16.482 / 16.561: Computer Architecture and Design

Spring 2015

Homework #7 Solution

1. *(50 points) For each of the following memory hierarchies, calculate the average memory access time. If you end up with a fractional number of cycles, round up—there isn't much you can do (besides read/write the register file) in half a cycle!*

a. *The cache takes 1 cycle to access and has a 5% miss rate, main memory takes 200 cycles to access and has an 8% miss rate, and the disk takes 30,000 cycles to access.*

Solution: Remember: AMAT = (hit time) + (miss rate) × (miss penalty)
where the miss penalty is simply the AMAT for the next level of the memory hierarchy. Therefore:

$$\begin{aligned} AMAT &= 1 + (.05)(AMAT_{main\ memory}) \\ &= 1 + (.05)(200 + (.08)(AMAT_{disk})) \\ &= 1 + (.05)(200 + (.08)(30{,}000)) \\ &= 1 + (.05)(200 + 2400) = 1 + (.05)(2600) = 1 + 130 = 131 \text{ cycles} \end{aligned}$$

b. *The cache takes 3 cycles to access and has a 92% hit rate, main memory takes 400 cycles to access and has a 98% hit rate, and the disk takes 55,000 cycles to access.*

Solution: Same idea as part (a), but we're given hit rates and have to derive miss rates; remember that (miss rate) = 1 – (hit rate):

$$\begin{aligned} AMAT &= 3 + (.08)(AMAT_{main\ memory}) \\ &= 3 + (.08)(400 + (.02)(AMAT_{disk})) \\ &= 3 + (.08)(400 + (.02)(55{,}000)) \\ &= 3 + (.08)(400 + 1100) = 3 + (.08)(1500) = 3 + 120 = 123 \text{ cycles} \end{aligned}$$

c. *This problem deals with a multi-level cache, as discussed in class. The cache levels are listed in terms of their order in the memory hierarchy—an access initially goes to the level 1 (L1) cache. If there is a miss in the L1 cache, you then check the level 2 (L2) cache, then the level 3 (L3) cache, and then main memory.*

The L1 cache takes 1 cycle to access, with a 96% hit rate. The L2 cache takes 25 cycles on each access and has a 95% hit rate. The L3 cache takes 80 cycles to access and has a 98% hit rate. Main memory takes 600 cycles to access, with an 88% hit rate, while the disk takes 50,000 cycles to access.

Solution: Again, we're essentially doing the same calculation; there's just 5 levels in the memory hierarchy, rather than the 3 we're used to:

$$
\begin{aligned}
AMAT &= 1 + (.04)(AMAT_{L2\ cache}) \\
&= 1 + (.04)(25 + (.05)(AMAT_{L3\ cache})) \\
&= 1 + (.04)(25 + (.05)(80 + (.02)(AMAT_{main\ memory}))) \\
&= 1 + (.04)(25 + (.05)(80 + (.02)(600 + (.12)(AMAT_{disk})))) \\
&= 1 + (.04)(25 + (.05)(80 + (.02)(600 + (.12)(50{,}000)))) \\
&= 1 + (.04)(25 + (.05)(80 + (.02)(600 + 600))) \\
&= 1 + (.04)(25 + (.05)(80 + (.02)(6600))) \\
&= 1 + (.04)(25 + (.05)(80 + 132)) \\
&= 1 + (.04)(25 + (.05)(212)) \\
&= 1 + (.04)(25 + 10.6) = 1 + (.04)(35.6) = 1 + 1.424 = 2.424 \approx 3 \text{ cycles}
\end{aligned}
$$

2. *(50 points) You are given a system which has a 16-byte, write-back cache with 4-byte blocks. The cache is direct mapped.*

a. *(10 points) If each address uses 8 bits, what size are the offset, index, and tag?*

Solution: Since the blocks are $4 = 2^2$ bytes, the offset is **2 bits**.
The cache contains 16 / (4 * 1) = 4 lines, so the index is **2 bits**.
The tag is 8 – 2 – 2 = **4 bits.**

b. *(40 points) Assume the initial memory state shown below for the first 16 bytes and last 16 bytes of memory (note: all addresses are listed in decimal):*

NOTE: SEE ACTUAL ASSIGNMENT FOR MEMORY CONTENTS

For each access in the sequence listed below, show the cache state, indicate what register (if any) changes, and indicate if any memory blocks are written back and if so, what addresses and values are written. The cache state should carry over from one access to the next. As above, assume 8-bit addresses. Also, assume the cache is initially empty.

Solution: The table below shows the effects of each access; note that changes made to the cache for each access are shown in bold.

Access	Modified register	Cache state							Modified mem. block
		V	D	Tag	Data				
`lb $t0,3($zero)`	$t0 = 3	**1**	**0**	**0000**	**20**	**8**	**27**	**3**	None
		0							
		0							
		0							
`sb $t0,1($zero)`	None	1	**1**	0000	20	**3**	27	3	None
		0							
		0							
		0							
`lb $t1, 241($zero)`	$t1 = 67	**1**	**0**	**1111**	**15**	**67**	**78**	**19**	Bytes 0-3 = [20 3 27 3]
		0							
		0							
		0							

<table>
<tr><td rowspan="4"><code>sb $t1, 0($zero)</code></td><td rowspan="4">None</td><td>1</td><td>1</td><td>0000</td><td>67</td><td>3</td><td>27</td><td>3</td><td rowspan="4">None</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4"><code>lb $t0,
12($zero)</code></td><td rowspan="4">$t0 = 126</td><td>1</td><td>1</td><td>0000</td><td>67</td><td>3</td><td>27</td><td>3</td><td rowspan="4">None</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
<tr><td rowspan="4"><code>sb $t1,
241($zero)</code></td><td rowspan="4">None</td><td>1</td><td>1</td><td>1111</td><td>15</td><td>67</td><td>78</td><td>19</td><td rowspan="4">Bytes 0-3 =
[67 3 27 3]</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
<tr><td rowspan="4"><code>sb $t0,
10($zero)</code></td><td rowspan="4">None</td><td>1</td><td>1</td><td>1111</td><td>15</td><td>67</td><td>78</td><td>19</td><td rowspan="4">None</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td>0000</td><td>110</td><td>72</td><td>126</td><td>127</td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
<tr><td rowspan="4"><code>lb $t1,
251($zero)</code></td><td rowspan="4">$t1 = 93</td><td>1</td><td>1</td><td>1111</td><td>15</td><td>67</td><td>78</td><td>19</td><td rowspan="4">Bytes 8-11 =
[110 72 126
127]</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td>1111</td><td>101</td><td>71</td><td>89</td><td>93</td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
<tr><td rowspan="4"><code>lb $t3,
248($zero)</code></td><td rowspan="4">$t3 = 101</td><td>1</td><td>1</td><td>1111</td><td>15</td><td>67</td><td>78</td><td>19</td><td rowspan="4">None</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td>1111</td><td>101</td><td>71</td><td>89</td><td>93</td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
<tr><td rowspan="4"><code>lb $t4,
243($zero)</code></td><td rowspan="4">$t4 = 19</td><td>1</td><td>1</td><td>1111</td><td>15</td><td>67</td><td>78</td><td>19</td><td rowspan="4">None</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td>1111</td><td>101</td><td>71</td><td>89</td><td>93</td></tr>
<tr><td>1</td><td>0</td><td>0000</td><td>126</td><td>85</td><td>2</td><td>6</td></tr>
</table>