حل تمرین امتیازی **مهلت تحویل ساعت ۲۴ روز دوشنبه ۲۹ خرداد**

به موارد زیر توجه کنید:

۱- حتما نام و شماره دانشجویی خود را روی پاسخ‌نامه بنویسید.

۲- کل پاسخ تمرینات را در قالب یک فایل pdf با شماره دانشجویی خود نام‌گذاری کرده در سامانه CW بارگذاری کنید.

۳- این تمرین ۱۰۰ نمره دارد که معادل یک نمرهٔ امتیازی اضافه بر نمره کلی درس است.

۴- در صورت مشاهده هر گونه مشابهت نامتعارف هر دو (یا چند) نفر کل نمره این تمرین را از دست خواهند داد.

۱- (۱۵ نمره) توضیح دهید کد RTL زیر چه کاری انجام می‌دهند و سپس شماتیک سخت‌افزاری را رسم کنید که این کد را اجرا می‌کند.

```
C'.S:          R1 ← n,  R2 ← 0,  C ← 1
C.OR(R1):      R2 ← R2+1,  R1 ← shr(R1)
C.(OR(R1)'): R3 ← R2,  C ← 0
```

پاسخ:

می‌توانیم فرض کنیم C خروجی یک D-FF است و R1 و R2 ثبات هستند. این کد به ترتیب زیر عمل می‌کند:

اگر C=0 و S=1، مقدار n در R1 نوشته می‌شود، C برابر با یک می‌شود و مقدار R2 هم صفر می‌شود.

اگر C=1 باشد، تا زمانی که مقدار R1 غیرصفر باشد در هر کلاک یک واحد به R2 اضافه می‌شود و R1 یک بار به سمت راست شیفت می‌خورد. (بر دو تقسیم می‌شود)

اگر C=1، وقتی R1 صفر شود، محتوای R2 وارد R3 می‌شود و C دوباره صفر می‌شود.

این کد آنقدر R1 را به راست شیفت می‌دهد تا محتوای آن صفر شود و هر بار یک واحد به R2 اضافه می‌کند.

بنابراین یک به علاوهٔ جایگاه پرارزش‌ترین یک را در R1 برمی‌گرداند که معادل است با $\lfloor log_2^n \rfloor + 1$

۲- (۱۵ نمره) فرض کنید برای نمایش اعداد ممیز شناور ۱۶ بیتی از رابطهٔ زیر استفاده می‌کنیم.

$$[\left(\sum_{k=0}^{8} \overline{b_k}\, 2^{8-k}\right) - 2^8 + 1] \times 2^{(\Sigma_{k=9}^{15} b_k 2^{k-9})-64}$$

الف- کمترین و بیشترین عدد مثبت قابل نمایش در این ساختار را به دست آورید.

ب- کمترین و بیشترین عدد منفی قابل نمایش در این ساختار را به دست آورید.

ج- در این ساختار صفر را چطور نمایش می‌دهیم؟

د- آیا در این روش، هر عدد نمایش یکتایی دارد؟ توضیح دهید.

پاسخ:

الف- در این روشِ نمایش، یک عددِ اعشاری به شکل $X = F \times 2^{E-64}$ نمایش داده می‌شود. بیت‌های ۰ تا ۸ برای ساختِ F بیت‌های ۹ تا ۱۵ برای ساخت E استفاده می‌شوند.

با توجه به این که ۷ بیت برای نمایش E در اختیار داریم، E می‌تواند بین ۰ تا ۱۲۷ تغییر کند. توانِ دو E-64 است، بنابراین توانِ دو بین ۶۴- و ۶۳ تغییر می‌کند.

F را می‌توانیم به صورتِ زیر ساده کنیم:

$$F = \left(\sum_{k=0}^{8} \overline{b_k}\, 2^{8-k}\right) - 2^8 + 1 = \left(\sum_{k=0}^{8} (1-b_k)\, 2^{8-k}\right) - 2^8 + 1 = \left(\sum_{k=0}^{8} 2^{8-k}\right) - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right) - 2^8 + 1$$
$$= (2^9 - 1 - 2^8 + 1) - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right) = 2^8 - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right)$$

الف- برای نمایش کمترین عدد مثبت باید F کوچکترین مقدار مثبت ممکن باشد (۱) و توان باید منفی‌ترین مقدار ممکن (۶۴-)، باشد:

$$F = 2^8 - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right) = 1 \implies \sum_{k=0}^{8} b_k 2^{8-k} = 2^8 - 1 = (01111111) \implies b_0 = 0, b_1..b_8 = 1$$

و بیت‌های E هم همه صفر باشند: ۰۰۰۰۰۰۰۰۱۱۱۱۱۱۱۱۰ که معادلِ عدد 2^{-64} خواهد بود.

