**1.** ابتدا تعداد بیت های آدرس را مشخص میکنیم. از آنچا که حافظه 16MB و Byte-Addressable است، پس آدرس ما بیتی است.

**آ)** direct mapped:

*2-way associative :*

**ب)** *کافیست با فرمول محاسبه کنیم (تعداد کلاک ها در Hit Time کش سطح اول را همان CPI پایه در نظر میگیریم)*

**2. آ)** *فرض* میکنیم که سیستم ما Byte Addressable است.از آنجایی که پس هر بلاک بایت است که معادل 8 کلمه 32 بیتی است.

**ب)** مجددا از آنجایی که و 𝑎𝑠𝑠𝑜𝑐𝑖𝑎𝑡𝑖𝑣𝑖𝑡𝑦 × 𝑠𝑒𝑡 𝑐𝑜𝑢𝑛𝑡𝑠 = 𝑐𝑎𝑐ℎ𝑒 𝑒𝑛𝑡𝑟𝑖𝑒𝑠 پس درایه داریم.

**ج)** در هر خط، یک بلاک داده معادل ×832بیت داده داریم. 22 بیت tag و 1بیت validنیز در هر cache entry داریم. (میتوان dirty bit, status bit و.. را نیز به همین منوال در نظر گرفت) پس در کل:

**د)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Address | Binary | Tag bits[2:0] | Index bits | Offset bits | Set number (index in demical) | Miss/Hit |
| 0 | 000000000000 | 000 | 00000 | 00000 | 0 | Miss |
| 4 | 000000000100 | 000 | 00000 | 00100 | 0 | Hit |
| 16 | 000000010000 | 000 | 00000 | 10000 | 0 | Hit |
| 132 | 000010000100 | 000 | 00100 | 00100 | 4 | Miss |
| 232 | 000011101000 | 000 | 00111 | 01000 | 7 | Miss |
| 160 | 000010100000 | 000 | 00101 | 00000 | 5 | Miss |
| 1024 | 010000000000 | 001 | 00000 | 00000 | 0 | Miss |
| 30 | 000000011110 | 000 | 00000 | 11110 | 0 | Miss |
| 140 | 000010001100 | 000 | 00100 | 01100 | 4 | Hit |
| 3100 | 110000011100 | 011 | 00000 | 11100 | 0 | Miss |
| 180 | 000010110100 | 000 | 00101 | 10100 | 5 | Hit |
| 2180 | 100010000100 | 010 | 00100 | 00100 | 4 | Miss |

**ه)** نرخ برخورد معادل تعداد Hit ها به کل درخواست ها برابر است.

**3.** ابتدا مانند سوالات قبل بیت هارا محاسبه میکنیم. بیت برای آفست، برای ایندکس و بیت برای برچسب میخواهیم. حال کافیست با منطق little endian هر بایت اشاره شده را پشت باید بعدیش بگذاریم تا به کلمه اشاره شده برسیم.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Tag bits | Index bits | Offset bits | Set number  (index in demical) | Hit / Miss |
| 0x0000 | 00000000 | 000 | 00000 | 0 | Miss |
| 0x4A32 | 01001010 | 001 | 10010 | 1 | Miss |
| 0x0002 | 00000000 | 000 | 00010 | 0 | Hit |
| 0xB830 | 10111000 | 001 | 10000 | 1 | Miss |
| 0x0004 | 00000000 | 000 | 00100 | 0 | Hit |
| 0xB832 | 10111000 | 001 | 10010 | 1 | Hit |
| 0x0006 | 00000110 | 000 | 00110 | 0 | Hit |
| 0xB834 | 10111000 | 001 | 10100 | 1 | Hit |
| 0x0008 | 00000000 | 000 | 01000 | 0 | Hit |
| 0xB822 | 10111000 | 001 | 00010 | 1 | Hit |
| 0x000A | 00000000 | 000 | 01010 | 0 | Hit |
| 0x4A32 | 01001010 | 001 | 10010 | 1 | Miss |
| 0x000C | 00000000 | 000 | 01100 | 0 | Hit |
| 0x311A | 00110001 | 000 | 11010 | 0 | Miss |
| 0x000E | 00000000 | 000 | 01110 | 0 | Miss |
| 0x3142 | 00110001 | 010 | 00010 | 2 | Miss |

نرخ برخورد معادل تعداد Hit ها به کل درخواست ها و برابر است.

**4. آ)** هرکدام از 6 دستور درون حلقه باید از حافظه fetch شوند و دو دستور lb وsb نیز با حافظه کار میکنند. پس سر جمع در هر اجرا 8 بار کار با حافظه داریم.

