HW-08
معماری کامپیوتر دکتر عطارزاده

سوال ۱)

الف) به صورت direct mapped و ۱۶ بلوک که هرکدام شامل یک word هستند.
16 بلوک است پس ۴ بیت فضای آدرس برای index ها داریم و به جز (۲ بیت داخل ورد) 26 بیت برای تگ ها باید در نظر بگیریم. خود دیتا هم که ۳۲ بیت است(۴بایت یا در واقع یک ورد)
(تگ ها ۲۶ بیتی هستندو در مواردی که جلوی آنها تگ کمتر از ۲۶ بیت است به خاطر وجود صفر قبل عدد است.)

0x03

آدرس باینری کلمه : 00011
برچسب تگ : 00000000000000000000000000
اندیس متناظر در حافظه نهان : 0011
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم .

0xb4

آدرس باینری کلمه : 10110100
برچسب تگ : 00001011
اندیس متناظر در حافظه نهان : 0100
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم .

0x2b

آدرس باینری کلمه : 000101011
برچسب تگ : 000010
اندیس متناظر در حافظه نهان : 1011
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

0x02

آدرس باینری کلمه: 00010
برچسب تگ: 00000000
اندیس متناظر در حافظه نهان : 0010
هیت یا میس بودن: اولین بار میس می‌شود ولی چون : compulsory است در نظر نمیگیریم .

0xbb

آدرس باینری کلمه: 10111011

برچسب تگ: 0001011
اندیس متناظر در حافظه نهان: 1011
هیت یا میس بودن: میس می‌شود چون قبلا ***0x2b*** را در همین اندیس کش ریخته ایم و conflict miss به وجود می آید. طبق LRU حال هم همین 0xbbرا جایگزین می‌کنیم .

0x58

آدرس باینری کلمه : 1011000
برچسب تگ : 000101
اندیس متناظر در حافظه نهان : 1000
هیت یا میس بودن: اولین بار میس می‌شود ولی چون compulsory است در نظر نمیگیریم.

0xbe

آدرس باینری کلمه : 10111110
برچسب تگ : 0001011
اندیس متناظر در حافظه نهان : 1110
هیت یا میس بودن: اولین بار میس می‌شود ولی چون : compulsory است در نظر نمیگیریم .

0x2e

آدرس باینری کلمه: 00101110
برچسب تگ: 0010
اندیس متناظر در حافظه نهان:1110
هیت یا میس بودن : میس می‌شود چون قبلا 0xbe را در همین اندیس کش ریخته ایم و conflict miss به وجود می آید. طبق LRU حال هم همین 0x2e را جایگزین می‌کنیم .

0xb5

آدرس باینری کلمه : 10110101
برچسب تگ : 0001011
اندیس متناظر در حافظه نهان : 0101
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

0x2c

آدرس باینری کلمه : 101100
برچسب تگ : 0000010
اندیس متناظر در حافظه نهان : 1100
هیت یا میس بودن: اولین بار میس می‌شود ولی چون : compulsory است در نظر نمیگیریم

0xb5

آدرس باینری: 10110101
برچسب تگ: 1011
اندیس متناظر در حافظه نهان: 0101

هیت یا میس بودن: هیت میشود چون در همین بلوک از قبل وجود داشته است.

0xfd

آدرس باینری کلمه : 11111101

برچسب تگ : 0001111

اندیس متناظر در حافظه نهان : 1101

هیت یا میس بودن: اولین بار میس می‌شود ولی چون compulsory است در نظر نمیگیریم .

برای به دست آوردن نرخ میس و هیت به شکل زیر عمل میکنیم:

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = \frac{2}{12} = 0.1\overline{6}$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = \frac{10}{12} = 0.8\overline{3} = 83\%$$

ب)چون ظرفیت همان است پس این بار ۸ بلوک خواهیم داشت . ۳ بیت برای ایندکس در نظر می‌گیریم و یک بیت برای blockOffset و ۲۷ بیت برای تگ خواهیم داشت.

در اینجا بر خلاف حالت قبل باید از spatiality هم بهره ببریم. به این شکل که هرگاه دیتای جدیدی را به کش اوردیم نزدیک ترین دیتای مربوط به آ را نیز به کش بیاوریم.