برای نمایش بزرگترین عدد مثبت F باید بیشترین مقدار را داشته باشد و توان هم باید بیشترین مقدار باشد:

$$F_{max} = 2^8 - 0 = 256 \implies X_{max} = 256 \times 2^{63} = 2^{71}$$

در نمایش این عدد همه بیت‌های بخش توان یک و همه بیت‌های بخش کسری صفر هستند.

ب- برای نمایش منفی‌ترین عدد، باید F منفی‌ترین مقدار را داشته باشد و توان بیشترین مقدار باشد:

$$F_{min} = 2^8 - (2^9 - 1) = 256 - 512 + 1 = -255 \implies X_{min} = -255 \times 2^{63} = -2^{71} + 1$$

در نمایش این عدد باید همه بیت‌های بخش کسری یک و همه بیت‌های توان هم یک باشد.

برای نمایش بزرگترین عدد منفی F باید کوچکترین مقدار منفی باشد (۱-) و توان هم منفی‌ترین مقدار ممکن:

$$F = 2^8 - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right) = -1 \implies \sum_{k=0}^{8} b_k 2^{8-k} = 2^8 + 1 \implies b_0 = b_8 = 1, b_i =_{i\neq 0,8} 0$$

که نمایشِ باینری عدد به صورت ۰۰۰۰۰۰۰۱۰۰۰۰۰۰۰۱ می‌شود (معادلِ عدد -2^{-64}

ج- برای نمایش صفر کافی است:

$$F = 2^8 - \left(\sum_{k=0}^{8} b_k 2^{8-k}\right) = 0 \implies \sum_{k=0}^{8} b_k 2^{8-k} = 2^8$$

و توان هر مقداری می‌تواند داشته باشد. بنابراین $(0)_{10} =$ ××××××× 000000001

د- در این روشِ نمایش هر عدد نمایشِ یکتایی ندارد. مثل صفر یا عدد یک را می‌توانیم به چند صورت نمایش بدهیم:

$(256 - 255) \times 2^0$
$(256 - 254) \times 2^{-1}$
...

۳- (۱۵ نمره) یک محک از سه برنامهٔ A، B و C تشکیل شده است و قرار است برای مقایسهٔ کارایی سه پردازنده به کار رود. جدول ۱ نشان‌دهنده درصد دستورات موجود در هر یک از سه برنامه A، B و C است. هم‌چنین CPI مربوط به هر یک از پردازنده‌های مختلف در جدول ۲ آمده است. نرخ ساعت هر یک از سه پردازنده عبارت است:

$\text{CPU1} = 1.2\ GHz$, $\text{CPU2} = 1.4\ GHz$, $\text{CPU3} = 1.3\ GHz$

این محک به پردازنده‌ای که بتواند سریع‌ترین زمان میانگین (هندسی) اجرا را ثبت کند ۱۰۰ امتیاز و به مابقی پردازنده‌ها امتیازی متناسب با سرعت اجرای آن‌ها نسبت به سریع‌ترین پردازنده تخصیص می‌دهد. با توجه به اطلاعات داده شده، امتیاز هر یک از پردازنده‌ها را به دست آورید. (تعداد دستورات سه برنامه A، B و C را یکسان در نظر بگیرید)

جدول ۱

	A	B	C
R-type	%12	%40	%33
I-type	%35	%43	%22
branch	%8	%4	%20
load/store	%45	%13	%25

جدول ۲

	CPU1	CPU2	CPU3
R-type	4	3	5
I-type	4	5	5
branch	2	3	1
load/store	5	4	3

پاسخ:

طبق جدول زیر CPU2 بهترین زمانِ اجرا را دارد:

	A on CPU1	B on CPU1	C on CPU1		A on CPU2	B on CPU2	C on CPU2		A on CPU3	B on CPU3	C on CPU3	
R-type	0.48	1.6	1.32		0.36	1.2	0.99		0.6	2	1.65	
I-type	1.4	1.72	0.88		1.75	2.15	1.1		1.75	2.15	1.1	
branch	0.16	0.08	0.4		0.24	0.12	0.6		0.08	0.04	0.2	
load/store	2.25	0.65	1.25		1.8	0.52	1		1.35	0.39	0.75	
	4.29	4.05	3.85		4.15	3.99	3.69		3.78	4.58	3.7	
	3.58	3.38	3.21	**3.38**	2.96	2.85	2.64	**2.81**	2.91	3.52	2.85	**3.08**
				83.17				**100**				**91.41**

۴- (۱۵ نمره) می‌خواهیم یک پردازنده بسازیم که هر دستور را در ۵ مرحله اجرا می‌کند. زمان مورد نیاز برای اجرای هر مرحله به شرح زیر است:

Fetch	Decode	Execute	Memory	Write back
200 ps	150 ps	250 ps	300 ps	100 ps

الف- اگر بخواهیم این پردازنده را به صورت تک‌چرخه‌ای بسازیم، حداقلِ زمان هر چرخهٔ ساعت باید چقدر باشد؟

ب- اگر بخواهیم دستورات را در یک خط لولهٔ ۵ مرحله‌ای اجرا کنیم، حداقلِ زمان هر چرخهٔ ساعت باید چقدر باشد؟

ج- فرض کنید ۲۵٪ دستوراتی که روی این پردازنده اجرا می‌شود از نوع load، ۱۵٪ از نوع store، ۳۰٪ از نوع R-type و بقیه از نوع پرش‌های شرطی هستند، در یک پیاده‌سازی تک‌چرخه‌ای از این پردازنده، چند درصد مواقع پردازنده بیکار است؟ (منظور این است که اجرای یک دستور تمام شده اما چون هنوز به لبهٔ بعدی ساعت نرسیدیم، پردازنده هیچ کاری انجام نمی‌دهد.)

د- با همان ترکیب دستوراتِ بندِ ج، در پیاده‌سازی خط لولهٔ ۵ مرحله‌ای، اجرای هر دستور به طور متوسط چقدر طول می‌کشد؟

ه- حال تصمیم می‌گیریم برای افزایش کارایی از معماری SuperScalar استفاده کنیم. به این منظور، دو مسیر داده را به صورت موازی قرار می‌دهیم تا پردازنده بتواند دو دستور را همزمان روی هر کدام از دو مسیر داده اجرا کند. حداکثر افزایش کارایی چقدر خواهد بود و در چه شرایطی به دست می‌آید.

پاسخ:

الف- به اندازهٔ جمعِ زمان همه مراحل، یعنی ۱۰۰۰ پیکوثانیه، معادل یک نانوثانیه

ب- به اندازهٔ زمان طولانی‌ترین مرحله، یعنی ۳۰۰ پیکوثانیه، معادل ۰٫۳ نانوثانیه

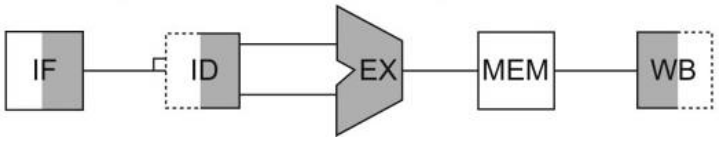
ج- طبق جدولِ زیر سهم بیکار بودن پردازنده برای هر نوع دستور را به دست می‌آوریم. می‌بینیم که در مجموع از هر ۱۰۰۰ پیکوثانیه ۲۲۵ پیکوثانیه پردازنده بیکار است که معادل ۲۲٫۵درصد است.

instr. type	load	store	R-type	Cond. branch	
rate	0.25	0.15	0.30	0.30	
idle time	0	100	300	400	
idle rate	0	15	90	120	**225**

د- اجرای متوسط هر دستور در پردازندهٔ خط لوله ۱۵۰۰ پیکوثانیه طول می‌کشد. اما اگر تعداد دستورات زیاد باشد، در هر ۳۰۰ پیکوثانیه یک دستور کارش را به پایان می‌رساند.

ه- کارایی حداکثر دو برابر می‌شود. البته به شرطی که ترتیبِ دستورات طوری باشد که همیشه هر دو مسیر داده در حال اجرای یک دستور باشد که البته عملا به خاطرِ انواعِ وابستگی‌های بین دستورات ممکن نیست.

۵- (۲۵ نمره) فرض کنید یک پردازندهٔ MIPS مبتنی بر خط لولهٔ ۵ مرحله‌ای مطابق شکل زیر در اختیار داریم.



