 \times 10^9 [\text{Hz}]}{1 \times 10^6} = 2,0$$

$$c) \text{ #ciclos} = CPI \times \#instr$$

$$\text{#ciclos (P1)} = 2,6 \cdot 10^6$$

$$\text{#ciclos (P2)} = 2 \cdot 10^6$$

D)

Notar que el programa incluye, dependiendo del compilador, más menos las sgtes instrucciones:

beq (verificar si seguir iterando)

lw (cargar tangana[i])

add (sumar -10 a lo cargado arriba)

sw (guardar en tangenica[i])

addi (actualizar índice)

j (saltar para nueva iteración)

CPI monociclo = 1 pues cada instrucción se ejecuta

$$\text{CPI pipeline} = \frac{\text{ciclos para llenar pipeline} + \text{ciclos por cada instr.} + \text{burbuja}}{\text{total instr.}} \approx \frac{7}{6}$$

ciclos para llenar pipeline
 ciclos por cada instr.
 burbuja

Se podría argumentar que el compilador reordene instrucciones para evitar burbuja insertando addi entre lw, add. En tal caso, CPI $\approx 1 + 2(1/6) + 1/6 = 1.67$

$$e) \text{tiempo monociclo} = \frac{1 \times 6 \times 10^4}{1 \times 10^9} = 6 \times 10^{-5} \text{ s}$$

$$\text{tiempo pipeline} = \frac{\frac{7}{6} \times 6 \times 10^4}{3 \times 10^9} = \frac{7}{3} \times 10^{-5} \text{ s}$$



Pregunta 2. Procesador Monociclo (2 pts.)

Considere el procesador MIPS monociclo visto en clases (Figura 1). Suponga que en determinado ciclo de reloj se está ejecutando la instrucción '10101100011000100000000000010100' que está almacenada en la dirección de memoria de instrucciones 0x000001A0. Asuma que la memoria de datos contiene sólo ceros y que los registros del procesador tienen los siguientes valores al inicio del ciclo en que se realiza "fetch" de esta instrucción:

R0	R1	R2	R3	R4	R5	R6	R8	R12	R31
0	-1	2	-3	-4	10	6	8	2	-16

- A) (0,2 pts.) ¿Cuáles son las salidas de los bloques "sign-extend" y el "shift left 2" de más arriba en la Figura 1?
- B) (0,3 pts.) ¿Cuáles son los valores de *todas* las entradas de la unidad ALU Control?
- C) (0,3 pts.) ¿Cuál es el nuevo valor del contador de programa al terminar la ejecución de esta instrucción? Además, marque en la Figura 1 el camino de datos por el cual este valor es determinado.
- D) (0,4 pts.) Para cada multiplexor, indique los valores de salida.
- E) (0,4 pts.) Para la ALU y las dos unidades de suma (Add), ¿cuáles son los valores de entrada?
- F) (0,4 pts.) ¿Cuáles son los valores de *todas* las entradas del archivo de registros (unidad "Registers")?

Notar que instrucción es "sw" pues opcode = 101011 = 43 (Tabla 2)

A) "Sign-extend" : 00 ... 010100 (32 bits)

El "immediate" es positivo → se concatenan 16 ceros a la izquierda

"shift left 2" : 0001100010...01010000 (28 bits)

A los 26 bits de más la derecha se le agregan 2 ceros a la derecha

B) ALUOp : 00 (se deduce de Tabla 1)

Instruction [5-0] : 010100

c) Nuevo valor PC : 0x0000001A4 (simplemente sumar 4)
* ver camino en Fig.

D) mux WrReg : 2 o 0 (RegDst es "don't care" para sw)

mux ALU : 20

mux Mem/ALU : X

mux Branch : PC+4 (parte c)

mux Jump : PC+4 (parte c)

E)

ALU : -3 y 20

Add (PC+4) : PC , 4

Add (Branch) : PC+4 , 20x4

F)

Read Register 1 : 00011 = 3

Read Register 2 : 00010 = 2

Write Register : 00000 = 0

Write Data : 0 (mem de datos con ceros)

