پاسخ مسئله‌ی ۱.

الف

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

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{CPI_۱ \times \frac{۱}{CR_۱}}{CPI_۲ \times \frac{۱}{CR_۲}} = \frac{\frac{۵}{۲/۵}}{\frac{۱}{۲}} = ۴$$

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

ب

EX to 1st Only	MEM to 1st only	Ex to 2nd Only	MEM to 2nd Only	EX to 1st and EX to 2nd
5%	20%	10%	10%	5%

فرض می‌کنیم درصد دستوراتی که وابستگی دارند مطابق جدول فوق است، پس برای هر نوع از دستورات تعداد چرخه تاخیر را محاسبه می‌کنیم و سپس با فرض این که هیچ $Forwarding$ ای نداریم، تسریع را حساب می‌کنیم:

$$\begin{cases} EX\ to\ ۱st\ Only = ۲DelayCycle \\ MEM\ to\ ۱st\ Only = ۲DelayCycle \\ EX\ to\ ۲nd\ Only = ۱DelayCycle \\ MEM\ to\ ۲nd\ Only = ۱DelayCycle \\ EX\ to\ ۱st = ۲DelayCycle \\ EX\ to\ ۲nd = ۲DelayCycle \end{cases}$$

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{\frac{CPI_۱}{CR_۱}}{\frac{CPI_۲}{CR_۲}} = \frac{\frac{۵}{۲/۵}}{\frac{۰/۰۵\times۳+۰/۲\times۳+۰/۱\times۲+۰/۱\times۲+۰/۰۵\times۳+۰/۵\times۱}{۲}} \simeq ۲/۲$$

ج

اگر داده‌ها را به صورت زودهنگام در اختیار داشته باشیم باید فقط برای وابستگی $MEM\ to\ ۱st\ only$ ۱ کلاک به تاخیر کنیم. پس:

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{\frac{CPI_۱}{CR_۱}}{\frac{CPI_۲}{CR_۲}} = \frac{\frac{۵}{۲/۵}}{\frac{۰/۲\times۲+۰/۸\times۱}{۲}} = \frac{۱۰}{۳} \simeq ۳/۳$$

د

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

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{\frac{CPI_۱}{CR_۱}}{\frac{CPI_۲}{CR_۲}} = \frac{\frac{۵}{۲/۵}}{\frac{۰/۲\times۲+۰/۱\times۲+۰/۷\times۱}{۲}} \simeq ۳/۱$$

۵

اگر همه پرش‌ها را انجام نشده فرض کنیم، آن‌وقت تسریع برابر است با:

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{\frac{CPI_۱}{CR_۱}}{\frac{CPI_۲}{CR_۲}} = \frac{۵}{۰/۲\times۲+۰/۱\times(۰/۲\times۱+۰/۸\times۲)+۰/۷\times۱} \times \frac{۲}{۴} \simeq ۳/۱۲$$

پاسخ مسئله‌ی ۲.

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

$$ClockCycle_۱ = \frac{۱}{ClockRate_۱} = \frac{۱}{۴\times۱۰^۹} = ۰/۲۵ns \longrightarrow ClockCycle_۲ = ۰/۴۵ns$$

$$SpeedUp = \frac{T_۱}{T_۲} = \frac{CPI_۱ \times ClockCycle_۱}{CPI_۲ \times ClockCycle_۲} = \frac{۰/۴\times۴+۰/۳\times۴+۰/۳\times۱۰}{۱} \times \frac{۰/۲۵}{۰/۴۵} \simeq ۳/۲۲$$

پاسخ مسئله‌ی ۳.

ترتیب اولیه دستورات به این صورت است:

```
add     $3, $1,     $5
sub     $2, $1,     $5
lw      $5, 0($3)
addi    $4, $5,     1
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:

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

الف

باید جایگاه I_5 را طوری تغییر بدهیم که حداقل ۲ مرحله زودتر از I_6 اجرا شود.

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

البته در بخش‌های بعدی مجبور هستیم که جای دستورات $I2, I5$ را تعویض کنیم چون در بخش بعدی فاصله آن کمتر از ۲ می‌شود.

ب

در این بخش ما دستوری را بعد از I_6 قرار می‌دهیم که هر بار اجرا شود و ارتباطی با شرط پرش نداشته باشد.

```
I1:     lw      R1, 0(R2)       ; R1 ← Memory[R2]
I5:     addi    R4, R4,     -1  ; R4 ← R4-1
I2:     addi    R1, R1,     1   ; R1 ← R1+1
I3:     sw      R1, 0(R2)       ; Memory[R2] ← R1
I6:     bne     R4, R0,     I1  ; branch if R4!=0
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، تقریبا به تعداد حلقه‌ها نیاز به کلاک داریم.