0x03

آدرس باینری کلمه : 00011

برچسب تگ : 00000000000000000000000

بلاک افست: 1

اندیس متناظر در حافظه نهان :001 1

هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

همچنین دیتایی که به خاطر spatiality به این بلوک وارد میشود را با رنگ آبی نشان داده ایم.

0xb4

آدرس باینری کلمه : 10110100

برچسب تگ : 00001011

بلاک آفست: 0

اندیس متناظر در حافظه نهان : 010

هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

0x2b

آدرس باینری کلمه : 000101011

برچسب تگ : 000010

بلاک آفست: 1
اندیس متناظر در حافظه نهان : 101
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

0x02

آدرس باینری کلمه: 00010
برچسب تگ: 00000000
بلاک آفست: 0
اندیس متناظر در حافظه نهان : 001
هیت یا میس بودن: این داده هیت میشود چون قبلا به خاطر نزدیکی مکانی با داده 0x03 در کش نوشته شده است.

0xbb

آدرس باینری کلمه: 10111011
برچسب تگ: 0001011
بلاک آفست: 1
اندیس متناظر در حافظه نهان: 101 5
وضعیت هیت یا میس خوردن: اینجا *conflict miss* داریم و داده میس میشود و xbb و نزدیک ترین داده اطراف آن که xbe است در ست پنجم جایگزین میشوند.
همچنین دیتایی که به خاطر spatiality به این بلوک وارد میشود را با رنگ آبی نشان داده ایم.

0x58

آدرس باینری کلمه : 1011000
برچسب تگ : 000101
بلاک آفست: 0
اندیس متناظر در حافظه نهان : 100
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.
همچنین دیتایی که به خاطر spatiality به این بلوک وارد میشود را با رنگ آبی نشان داده ایم.

0xbe

آدرس باینری کلمه : 10111110
برچسب تگ : 0001011
بلاک آفست: 0
اندیس متناظر در حافظه نهان : 111
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.

0x2e

آدرس باینری کلمه: 00101110
برچسب تگ: 0010
بلاک آفست: 0

اندیس متناظر در حافظه نهان:111
وضعیت هیت یا میس خوردن: اینجا *conflict miss* داریم و داده میس میشود و 0x2e و نزدیک ترین داده اطراف آن که 0x2c است در ست پنجم جایگزین میشوند.

0xb5

آدرس باینری کلمه : 10110101
برچسب تگ : 1011
بلاک آفست: 1
اندیس متناظر در حافظه نهان : 010
هیت یا میس خوردن: این داده هیت میشود چون قبلا به خاطر نزدیکی مکانی با داده 0xb4 در کش نوشته شده است.

0x2c

آدرس باینری کلمه : 101100
برچسب تگ : 0000010
بلاک آفست: 0
اندیس متناظر در حافظه نهان : 110
هیت یا میس بودن: اولین بار میس می‌شود ولی چون *compulsory* است در نظر نمیگیریم.
همچنین دیتایی که به خاطر spatiality به این بلوک وارد میشود را با رنگ آبی نشان داده ایم.

0xb5

آدرس باینری: 10110101
بلاک آفست: 1
برچسب تگ: 1011
اندیس متناظر در حافظه نهان: 010
هیت یا میس خوردن: این داده هیت میشود چون قبلا به خاطر نزدیکی مکانی با داده 0xb4 در کش نوشته شده است.

0xfd

آدرس باینری کلمه : 11111101
بلاک آفست: 1
برچسب تگ : 0001111
اندیس متناظر در حافظه نهان :110
وضعیت هیت یا میس خوردن: اینجا *conflict miss* داریم و داده میس میشود و 0x2e و نزدیک ترین داده اطراف آن که 0xbe است در ست پنجم جایگزین میشوند.
همچنین دیتایی که به خاطر spatiality به این بلوک وارد میشود را با رنگ آبی نشان داده ایم.
در این حالت تعداد کل میس ها ۳ عدد است.