RegWrite : 0

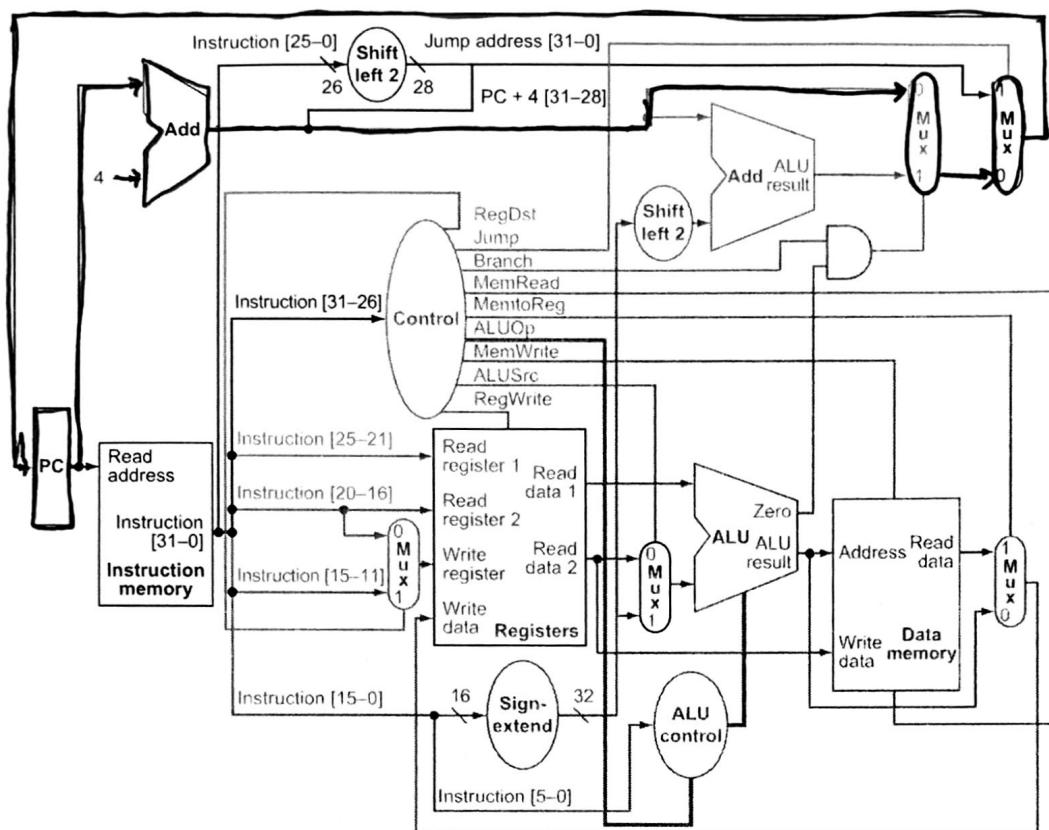


Figura 1. Diagrama de un procesador monociclo. Se ha marcado el camino de datos de la instrucción "jump".

Tabla 2. Opcode de algunas instrucciones MIPS

Instrucción	Opcode (decimal)
slt	0
add	0
sub	0
j	2
beq	4
bne	5
addi	8
lw	35
sw	43

Tabla 1. Tabla de verdad de ALU Control ($X = \text{"don't care"}$)

ALUOp	Campo "Funct"	Salida de ALU Control	Operación de ALU
00	XXXXXX	0010	Suma
X1	XXXXXX	0110	Resta
IX	XX0000	0010	Suma
IX	XX0010	0110	Resta
IX	XX0100	0000	"Y" lógico
IX	XX0101	0001	"O" lógico
IX	XX1010	0111	"Set on less than"



Pregunta 3. Pipeline (2 pts.)

Consideré un procesador MIPS con *pipeline* de 5 etapas, en el que, de todas las instrucciones ejecutadas, las siguientes fracciones de instrucciones tienen un tipo específico de dependencia de datos. El tipo de dependencia es identificado por la etapa que produce el resultado, EX o MEM, y la instrucción que requiere el resultado, es decir, la primera instrucción que sigue a aquella que produce el resultado, la segunda instrucción que sigue, o ambas. Asuma que el CPI del procesador sin riesgos de datos es 1.