قطعه برنامهٔ زیر را در نظر بگیرید.

```
lw $2,20($3)
and $12,$2,$5
or $13,$6,$2
```

الف- وابستگی‌های داده‌ای آن را مشخص کنید.

ب- با رسم جدول نشان دهید اجرای این قطعه در یک خط لوله بدون forwarding چند چرخه طول می‌کشد.

ج- با رسم جدول نشان دهید اجرای همین قطعه در یک خط لوله با امکان forwarding از مرحلهٔ MEM به EX چند چرخه طول می‌کشد. (منظور از این است که خروجی حافظه در چرخه بعدی وارد ALU شود. به عبارت دیگر، خروجی حافظه در همان چرخه روی ثباتی که بین لایهٔ ID و EX قرار دارد، نوشته شود.)

د- فرض کنید دستورِ اولِ این قطعه به جای lw، sw بود. یک بار دیگر به سه سوال بندهای الف تا ج پاسخ دهید. حال، این قطعه برنامه را در نظر بگیرید:

```
add $2,$3,$3
add $4,$2,$5
add $8,$6,$2
```

ه- وابستگی‌های داده‌ای آن را مشخص کنید.

و- با رسم جدول نشان دهید اجرای این قطعه در یک خط لوله بدون forwarding چند چرخه طول می‌کشد.

ز- آیا می‌توانیم با اضافه کردن امکان forwarding از مرحلهٔ EX به EX، این قطعه برنامه را بدون تعلیق (stall) اجرا کنیم؟ (منظور از این است که خروجی ALU در چرخه بعدی وارد ALU شود. به عبارت دیگر، خروجی ALU در همان چرخه روی ثباتی که بین دو لایهٔ ID و EX قرار دارد، نوشته شود.)

ح- اگر پاسخ شما به سوال بند قبل مثبت است، حساب کنید اجرای این قطعه چند چرخه طول می‌کشد و اگر پاسختان منفی است، توضیح دهید چه قابلیت دیگری باید به خط لوله اضافه کنید که بتوانیم برنامه را بدون تعلیق اجرا کنیم.

این بار قطعه برنامه زیر را در نظر بگیرید:

```
and $8,$2,$5
sw $2,0($8)
add $3,$2,$5
sw $3,0($9)
```

ط- وابستگی‌های داده‌ای آن را مشخص کنید.

ی- با رسم جدول نشان دهید اجرای این قطعه در یک خط لوله بدون forwarding چند چرخه طول می‌کشد.

ک- برای اجرای بدون تعلیق این قطعه برنامه به چه نوع forwarding نیاز داریم؟ توضیح دهید.

ل- فرض کنید دو دستورِ دوم و چهارم این قطعه به جای sw، lw باشند. بار دیگر به سه سوال قبل پاسخ دهید.

پاسخ:

الف- دستورِ دوم به خاطرِ مقدارِ ثبات $2 به دستور اول وابسته است.

ب- ۹ چرخه:

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

ج- ۸ چرخه

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

د- اگر دستور اول sw بود، هیچ وابستگی وجود نداشت و اجرای این سه خط روی هم ۷ چرخه طول می‌کشید.

	1	2	3	4	5	6	7	8	9
sw	IF	ID	EX	MEM	WB				
and		IF	ID	EX	MEM	WB			
or			IF	ID	EX	MEM	WB		

ه- به خاطرِ ثباتِ $2 دو دستورِ دوم و سوم به دستورِ اول وابسته است.

و- ۹ چرخه

	1	2	3	4	5	6	7	8	9
add $2,...	IF	ID	EX	MEM	WB				
add $4,...		-	-	IF	ID	EX	MEM	WB	
add $8,...					IF	ID	EX	MEM	WB

ز- خیر، فقط همین نوع forwarding کافی نیست، چون همچنان دستور سوم وقتی به مرحلهٔ ID می‌رسد، مقدار $2 را نیاز دارد و باید منتظر بماند تا مرحلهٔ WB دستور اول تمام شود.

ح- باید این امکان را هم داشته باشیم که مقدار ثبات را در صورت نیاز از مرحلهٔ MEM هم به ورودی ALU بدهیم.

ط- دستور دوم به خاطرثبات $8 به دستور اول و دستور چهارم به خاطر $3 به دستور سوم وابسته است.

