



پاسخ مسئله‌ی ۱.

الف

در این بخش فرض می‌کنیم هیچ وابستگی‌ای بین دستورات وجود ندارد.

$$SpeedUp = \frac{T_1}{T_4} = \frac{CPI_1 \times \frac{1}{CR_1}}{CPI_4 \times \frac{1}{CR_4}} = \frac{\frac{5}{1/5}}{\frac{1}{1/4}} = 4$$

اولین حلقه ۴ کلاک نیاز دارد و حلقه‌های بعدی در ۱ کلاک اجرا می‌شوند.

ب

$$\begin{cases} EX \text{ to } 1^{st} \text{ Only} = 2 \text{ Delay Cycle} \\ MEM \text{ to } 1^{st} \text{ Only} = 0 \text{ Delay Cycle} \\ EX \text{ to } 2^{nd} \text{ Only} = 1 \text{ Delay Cycle} \\ MEM \text{ to } 2^{nd} \text{ Only} = 0 \text{ Delay Cycle} \\ EX \text{ to } 1^{st} = 2 \text{ Delay Cycle} \\ EX \text{ to } 2^{nd} = 2 \text{ Delay Cycle} \end{cases}$$

$$SpeedUp = \frac{T_1}{T_4} = \frac{\frac{CPI_1}{CR_1}}{\frac{CPI_4}{CR_4}} = \frac{5 \times 2}{(0.5 \times 3 + 0.2 \times 1 + 0.1 \times 2 + 0.1 \times 1 + 0.5 \times 3 + 0.5 \times 3 + 0.5 \times 1) \times 2/5} \approx 3.07$$

ج

فقط برای وابستگی $MEM \text{ to } 1^{st} \text{ only}$ باید ۱ کلاک به تاخیر اضافه کنیم. پس:

$$SpeedUp = \frac{5 \times 2}{(0.2 \times 2 + 0.8 \times 1) \times 2/5} = \frac{10}{1.2} \approx 3.33$$

د

در این صورت برای محاسبه شرط Branch باید یک چرخه Stall داشته باشیم. پس:

$$SpeedUp = \frac{5 \times 2}{(0.2 \times 2 + 0.1 \times 2 + 0.7 \times 1) \times 2/5} \approx 3.08$$

ه

$$SpeedUp = \frac{5}{0.2 \times 2 + 0.1 \times (0.2 \times 1 + 0.8 \times 2) + 0.7 \times 1} \times \frac{2}{2/5} \approx 3.12$$

پاسخ مسئله‌ی ۲.

با توجه به اطلاعات صورت سوال، تسریع را به تقریب محاسبه می‌کنیم:

$$ClockCycle_1 = \frac{1}{ClockRate_1} = \frac{1}{4 \times 10^8} = 0.25ns \longrightarrow ClockCycle_2 = 0.45ns$$

$$SpeedUp = \frac{T_1}{T_2} = \frac{CPI_1 \times ClockCycle_1}{CPI_2 \times ClockCycle_2} = \frac{0.4 \times 4 + 0.3 \times 4 + 0.3 \times 10}{1} \times \frac{0.25}{0.45} \simeq 3/2$$

پاسخ مسئله‌ی ۳.

ترتیب اولیه دستورات به این صورت است:

۱	add	\$3, \$1,	\$5
۲	sub	\$2, \$1,	\$5
۳	lw	\$5, 0(\$3)	
۴	addi	\$4, \$5,	1
۵	add	\$5, \$4,	\$1

الف

در این بخش اجرای دستورات را در حالتی که هیچ *Forwarding* ای وجود نداشته باشد نشان می‌دهیم:

Instruction	Clock1	Clock2	Clock3	Clock4	Clock5	Clock6	Clock7	Clock8	Clock9	Clock10	Clock11	Clock12	Clock13	Clock14	Clock15
add \$3,\$1,\$5	F	D	X	M	W										
sub \$2,\$1,\$5		F	D	X	M	W									
lw \$5,0(\$3)			d*	F	D	X	M	W							
addi \$4,\$5,1					d*	d*	F	D	X	M	W				
add \$5,\$4,\$1								d*	d*	F	D	X	M	W	

