CS61c Summer 2014 Discussion 7 – Caches

1 T:I:O Problems

An address breaks down int T:I:O.

- **Offset** - "column index" - location of address within block
- **Index** - "row index" - location (row) of block within cache
- **Tag** - "block identifier" - is this the right block?

Fill out the following table. Assume all caches are write-back.

Address Bits	**Cache Data Size**	**Block Size**	**Tag Bits**	**Index Bits**	**Offset Bits**	**Total Row Bits**
16	4 KiB	4 B				
32	32 KiB	16 B				
32			16	12		
64	2048 KiB			14		1069

Fill out the following table. Assume all caches are write-through.

Address Bits	**Cache Data Size**	**Cache Type**	**Block Size**	**Tag Bits**	**Index Bits**	**Offset Bits**	**Total Row Bits**
16	16 KiB	Direct Mapped	8 B				
16	16 KiB	2-Way Set Associative			10		
16	16 KiB	4-Way Set Associative				3	69
16	16 KiB	Fully Associative		13			
32		Direct Mapped	16 B		12		
32	64 KiB	Fully Associative	16 B				157
8	32 B				1	2	

2 Cache hits and misses

Define the following cache terms:

1. **Cache hit -**
2. **Cache miss -**
3. **Cache miss, block replacement -**

We have a byte-addressed computer, using a 32B cache with 8B blocks. The following byte memory addresses are accessed in order. Classify each access as a cache hit (**H**), miss (**M**), or miss with replacement (**R**).

1. `0x00000004`
2. `0x00000005`
3. `0x00000068`
4. `0x000000C8`
5. `0x000000DD`
6. `0x00000045`
7. `0x00000004`
8. `0x000000C8`

3 Analyzing C Code

```
#define NUM_INTS 8192
int A[NUM_INTS]; /* AT ADDRESS 0x100000 */
int i, total = 0;
for(i=0;i<NUM_INTS;i+=128) { A[i] = i; }      /* LINE 1 */
for(i=0;i<NUM_INTS;i+=128) { total += A[i]; } /* LINE 2 */
```

Let's say you have a byte-addressed computer with a total address space of 1MiB. It features a 16KiB CPU cache with 1KiB blocks.

1. How many bits make up a memory address on this computer?
2. What is the T:I:O breakdown?
3. Calculate the cache hit rate for the line marked Line 1:
4. Calculate the cache hit rate for the line marked Line 2:
5. How could you optimize this computation?

4 Average Memory Access Time

AMAT is the average (expected) time it takes for memory access. It can be calculated using the following formula: **AMAT = hit time + miss rate ∗ miss penalty**. Remember that the miss penalty is the additional time it takes for memory access in the event of a cache miss. Therefore, a cache miss takes **hit time + miss penalty time**.

Suppose that you have a cache system with the following properties. What is the AMAT?

- L1$ hits in 1 cycle (local miss rate 25%)
- L2$ hits in 10 cycles (local miss rate 40%)
- L3$ hits in 50 cycles (global miss rate 6%)
- Main memory hits in 100 cycles (always hits)