ی- ۱۲ چرخه

	1	2	3	4	5	6	7	8	9	10	11	12
and $8,…	IF	ID	EX	MEM	WB							
sw $2,…		-	-	IF	ID	EX	MEM	WB				
add $3,…					IF	ID	EX	MEM	WB			
sw $3,…						-	-	IF	ID	EX	MEM	WB

ک- در هر دو مورد باید خروجی واحد EX در همان چرخه در اختیار دستور بعدی قرار بگیرد. درست است که در دستورِ آخر تا پیش از مرحلهٔ MEM نیازی به مقدار $3 ندارد، اما اگر زمانی که خروجی واحد EX در همان چرخه به دستور بعدی داده شود (وارد ثباتی شود که بین لایهٔ ID و EXE قرار دارد)، مشکل وابستگی دادهٔ دستور ۴ به ۳ هم برطرف می‌شود. اگر این مورد را درنظر نگیریم، برای حل وابستگی دستور ۴ به ۳ باید forwarding از مرحلهٔ EX به MEM داشته باشیم.

ل- اگر به جای دو دستور sw، دستور lw داشته باشیم، دستور ۲ به دستور ۱ وابسته است و دستور ۳ هم به دستور ۲ وابسته است. اجرای این چهار دستور، بدون مدارهای forwarding ۱۰ چرخه طول می‌کشد.

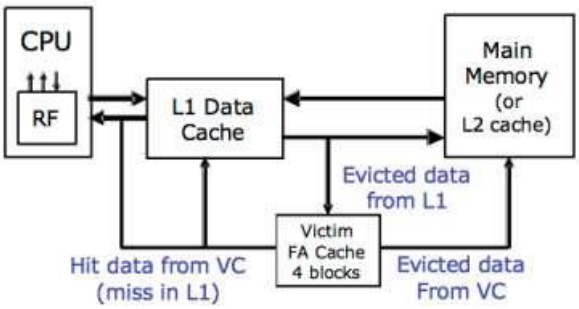
	1	2	3	4	5	6	7	8	9	10	11	12
and $8,…	IF	ID	EX	MEM	WB							
lw $2,…		-	-	IF	ID	EX	MEM	WB				
add $3,…							IF	ID	EX	MEM	WB	
lw $3,…								IF	ID	EX	MEM	WB

برای برطرف کردن آن‌ها نیاز به مدار forwarding از مرحلهٔ EX به مرحلهٔ EX دستور بعدی داریم و یک مدار forwarding از مرحلهٔ MEM به EXE و البته باز هم به یک چرخه تعلیق نیاز داریم.

	1	2	3	4	5	6	7	8	9
and $8,…	IF	ID	EX	MEM	WB				
lw $2,…		IF	ID	EX	MEM	WB			
add $3,…			-	IF	ID	EX	MEM	WB	
lw $3,…					IF	ID	EX	MEM	WB

۶- (۱۵ نمره) هر چند زمانِ دسترسی (access time) در حافظه‌های نهان با دسترسی مستقیم (direct map) کمتر از حافظه‌های شبه‌انجمنی (set-associative) است، تعدادِ فقدان‌های ناشی از تضاد (conflict misses) در آنها بیشتر است. برای کاهشِ این فقدان‌ها روشی پیشنهاد شده است[1] با عنوانِ به‌کارگیری حافظهٔ نهان قربانی (Victim cache). چگونگی کارکردِ حافظهٔ نهانِ قربانی در شکلِ زیر دیده می‌شود.

[1] Jouppi, N. P. (1990-05-01). Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. 17th Annual International Symposium on Computer Architecture, 1990. Proceedings. pp. 364–373. doi:10.1109/ISCA.1990.134547. ISBN 0-8186-2047-1.



در این ساختار دسترسی به هر داده طی مراحل زیر انجام می‌شود:

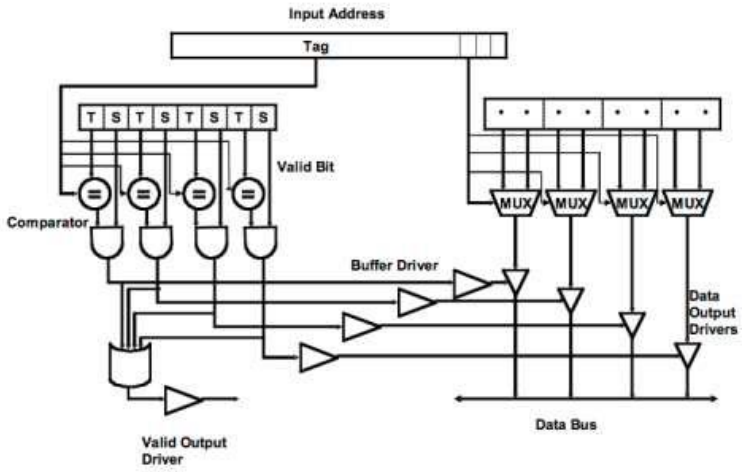
۱- حافظهٔ L1 بررسی می‌شود. اگر حاویِ دادهٔ موردنظر باشد، داده به پردازنده انتقال می‌یابد.

۲- اگر داده در L1 نباشد، حافظهٔ قربانی بررسی می‌شود. اگر حاویِ داده باشد، داده به L1 منتقل شده و از آنجا به پردازنده انتقال می‌یابد. اگر لازم باشد این داده روی داده‌ای قدیمی در L1 نوشته شود، دادهٔ جایگزین‌شده به حافظهٔ قربانی منتقل می‌شود، به این ترتیب که این داده در انتهای یک صف FIFO در حافظهٔ قربانی قرار می‌گیرد.

۳- اگر دادهٔ موردنظر نه در L1 باشد و نه در حافظهٔ قربانی، داده از حافظهٔ اصلی بازیابی شده و در L1 قرار می‌گیرد. این بار نیز اگر این داده به جای دادهٔ دیگری نوشته شود، دادهٔ قدیمی به انتهای صف حافظهٔ قربانی افزوده می‌شود و اگر صف پر بود، یک داده از ابتدای صف حذف می‌شود. اگر در مدتی که این داده در حافظه نبوده تغییری کرده باشد، نسخهٔ جدید آن روی حافظهٔ اصلی نوشته می‌شود.

توجه کنید داده‌ای که روی L1 ذخیره شده، روی حافظهٔ قربانی نیست و برعکس، داده‌ای که روی حافظهٔ قربانی ذخیره شده، روی L1 نیست، به عبارت دیگر L1 و حافظهٔ قربانی هیچ دادهٔ مشترکی ندارند.

ساختار داخلی حافظهٔ قربانی در شکل زیر دیده می‌شود. در این شکل، آدرس ورودی یک آدرس ۳۲ بیتی است که شماره بایت داده در حافظهٔ اصلی را نشان می‌دهد.



با توجه به شکل به این سوالات پاسخ دهید:

الف- این حافظه چند مجموعه دارد؟

ب- در هر مجموعه چند بایت داده ذخیره شده است؟

ج- هر واحدِ داده (کلمه) شامل چند بایت است؟ به عبارتِ دیگر، هر بار همزمان چند بایت را می‌توانیم از این حافظه بخوانیم؟

فرض کنید نرخ فقدان داده در L1 ۱۰٪ است و اگر داده‌ای در L1 نباشد، به احتمال ۰,۱۵ در حافظهٔ قربانی خواهد بود. همچنین فرض کنید بازیابی داده از حافظهٔ اصلی ۵۰ چرخه و بازیابی داده از حافظهٔ قربانی ۴ چرخه طول می‌کشد.

د- حساب کنید با به‌کارگیری حافظهٔ قربانی، متوسط زمان دسترسی به حافظه (AMAT) چقدر بهتر می‌شود؟ زمان جستجو در حافظهٔ قربانی را ناچیز فرض کنید.

پاسخ:

الف- از روی شکل دیده می‌شود که این حافظه کاملا انجمنی است و بنابراین فقط یک مجموعه دارد.

ب و ج- طبق شکل سه بیت سمت راست نقش آفست دارند، بنابراین در هر بلوک تنها ۸ بایت قرار دارد و چون فقط بیت سوم از سمتِ راست برای تعیین کلمه‌ای که باید روی گذرگاه قرار بگیرید استفاده می‌شود، پس هر کلمه ۴ بایت دارد.

د-

no victim cache: $AMAT_1 = L1\ hit\ time + 0.1 \times 50 = L1\ hit\ time + 5\ cycle$

with victim cache: $AMAT_2 = L1\ hit\ time + 0.1 \times 0.15 \times 4 + 0.1 \times 0.85 \times 50 = L1\ hit\ time + 4.31\ cycle$

$AMAT_1 - AMAT_2 = 0.69\ cycle$