Example Problems & Solutions
Memory – Module-4

Cache Performance

- Two measures that characterize the performance of a cache are the **hit ratio** and the **effective access time**

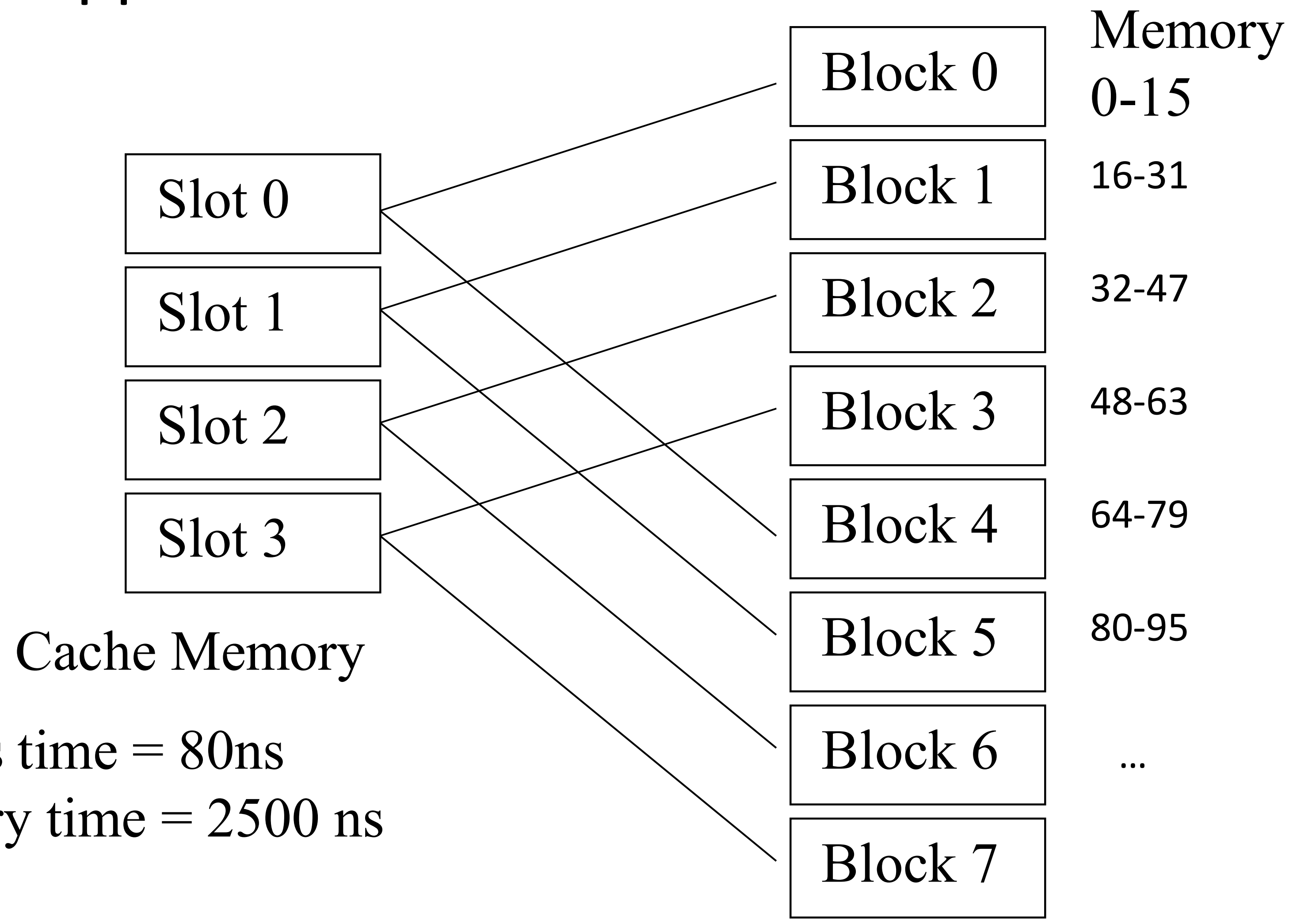
$$\text{Hit Ratio} = \frac{\text{(Number of times referenced words are in cache)}}{\text{(Total number of memory accesses)}}$$

