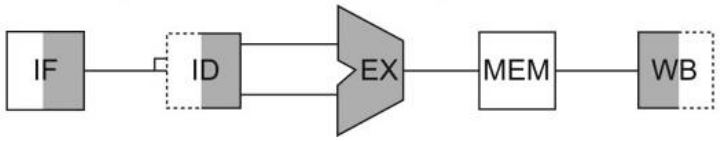
حل تمرین اختیاری چهار

۱- مسیردادهٔ ۵ مرحله‌ای مبتنی بر خطِ لولهٔ پردازندهٔ MIPS را مطابق شکل ۱ زیر در نظر بگیرید.



شکل ۱- بلوک دیاگرام مسیر دادهٔ ۵ مرحله‌ای مبتنی بر خط لوله در MIPS

نحوهٔ اجرای سری دستورات زیر را در جدول‌های زیر پر کنید. در صورت نیاز برای خانه‌های مهم جدول توضیحات اضافه نیز ارائه دهید. (سطر اول جدول برای نمونه پر شده است)

```
lw $2, 20($3)
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```

الف- بدون forwarding:

	1	2	3	4	5	6	7	8	9	10	11
lw	IF	ID	EX	MEM	WB						
and		IF	-	-	ID	EX	MEM	WB			
or					IF	ID	EX	MEM	WB		
add						IF	ID	EX	MEM	WB	
sw							IF	ID	EX	MEM	WB

پاسخ: دستور and برای اجرا به محتوای $2 نیاز دارد که در آخرین مرحله از اجرای دستور lw آماده می‌شود. بنابراین مرحله ID دستورِ and باید تا مرحلهٔ WB دستور lw به تاخیر بیفتد.

ب- با استفاده از forwarding:

	1	2	3	4	5	6	7	8	9	10	11
lw	IF	ID	EX	MEM	WB						
and		IF	-	ID	EX	MEM	WB				
or				IF	ID	EX	MEM	WB			
add					IF	ID	EX	MEM	WB		
sw						IF	ID	EX	MEM	WB	

پاسخ: وابستگی داده‌ای دستور and به lw با استفاده از forwarding برطرف می‌شود، اما چون داده بعد از مرحلهٔ MEM قابل استفاده است، دستور and باز هم به یک چرخه توقف نیاز خواهد داشت.

۲- مسیردادهٔ ۵ مرحله‌ای مبتنی بر خطِ لولهٔ پردازندهٔ MIPS را مطابق شکل ۱ در نظر بگیرید. فرض کنید برای رفع وابستگی داده از forwarding استفاده شده و نیز یک مدار اضافه برای برآورد شرط پرش و محاسبهٔ آدرس مقصد به مرحلهٔ ID اضافه شده است. روند اجرای سری دستورات زیر را بررسی کنید و تعداد کل چرخه‌های موردنیاز برای اجرای این دستورات را در هر یک از دو حالت زیر به‌دست آورید.