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = \frac{3}{12} = 0.0.25 = 25\%$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = \frac{9}{12} = 0.75 = 75\%$$

ج) ابتدا می بینیم که برای آدرس دهی در این فضا به چند بیت نیاز داریم:

Capacity = 32 kilobytes = 2^{15}

N=2

b= 8 words

Word = 4 byte

B = C/b

به ۱۵ بیت نیاز داریم. با توجه به اینکه هر کلمه ۴ بایت است دو بیت سمت راست به byte Offset و در هر بلوک ۸ کلمه قرار میگیرد پس به سه بیت(لگاریتم هشت بر مبنای دو) برای مشخص کردن محل این کلمه ها در هر بلاک نیاز داریم.

Number of sets = log($2^{15-2-3-1}$)=9

حال بیت های مورد بحث شده را در آدرس دهی مشخص میکنیم:

byte offset = 2

Block offset = 3

Set = 9

Tag = 18

ابتدا ادرس ها را به شکل باینری مینویسیم:

Address = tag(18) set(9) blockoffset(3) byteoffset(2)

0x03 =00000000000000000 000000000 011 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست 0 و در یکی از راه ها قرار میگیرد. در کلمه ۳م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

0xb4 =00000000000000000 000010110 100 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست ۲۲ و در یکی از راه ها قرار میگیرد. در کلمه ۴م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

0x2b =00000000000000000 000000101 011 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست 5و در یکی از راه ها قرار میگیرد. در کلمه 3م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

هیت میشود چون به دلیل spatiality قبلا توسط 0x03 در این ست گذاشته شده است.

0xbb =00000000000000000 000010111 011 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست 23و در یکی از راه ها قرار میگیرد. در کلمه 3م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

0x58 =00000000000000000 000001011 000 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست 0 و در یکی از راه ها قرار میگیرد. در کلمه ۳م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

0xbe =00000000000000000 000010111 110 00

هیت میشود چون به دلیل spatiality قبلا توسط 0xbb در این ست گذاشته شده است.

0x2e =00000000000000000 000000101 110 00

هیت میشود چون به دلیل spatiality قبلا توسط 0x2b در این ست گذاشته شده است.

0xb5 =00000000000000000 000010110 101 00

هیت میشود چون به دلیل spatiality قبلا توسط 0xb4 در این ست گذاشته شده است

0x2c =00000000000000000 000000101 100 00

هیت میشود چون به دلیل spatiality قبلا توسط 0x2b در این ست گذاشته شده است

0xb5 =00000000000000000 000010110 101 00

هیت میشود چون به دلیل spatiality قبلا توسط 0xb4 در این ست گذاشته شده است

0xfd =00000000000000000 0000011111 101 00

اولین بار میس می‌شود ولی چون ***compulsory*** است در نظر نمیگیریم. در ست 31و در یکی از راه ها قرار میگیرد. در کلمه 5م آن راه ثبت مشود و ۷ داده نزدیک به خود را به علت نزدیکی مکانی در خانه های خالی آن راه قرار میدهد.

همانطور که مشاهده شد ما کانفلیکت میس نداریم و همانطور که صورت سوال خواسته بود اگر میس اجباری را معادل هیت بگیریم نرخ هیت 100 درصد و نرخ میس 0 درصد است.

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = \frac{0}{12} = 0 = 0\%$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = \frac{12}{12} = 1 = 100\%$$

ج-ا)با 4 برابر کردن N در واقع نگاشت 8 مسیره میشود. میدانیم که این عمل کانفلیکت میس ها را کاهش میدهد ولی همانطور که گفته شد هیچ کانفلیکت میس نداشتیم پس انتظار ایجاد تغییری نخواهیم داشت.
بیت های آدرس را مشخص میکنیم:

Capacity = 32 kilobytes = 2^{15}

N=8

b= 8 words

Word = 4 byte
B = C/b
Number of sets = log($2^{15-2-3-3}$)=7

حال بیت های مورد بحث شده را در آدرس دهی مشخص میکنیم:

byte offset = 2
Block offset = 3
Set = 7
Tag = 20

مشابه با قسمت ها قبل کش را پر میکنیم و از آن میخوانیم و میس و هیت ها را بررسی میکنیم. در این قسمت فقط نتیجه نهایی اورده شده است:

address	tag	set	blockoffset	state
0x03	000…000	0000000	011	hit
0xb4	000…000	0010110	100	hit
0x2b	000…000	0000101	011	hit
0x02	000…000	0000000	010	hit
0xbb	000…000	0010111	011	hit
0x58	000…000	0001011	000	hit
0xbe	000…000	0010111	110	hit
ox2e	000…000	0000101	110	hit
0xb5	000…000	0010110	101	hit
0x2c	000…000	0000101	100	hit
0xb5	000…000	0010110	101	hit
0xfd	000…000	0011111	101	hit

همانطور که انتظار میرفت همگی هیت شدند و نرخ میس ۰ است.