Effective Access Time Formula

- We want to determine the impact that the memory hierarchy has on the CPU
  - In a pipeline machine, we expect 1 instruction to leave the pipeline each cycle
    - the system clock is usually set to the speed of cache
    - but a memory access to DRAM takes more time, so this impacts the CPU's performance
  - On average, we want to know how long a memory access takes (whether it is cache, DRAM or elsewhere)
    - effective access time = hit time + miss rate * miss penalty
      - that is, our memory access, on average, is the time it takes to access the cache, plus for a miss, how much time it takes to access memory
  - With a 2-level cache, we can expand our formula:
    - average memory access time = hit time$_0$ + miss rate$_0$ * (hit time$_1$ + miss rate$_1$ * miss penalty$_1$)
  - We can expand the formula more to include access to swap space (hard disk)

Cache Performance Example

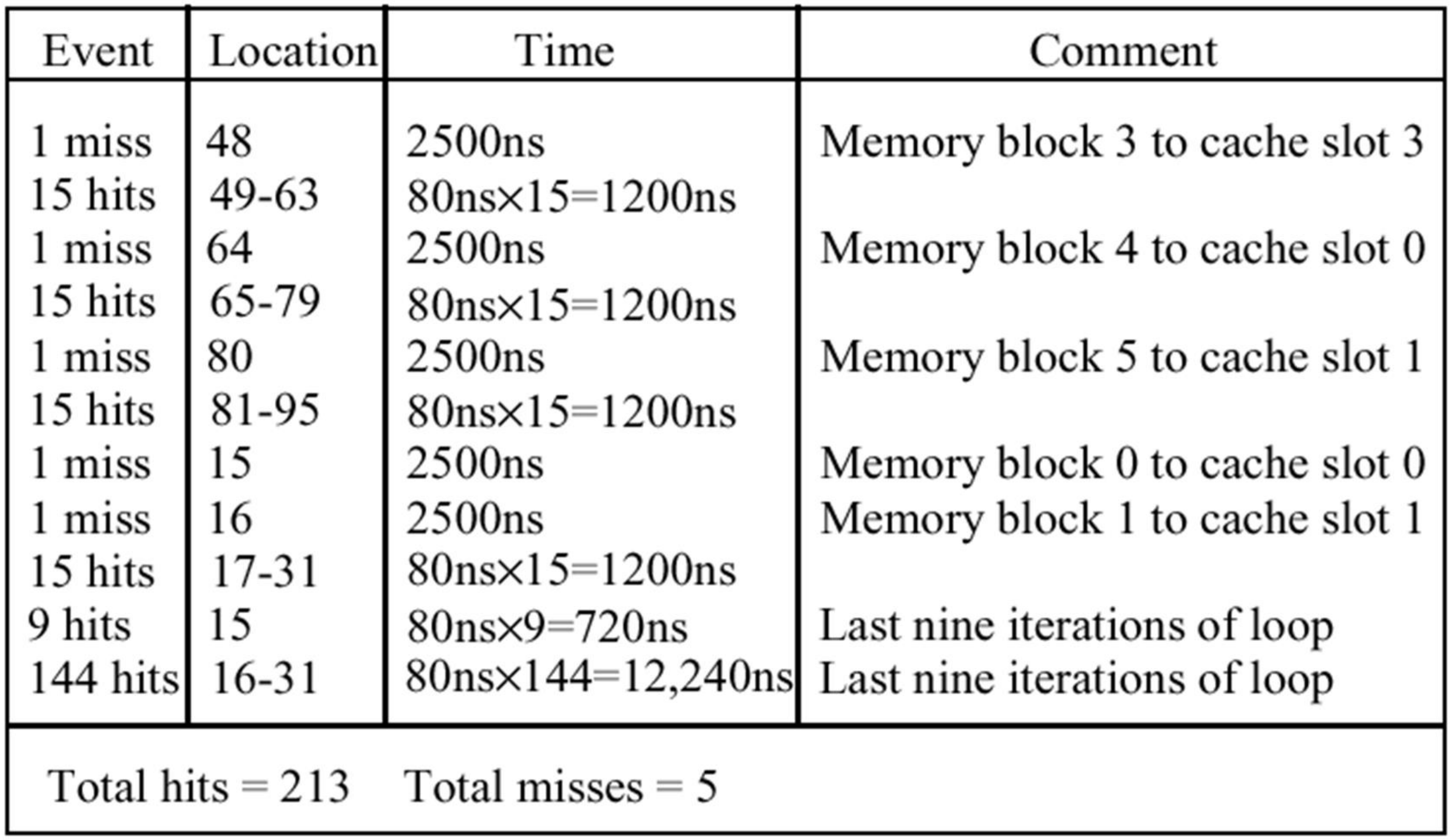
- Direct-Mapped Cache



Cache access time = 80ns
Main Memory time = 2500 ns

Cache Performance Example

- Sample program executes from memory location 48-95 once. Then it executes from 15-31 in a loop ten times before exiting.