ب

در این بخش اجرای دستورات را در حالتی که *Forwarding Full* وجود داشته باشد نشان می‌دهیم:

Instruction	Clock1	Clock2	Clock3	Clock4	Clock5	Clock6	Clock7	Clock8	Clock9	Clock10	Clock11	Clock12	Clock13	Clock14	Clock15
add \$3,\$1,\$5	F	D	X	M	W										
sub \$2,\$1,\$5		F	D	X	M	W									
lw \$5,0(\$3)			F	D	X	M	W								
addi \$4,\$5,1				d*	F	D	X	M	W						
add \$5,\$4,\$1						F	D	X	M	W					

ج

در این بخش اجرای دستورات را در حالتی که *Forwarding EX to EX* وجود داشته باشد نشان می‌دهیم:

Instruction	Clock1	Clock2	Clock3	Clock4	Clock5	Clock6	Clock7	Clock8	Clock9	Clock10	Clock11	Clock12	Clock13	Clock14	Clock15
add \$3,\$1,\$5	F	D	X	M	W										
sub \$2,\$1,\$5		F	D	X	M	W									
lw \$5,0(\$3)			F	D	X	M	W								
addi \$4,\$5,1				d*	d*	F	D	X	M	W					
add \$5,\$4,\$1						F	D	X	M	W					

پاسخ مسئله‌ی ۴.

ترتیب اولیه دستورات به این صورت است:

۱	I1:	lw	R1, 0(R2)		; R1 ← Memory[R2]
۲	I2:	addi	R1, R1, 1		; R1 ← R1+1
۳	I3:	sw	R1, 0(R2)		; Memory[R2] ← R1
۴	I4:	addi	R2, R2, 8		; R2 ← R2+8
۵	I5:	addi	R4, R4, -1		; R4 ← R4-1
۶	I6:	bne	R4, R0, I1		; branch if R4!=0

الف

باید جایگاه I_۵ را طوری تغییر بدهیم که حداقل ۲ مرحله زودتر از I_۶ اجرا شود.

۱	I1:	lw	R1, 0(R2)		; R1 ← Memory[R2]
۲	I5:	addi	R4, R4, -1		; R4 ← R4-1
۳	I2:	addi	R1, R1, 1		; R1 ← R1+1
۴	I3:	sw	R1, 0(R2)		; Memory[R2] ← R1
۵	I4:	addi	R2, R2, 8		; R2 ← R2+8
۶	I6:	bne	R4, R0, I1		; branch if R4!=0

ب

در این بخش ما دستوری را بعد از I_۶ قرار می‌دهیم که هر بار اجرا شود و ارتباطی با شرط پرش نداشته باشد.

۱	I1:	lw	R1, 0(R2)		; R1 ← Memory[R2]
۲	I5:	addi	R4, R4, -1		; R4 ← R4-1
۳	I2:	addi	R1, R1, 1		; R1 ← R1+1
۴	I3:	sw	R1, 0(R2)		; Memory[R2] ← R1
۵	I6:	bne	R4, R0, I1		; branch if R4!=0
۶	I4:	addi	R2, R2, 8		; R2 ← R2+8

ج

با توجه به خواسته سوال، جدول زیر را تشکیل می‌دهیم:

	Clock1	Clock2	Clock3	Clock4	Clock5	Clock6	Clock7	Clock8	Clock9	Clock10
I1	F	D	X	M	W					
I2		F	D	X	M	W				
I5			F	D	X	M	W			
I3				F	D	X	M	W		
I6					F	D	X	M	W	
I4						F	D	X	M	W

اگر فقط یک بار این حلقه اجرا شود، در مجموع ۱۰ کلاک زمان می‌برد، اما اگر چندین بار این حلقه تکرار شود، تقریباً به تعداد حلقه‌ها نیاز به کلاک داریم.