```
        addi $s0, $zero, 10
Loop:   addi $s0, $s0, -1
        bne $s0, $zero, Loop
        sub $t0, $t1, $t2
```

الف- بدون استفاده از branch prediction

ب- با فرض این که هر دستور پرشی به صورت پیش‌فرض انجام نمی‌شود.

	1	2	3	4	5	6	7	8	9	10
addi $s0, $zero, 10	IF	ID	EX	MEM	WB					
addi $s0, $s0, -1		IF	ID	EX	MEM	WB				
bne $s0, $zero, Loop			IF	-	ID	EX	MEM	WB		
sub $t0, $t1, $t2					IF					
addi $s0, $zero, 10						IF	ID	EX	MEM	WB

پاسخ: در هر دو حالت دور اول اجرای حلقه طبق جدول بالا انجام خواهد شد. وابستگی‌های دادهٔ دستورات دوم به اول و سوم به دوم به کمک مدارهای forwarding حل می‌شود. اما به خاطر وابستگی داده دستور سوم به دوم، آدرس مقصد و شرط پرش در مرحله ID آماده نمی‌شود و به یک چرخه توقف نیاز خواهیم داشت. در مواردی که حلقه اجرا خواهد شد (پرش انجام می‌شود) یک چرخه بین bne و addi فاصله می‌افتد. اما در حالت ب که پرش طبق پیش‌فرض انجام نمی‌شود، بار آخرِ اجرای حلقه هیچ فاصله‌ای بین اجرای دستور bne و sub وجود ندارد. بنابراین زمان اجرای کل دستورات در حالت الف و ب یک چرخهٔ ساعت با هم فرق دارد.

در بخشِ الف (بدونِ پیش‌بینیِ مقصدِ پرش) دستور اول یک بار و دستور آخر هم یک بار اجرا می‌شود. خود حلقه ۱۰ بار اجرا می‌شود و هر بار چهار دستور اجرا می‌شود (دو دستور addi و bne و دو حباب) پس در مجموع ۴۲ دستور برای اجرا داریم. در یک خط لولهٔ ۵ مرحله‌ای اجرای این ۳۲ دستور ۵+۴۱ یعنی ۴۶ چرخه ساعت طول می‌کشد. در بخش ب با توضیحی که در بند قبل دادیم زمان اجرا یک چرخه کمتر یعنی ۴۵ چرخه ساعت است.

این استدلال را به نوع دیگری هم می‌توانیم انجام بدهیم که به همین نتیجه می‌رسد:

در هر دو بخش بارِ اولِ اجرای حلقه از چرخهٔ ۲ شروع می‌شود. در بخش الف هر دور اجرای حلقه ۴ چرخه طول می‌کشد و بنابراین آخرین دستور در چرخهٔ ۴۱ شروع می‌شود و ۵ چرخه بعد یعنی در چرخهٔ ۴۶ کل برنامه اجرا شده است. در بخش ب آخرین دستور در چرخهٔ ۴۰ شروع می‌شود و کل برنامه در چرخهٔ ۴۵ تمام می‌شود.

۳- قطعه کد زیر طبق آنچه در جدول نشان داده شده است بر روی یک پردازنده مبتنی بر خط لوله اجرا می‌شود. مراحل مختلف این خط لوله عبارتند از:

F: Fetch
D: Decode
E: Execute
M: Memory
W: Write back

توجه کنید در هر دستور اولین عملوند سمت چپ مقصد (dst) و دو عملوند دیگر مبدا (src) هستند، برای مثال دستور "ADD A, B, C" به معنای A ← B+C است.

	Cycles	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	MUL R5, R6, R7	F	D	E1	E2	E3	E4	M	W										
1	ADD R4, R6, R7		F	D	E1	E2	E3	-	M	W									
2	ADD R5, R5, R6			F	D	-	-	E1	E2	E3	M	W							
3	MUL R4, R7, R7				F	-	-	D	E1	E2	E3	E4	M	W					
4	ADD R6, R7, R5							F	D	-	E1	E2	E3	M	W				
5	ADD R3, R0, R6								F	-	D	-	-	E1	E2	E3	M	W	
6	ADD R7, R1, R4										F	-	-	D	E1	E2	E3	M	W

الف- محاسبه نتیجهٔ هر عمل جمع یا ضرب چند چرخهٔ ساعت طول می‌کشد؟

ب- حداقل تعداد پورت‌های خواندن/نوشتن فایل ثباتی (register file) که در این معماری به کار می‌رود چند است؟ (توضیح دهید)

ج- آیا این معماری از forwarding استفاده می‌کند؟ توضیح دهید.

د- کد اسمبلی بالا را طوری تغییر دهید تا تعداد حباب‌های تزریق‌شده به خط لوله کمینه شود. شما مجاز به تغییر ترتیب و اضافه و کم کردن دستورات add و mul هستید اما دقت کنید پس از اجرای کد نباید نتیجه ذخیره‌شده در فایل ثبات با حالت فعلی متفاوت باشد.

پاسخ:

الف- اجرای جمع ۳ چرخه و اجرای ضرب ۴ چرخه طول می‌کشد.

ب- فایلِ ثبات حداقل دو پورت برای خواندن و یکی برای نوشتن نیاز دارد.

ج- اگر به توالیِ دستورات دقت کنیم، می‌بینیم که دستورِ خط ۲ برای اجرا به نتیجهٔ دستورِ خط ۰ نیاز دارد (R5). اگر forwarding نداشته باشیم این نتیجه در چرخهٔ ۸ قابل استفاده است، در حالی که طبقِ جدول این نتیجه در چرخهٔ ۷ در اختیار واحد اجرا قرار گرفته، یعنی مستقیم از خروجیِ E4 به ورودیِ E1 هدایت شده است، پس forwarding داریم. برای اطمینان می‌توانیم به دو دستورِ خط ۴ و ۲ هم توجه کنیم. دستورِ خطِ ۴ به نتیجهٔ دستور خطِ ۲ نیاز دارد که بدون forwarding در چرخهٔ ۱۱ آماده می‌شود اما در جدول می‌بینیم که نتیجه در چرخهٔ ۱۰ در اختیارِ E1 قرار گرفته است. همین‌طور نتیجهٔ اجرای دستور ۴ به جای چرخهٔ ۱۴ در چرخهٔ ۱۳ در اختیار دستورِ ۵ قرار گرفته است.

د- برای تغییر برنامه اول متوجه می‌بینیم که دستور خط ۱ کاملا اضافه است چون از نتیجهٔ آن هیچ جا استفاده نمی‌شود، بنابراین آن را حذف می‌کنیم و به برنامهٔ زیر می‌رسیم:

```
0    MUL R5, R6, R7
1    ADD R5, R5, R6
2    MUL R4, R7, R7
3    ADD R6, R7, R5
4    ADD R3, R0, R6
5    ADD R7, R1, R4
```

حالا کمکان دستور خط ۱ به دستور خط ۰ وابسته است. پس باید سعی کنیم هر کدام از دستورات بعدی را که نتیجهٔ اجرای برنامه را تغییر نمی‌دهند بین دو دستور خط ۰ و ۱ وارد کنیم.

دستور ۲ را می‌توانیم بیاوریم بالا. دستور ۳ را نمی‌توانیم پیش از دستور ۱ اجرا کنیم چون مقدار R6 را تغییر می‌دهد. دستور ۴ هم حتما باید بعد از دستور ۳ اجرا شود، پس آن را هم نمی‌توانیم جابه‌جا کنیم. دستور ۵ هم از نتیجهٔ دستور ۲ استفاده می‌کند و باید بعد از آن اجرا شود اما باید فاصله‌ای بین آنها باشد که وابستگی دادهٔ حاصل تاخیر چندانی ایجاد نکند. پس فعلا دستور ۲ را بین دو دستور ۰ و ۱ قرار می‌دهیم:

```
0    MUL R5, R6, R7
1    MUL R4, R7, R7
2    ADD R5, R5, R6
3    ADD R6, R7, R5
4    ADD R3, R0, R6
5    ADD R7, R1, R4
```

حالا دستورِ خطِ ۳ هم به دستورِ خط ۲ وابسته است و خوب است فاصله‌ای بین آنها ایجاد کنیم، اما هیچکدام از دستورات بعدی را نمی‌توانیم قبل از دستور ۳ اجرا کنیم.

از طرفی دو دستور ۳ و ۴ هم وابستگی دارند و می‌توانیم دستور ۵ را بین آنها قرار بدهیم که اجرای مجموعه دستورات کمی بهبود یابد..

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
MUL R5, R6, R7	F	D	E1	E2	E3	E4	M	W									
MUL R4, R7, R7		F	D	E1	E2	E3	E4	M	W								
ADD R5, R5, R6			F	D	-	-	E1	E2	E3	M	W						
ADD R6, R7, R5				F	-	-	D	-	-	E1	E2	E3	M	W			
ADD R7, R1, R4							F	-	-	D	E1	E2	E3	M	W		
ADD R3, R0, R6										F	D	-	E1	E2	E3	M	W