Event	Location	Time	Comment
1 miss	48	2500ns	Memory block 3 to cache slot 3
15 hits	49-63	80ns×15=1200ns	
1 miss	64	2500ns	Memory block 4 to cache slot 0
15 hits	65-79	80ns×15=1200ns	
1 miss	80	2500ns	Memory block 5 to cache slot 1
15 hits	81-95	80ns×15=1200ns	
1 miss	15	2500ns	Memory block 0 to cache slot 0
1 miss	16	2500ns	Memory block 1 to cache slot 1
15 hits	17-31	80ns×15=1200ns	
9 hits	15	80ns×9=720ns	Last nine iterations of loop
144 hits	16-31	80ns×144=12,240ns	Last nine iterations of loop
Total hits = 213	Total misses = 5		

Cache Performance Example

- Hit Ratio: 213 / 218 = 97.7%
- Effective Access Time:
  ((213)*(80ns)+(5)(2500ns)) / 218 = 136 ns
- Although the hit ratio is high, the effective access time in this example is 75% longer than the cache access time due to the large amount of time spent during a cache miss

Example . . .

- If a direct mapped cache has a hit rate of 95%, a hit time of 4 ns, and a miss penalty of 100 ns, what is the AMAT?

  AMAT = Hit time + Miss rate x Miss penalty = 4 + 0.05 x 100 = 9 ns

- If an L2 cache is added with a hit time of 20 ns and a hit rate of 50%, what is the new AMAT?

  AMAT = Hit Time$_{L1}$ + Miss Rate$_{L1}$ x (Hit Time$_{L2}$ + Miss Rate$_{L2}$ x Miss Penalty$_{L2}$)

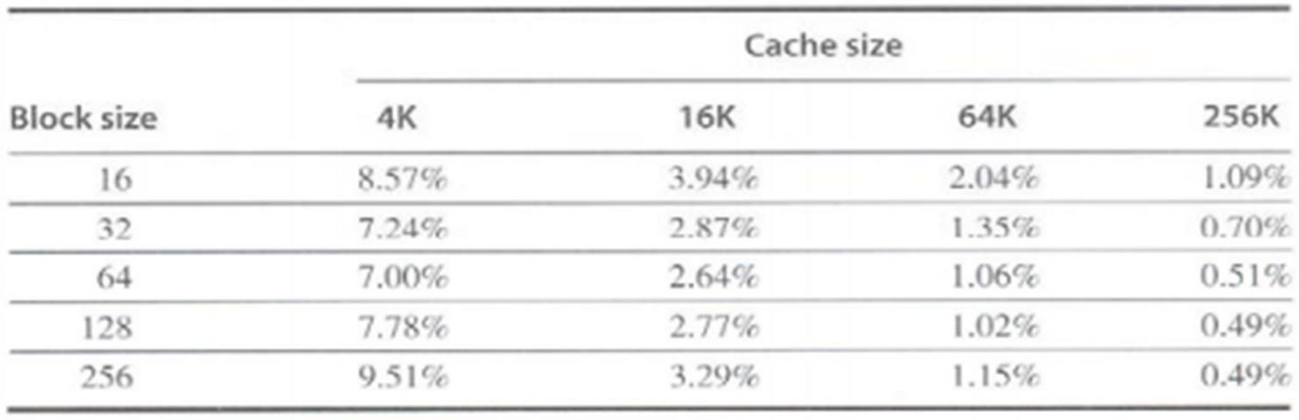
  = 4 + 0.05 x (20 + 0.5x100) = 7.5 ns

Another Example

- L1 cache has an access time of 5 ns and miss rate of 50%
- L2 cache has an access time of 50 ns and miss rate of 20%
- Main memory has an access time of 500 ns