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = \frac{0}{12} = 0 = 0\%$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = \frac{12}{12} = 1 = 100\%$$

ج-ب) با اضافه کردن اندازه بلاک ها از نزدیکی مکانی می توانیم بهره بیشتری ببریم و در نتیجه میس های اجباری کمتر میشوند ولی از آنجایی که ظرفیت کش ما ثابت است تعداد ست های ما کم میشود و این خود باعث بروز کانفلیکت میس میشود.

Capacity = 32 kilobytes = 2^{15}
N=2
b= 16 words
Word = 4 byte
B = C/b
Number of sets = log($2^{15-2-1-4}$)=8
حال بیت های مورد بحث شده را در آدرس دهی مشخص میکنیم:

byte offset = 2
Block offset = 4
Set = 8
Tag = 18

خانه های همرنگ در جدول زیر مستعد اتفاق افتادن کانفلیکت میس هستند. خانه 0xb4 پس از وارد کش شدن ۱۵ کلمه اطراف خودش که نیز شامل 0xbb,0xbe,0xb5 نیز هست را کنار خودش ذخیره میکند. همین اتفاق برای خانه های سبز رنگ نیز می افتد.

address	tag	set	blockOffset	state
0x03	000…000	00000000	0011	hit
0xb4	000…000	00001011	0100	hit
0x2b	000…000	00000010	1011	hit
0x02	000…000	00000000	0010	hit
0xbb	000…000	00001011	1011	hit
0x58	000…000	00000101	1000	hit
0xbe	000…000	00001011	1110	hit
ox2e	000…000	00000010	1110	hit

0xb5	000…000	00001011	0101	hit
0x2c	000…000	00000010	1100	hit
0xb5	000…000	00001011	0101	hit
0xfd	000…000	00001111	1101	hit

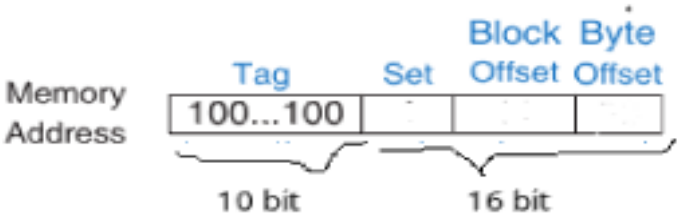
$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = \frac{0}{12} = 0 = 0\%$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = \frac{12}{12} = 1 = 100\%$$

سوال ۲) اول محاسبه میکنیم که برای آدرس دهی این فضا به چند بیت نیاز داریم.

64 megabyte = 64 * 1024 * 1024 byte
log (64 megabyte) = 26

برای آدرس دهی در این حافظه به ۲۶ بیت نیاز داریم.
طبق صورت سوال ۱۰ بیت به تگ اختصاص دارد و ۱۶ بیت دیگر به set و byteoffset , و block offset تعلق دارند.



میدانیم که n=4 است.
پس حافظه این کش به شکل زیر خواهد بود:
تعداد ست ها(دو به توان تعداد بیت هایی که به set تعلق دارد) * تعداد بیت های مورد نیاز برای ذخیره کردن دیتا (ذو به توان byte offset + block offset) * تعداد way ها در هر ست:

$$2^{set} * 2^{BlockOffset+ByteOffset} = 2^{16} * 4 = 2^{18} = 256KB$$

سوال ۳

با توجه به اینکه اندازه صفحه برابر 8KiB است، برای جستجوی مقدار یک آدرس (مثلا در یک کش VIVT)، نیاز به ۱۳ بیت برای index داریم (یعنی از روی ۱۳ بیت اول داده را در کش پیدا می‌کنیم) و بقیه آدرس به عنوان tag استفاده می‌شود. در کش توصیف شده، با توجه به این که اندازه هر بلاک برابر $4 * 64bits = 16\,bytes$ است و اندازه کل کش نیز 16KiB است، در نتیجه $2^{10} = 1024$ بلاک در مجموع داریم. برای دسترسی به این بلاک‌ها، نیاز به ۱۰ بیت index به علاوه‌ی ۴ بیت block offset داریم (اندازه هر بلاک $2^4\,bytes$ است). پس برای جستجوی داده در چنین کشی، نیاز به ۱۴ بیت داریم. اما virtual index های ما ۱۳ بیتی هستند و در نتیجه برای جستجوی داده در کش، به اندازه کافی بیت نداریم. پس چنین کشی مناسب نخواهد بود.

برای حل مشکل می‌توان تعداد way های کش را زیاد کرد. با فرض ثابت ماندن ظرفیت کش، با هر بار دو برابر کردن way های کش، تعداد بیت‌های index یکی کم می‌شود.

سوال ۴ - الف)

کد اسمبلی معادل کد C داده شده به صورت زیر است.

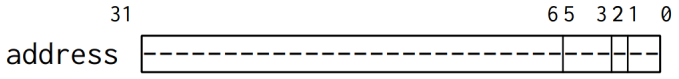
```
.data
a: .zero 16*8
b: .zero 8*16

.text
li t0, 0 # i loop variable
li t1, 8

la s0, a # load address at a
la s1, b # load address at b

for1:
    bge t0, t1, endfor1
    li t2, 0 # j loop variable
    li t3, 16
    for2:
        bge t2, t3, endfor2
        andi t4, t2, 1
        bne t4, zero, endif

        # &(b[i][j]) = b + (i * 8) + j
        slli t5, t0, 4
        add t5, t5, t2
        add t5, t5, s1

        lw t6, 0(t5) # t6 = b[i][j]
        mul t6, t6, t6 # t6 = b[i][j] * b[i][j]

        # &(a[j][i]) = a + (j * 16) + i
        slli t5, t2, 4 #t5 = j * 16
        add t5, t5, t0 #t5 = t5 + 0 = (j * 16) + i
        add t5, t5, s0 #t5 = t5 + s0 = (j * 16) + i + a

        sw t6, 0(t5) # a[j][i] = t6 = b[i][j] * b[i][j]

        endif:
        addi t2, t2, 1
        j for2
    endfor2:

    addi t0, t0, 1
    j for1
endfor1:
    nop
```

برای تعریف آرایه، از .data استفاده شده است که یک دستور اسمبلر (assembler directive) است.
همچنین برای ذخیره آدرس شروع دو آرایه موجود در سوال، از دستور la (به معنی load address) استفاده شده است (خط ۹ و ۱۰)؛ که یک label در کد را گرفته و آدرس آن label را در رجیستر مقصد ذخیره می‌کند.

سوال ۴ - ب)

نتیجه شبیه‌سازی در هر حالت به صورت زیر است.

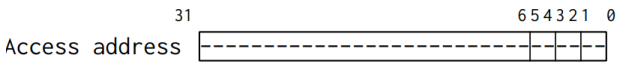
- اندازه بلوک ۲ کلمه: در این حالت باید ۸ بلوک داشته باشیم تا در مجموع ۱۶ کلمه ظرفیت داشته باشیم.



Index	V	D	Tag	Block 0	Block 1
0	1	1	0x00400003	0x00000000	0x00000000
1	1	0	0x00400003	0x00000000	0x00000000
2	1	0	0x00400003	0x00000000	0x00000000
3	1	0	0x00400003	0x00000000	0x00000000
4	1	1	0x00400003	0x00000000	0x00000000
5	1	0	0x00400003	0x00000000	0x00000000
6	1	0	0x00400003	0x00000000	0x00000000
7	1	0	0x00400003	0x00000000	0x00000000

در این حالت، hit rate گزارش شده برابر ۳۵٫۱۶٪ خواهد بود.