**ب)** این بار تنها 3 دستور ابتدایی حلقه اجرا میشوند که نیازمند 3 بار واکشی دستور (فرض کردیم بار اول است که این حلقه اجرا میشوند و نیاز به حافظه اصلی داریم) و 2 بار کار با حافظه توسط lb, sb هستند. پس سر جمع 5 دسترسی به حافظه دارد.

طبق صورت سوال هر بلوک 1 کلمه یا معادلا در زبان میپس 4 بایت است. بنابراین حافظه نهان ما 32 تا set دارد که برای نمایش آنها به 5 بیت برای index نیاز داریم. همچنین چون هر بلوک 4 بایت است نیازی به 2 بیت offset داریم.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I/D | Address | Index(5bits) | Offset(2bits) | Set number |
| lb $t2, 0($t0) | 0x0000 | 00000 | 00 | 0 |
| sb $t2, 0($t1) | 0x0004 | 00001 | 00 | 1 |
| beq $t2, $zero, Exit | 0x0008 | 00010 | 00 | 2 |
| addiu $t0, $t0, 1 | 0x000C | 00011 | 00 | 3 |
| addiu $t1, $t1, 1 | 0x0010 | 00100 | 00 | 4 |
| j StrCpy | 0x0014 | 00101 | 00 | 5 |
| T0 | 0x0080-0x0083 | *00000* | 00-11 | 0 |
| T1 | 0x0100-0x0103 | *00000* | 00-11 | 0 |

**ج)** اگر منظور از حافظه را مجموعه ی حافظه اصلی و حافظه نهان در نظر بگیریم 3\*8+5 برابر با 29 جواب ماست.

اما اگر منظور از حافظه را حافظه اصلی سیستم در نظر بگیریم. درین صورت در اولین بار اجرای حلقه 6 دستور ما درون کش قرار میگیریند و بجز دستور اول در حلقه های بعدی نیازی به واکشی بقیه دستورات نیست پس حلقه های دیگر 2+1 دسترسی به حافظه دارند. پس سر جمع 8 + 3\*3= 17 دسترسی به حافظه داریم.

**د)** طبق فرض در cacheاز سیاست no-write-allocateاستفاده میکنیم. نوشتن در حافظه را جزء دسترسی ها به کش حساب نمیکنیم (پس متناسب با ج در کل 29-4=25 دسترسی به کش داریم) . بنابر این در حلقه اول 7 بار میس (6 دستوری + یک دیتا) و در 3 حلقه بعدی هرکدام 2 بار میس (دستور اول + یک دیتا) داریم. پس در کل 13 میس داریم و:

**ه)** هر setکش دارای دو بلوک 4بایتی است. بنابراین دستور lbو داده حاصل از خواندن از مموری میتوانند همزمان در کش قرار بگیرند. در نتیجه صرفا در اولین بار اجرای حلقه Missمیخوریم و در دفعات دیگر اجرای حلقه فقط Hitمیخوریم:

و) این بار دیگر دستور نوشتن از حافظه جز دسترسی های کش محسوب میشود. پس این بار کل دسترسی های ما به مجموعه حافظه برابر دسترسی های ما به کش است که در ابتدای ج برابر با 29 (25+4) محاسبه اش کردم. حال این بار به میس های قبلی، نوشتن در حافظه نیز اضافه میشود. پس :

**5. آ)**

* *تعداد بلاک ها در حافظه:*
* *تعداد Block offset:*
* *اندازه Tag:*

**ب)** *فرض میکنیم به صورت Fully associative است. درین صورت داریم:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Tag bits | Offset bits | Cache status | Hit / Miss |
| 110001 | 1100 | 01 | C[0]=[110000:110011] | Miss |
| 100111 | 1001 | 11 | C[1]=[100100:100111] | Miss |
| 001111 | 0011 | 11 | C[2]=[001100:001111] | Miss |
| 001100 | 0011 | 00 | C[2] | Hit |
| 010001 | 0100 | 01 | C[3]=[010000:010011] | Miss |
| 110010 | 1100 | 10 | C[0] | Hit |
| 100101 | 1001 | 01 | C[1] | Hit |
| 001110 | 0011 | 10 | C[2] | Hit |
| 100001 | 1000 | 01 | C[3]=[100000:100011] | Miss |
| 110001 | 1100 | 01 | C[0] | Hit |

**ج)** *مشابه بخش ب سوال اول عمل میکنیم. میدانیم در pipline از کلاک پنجم به بعد، درصورتی که با خطا مواجه نشویم در هر کلاک یک دستور کامل میشود. دستوراتی که از حافظه استفاده میکنند، load, store هستند که به طور میانگین و با محاسبه خطا تعداد کلاک هایشان از فرمول زیر بدست می آید.*

*بنا برین این دو دستوربه طور میانگین 2 کلاک اضافه تر میخواهد و 2 کلاک stall میخوریم. پس داریم:*