Sólo EX a 1a instr.	Sólo MEM a 1a instr.	Sólo EX a 2a instr.	Sólo MEM a 2a instr.	EX a 1era instr. y MEM a 2da instr.	Otras
5%	20%	5%	10%	10%	10%

Además, asuma que se tienen las siguientes latencias, en ps, para las etapas del *pipeline*. Note que, para la etapa EX, se entregan distintas latencias dependiendo si el procesador tiene o no adelantamiento (ad) y de qué tipo.

IF	ID	EX (sin ad)	EX (con ad completo)	EX (ad sólo de EX/MEM)	EX (ad sólo de MEM/WB)	MEM	WB
150	100	120	150	140	130	120	100

- A) (0,4 pts.) En el caso sin adelantamiento, ¿qué fracción de ciclos corresponden a esperas debido a riesgos de datos?
- B) (0,4 pts.) En el caso con adelantamiento completo (adelantamiento de todos los resultados que se puedan adelantar), ¿qué fracción de ciclos corresponden a esperas debido a riesgos de datos?
- C) (0,4 pts.) Suponga que es muy costoso tener los multiplexores de 3 entradas necesarios para un adelantamiento completo. Se quiere decidir si es mejor adelantar sólo de EX/MEM (adelantamiento del siguiente ciclo) o sólo de MEM/WB (adelantamiento de 2 ciclos). ¿Qué caso resulta en menos ciclos de espera?
- D) (0,4 pts.) ¿Cuál es la aceleración lograda al incorporar adelantamiento completo al procesador con *pipeline* sin adelantamiento?
- E) (0,4 pts.) Suponga que se incorpora un “adelantamiento de viaje en el tiempo”, llamado McFly, que elimina **todos** los riesgos de datos. Asuma que la circuitería para el adelantamiento McFly agrega 100 ps a la latencia de la etapa EX con adelantamiento completo. ¿Es más rápido este procesador McFly que el procesador con adelantamiento completo? ¿Por cuánto?

A) Dependencias con la 1a instrucción (35% de los casos) generan 2 esperas (notar que si hay dependencia con 1a y 2a instrucción también requiere 2 esperas).

Dependencias sólo con 2a instrucción (15%) generan 1 espera luego, $CPI = 1 + 35\% \times 2 + 15\% \times 1 = 1,85$

$$\rightarrow \text{Porcentaje de esperas } \frac{0,85}{1,85} = 46\%$$

B) Con adelantamiento completo sólo genera 1 espera aquellos riesgos de MEM a la 1a instrucción (caso Iw \rightarrow tipo R)

$$CPI = 1 + 20\% + 1 = 1,2$$

$$\rightarrow \text{Porcentaje de esperas } \frac{0,2}{1,2} = 16,6\%$$

C)

- Con adelantamiento de EX/MEM, solo EX a la se resuelve sin esperar, todas las otras dependencias requieren 1 espera. $CPI = 1 + 20\% \times 1 + 5\% \times 1 + 10\% \times 1 + 10\% \times 1 = 1,45$
- Con adelantamiento de MEM/WB: EX a 2a \rightarrow sin esperar
MEM a 7a \rightarrow 2 espera
EX a 1a \rightarrow 1 espera
 $CPI = 1 + 5\% \times 1 + 20\% \times 1 + 10\% \times 1 = 1,35$

\therefore Conviene adelantamiento de MEM/WB

D) En A), B) ya calculamos CPI

$$\text{Aceleración} = \frac{\text{tiempo CPU s/ad}}{\text{tiempo CPU c/ad}} = \frac{CPI_{s/ad}}{CPI_{c/ad}}$$

(se cancelan #instr.)

$$= \frac{1,85}{1,20} = 1,54$$

tasa reloj, pues son los mismos)

E)

Notar que la etapa BX es la más lenta, y por tanto define el tiempo de reloj mínimo.

$$\text{Tiempo por instrucción con adelantamiento} = CPI_{c/ad} \times 150 \text{ ps} = 180 \text{ ps}$$

$$\text{, , , , McFly} = CPI_{\text{perfecto}} \times 250 \text{ ps} = 250 \text{ ps}$$

\therefore No es más rápido el McFly. Aceleración $= \frac{180 \text{ ps}}{250 \text{ ps}} = 0,72$