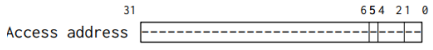
- اندازه بلوک ۴ کلمه: در این حالت، باید ۴ بلوک داشته باشیم.



Index	V	D	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000
1	1	0	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000
2	1	1	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000
3	1	0	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000

در این حالت، hit rate گزارش شده (دوباره) برابر ۳۹٫۸۴٪ خواهد بود.

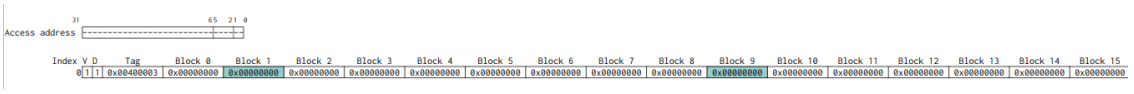
- اندازه بلوک ۸ کلمه: در این حالت، باید ۲ بلوک داشته باشیم.



Index	V	D	Tag	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
0	1	1	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
1	1	1	0x00400003	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

در این حالت، hit rate گزارش شده برابر ۳۶٫۷۲٪ است.

- اندازه بلوک ۱۶ کلمه: در این حالت، باید تنها ۱ بلوک داشته باشیم!

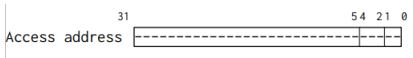


در این حالت، hit rate گزارش شده برابر ۲۵٪ خواهد بود.

سوال ۴ - ج)

نتیجه شبیه‌سازی به صورت زیر است.

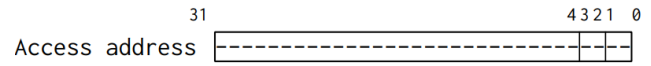
- حافظه نهان ۴ مسیره: اندازه بلوک باید برابر ۸ کلمه باشد تا در مجموع ۳۲ کلمه داشته باشیم.



Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
0	1	1	2	0x00800005	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	3	0x00800004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	1	0x00800006	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	0	0x00800007	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

در این حالت، hit rate گزارش شده برابر ۵۵٫۴۷٪ است.

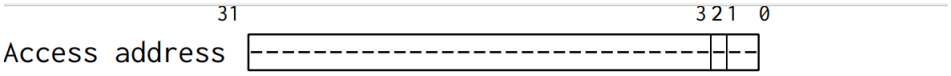
- حافظه نهان ۸ مسیره: اندازه بلوک باید برابر ۴ کلمه باشد.



Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	2	0x0100000c	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	0	0x0100000e	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	5	0x01000006	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	6	0x01000004	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	4	0x01000008	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	3	0x0100000a	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	7	0x01000002	0x00000000	0x00000000	0x00000000	0x00000000
	1	0	1	0x0100000f	0x00000000	0x00000000	0x00000000	0x00000000

در این حالت، hit rate گزارش شده برابر ۵۲٫۳۴٪ است.

- حافظه نهان ۱۶ مسیره: اندازه بلوک باید برابر ۲ کلمه باشد.



Index	V	D	LRU	Tag	Block 0	Block 1
0	1	1	4	0x02000010	0x00000000	0x00000000
	1	1	9	0x02000000	0x00000000	0x00000000
	1	1	8	0x02000004	0x00000000	0x00000000
	1	1	7	0x02000008	0x00000000	0x00000000
	1	1	5	0x0200000c	0x00000000	0x00000000
	1	0	11	0x0200001b	0x00000000	0x00000000
	1	1	3	0x02000014	0x00000000	0x00000000
	1	1	2	0x02000018	0x00000000	0x00000000
	1	1	0	0x0200001c	0x00000000	0x00000000
	1	0	10	0x0200001d	0x00000000	0x00000000
	1	0	6	0x0200001e	0x00000000	0x00000000
	1	0	1	0x0200001f	0x00000000	0x00000000
	1	0	15	0x02000016	0x00000000	0x00000000
	1	0	14	0x02000017	0x00000000	0x00000000
	1	0	13	0x02000019	0x00000000	0x00000000
	1	0	12	0x0200001a	0x00000000	0x00000000

در این حالت، hit rate گزارش شده برابر ۸۴٫۳۸٪ است.