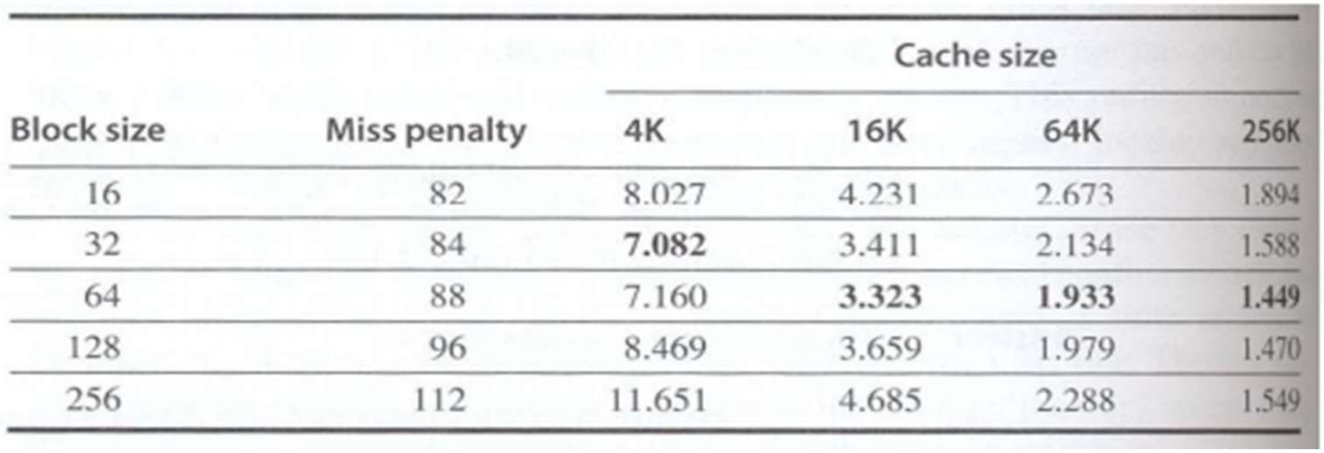
AMAT = 5ns + 0.5 * (50ns + 0.2 * 500ns) = 80 ns

AMAT-Average Memory Access Time



Block size	Cache size 4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Assume the main memory system takes 80 clock cycles of overhead per each access and then delivers 16 bytes every 2 clock cycles. Thus, it can supply 16 bytes in 82 cycles, 32 bytes in 84 cycles. Which block size has the smallest average memory access time for each cache size in the figure above? Assuming hit time 1 clock cycle, independent of block size.



Block size	Miss penalty	Cache size 4K	16K	64K	256K
16	82	8.027	4.231	2.673	1.894
32	84	**7.082**	3.411	2.134	1.588
64	88	7.160	**3.323**	**1.933**	**1.449**
128	96	8.469	3.659	1.979	1.470
256	112	11.651	4.685	2.288	1.549

Solution

average memory access time = hit time + miss rate x miss penalty

4K cache, block size 16 bytes: 1 + 8.57% x 82 = 8.027

4K cache, block size 32 bytes: 1 + 7.24% x 84 = 7.082

Example . . .

A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a direct-mapped manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) (b) When a program is executed, the processor reads data sequentially from the following word addresses: 128, 144, 2176, 2180, 128, 2176

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Solution . . .

(a) Block size = 64 bytes = 2^6 bytes = 2^6 words (since 1 word = 1 byte)
Therefore, Number of bits in the Word field = 6

- Cache size = 2K-byte = 2^{11} bytes
- Number of cache blocks = Cache size / Block size = $2^{11}/2^6 = 2^5$

Therefore, Number of bits in the Block field = 5

- Total number of address bits = 16

Therefore, Number of bits in the Tag field = 16 - 6 - 5 = 5
For a given 16-bit address, the 5 most significant bits, represent the Tag, the next 5 bits represent the Block, and the 6 least significant bits represent the Word.

Solution . . .

(b) The cache is initially empty. Therefore, all the cache blocks are invalid.
Access # 1: Address = $(128)_{10} = (0000000010000000)_2$

(Note: Address is shown as a 16-bit number, because the computer uses 16-bit addresses)

For this address, Tag = 00000, Block = 00010, Word = 000000

Since the cache is empty before this access, this will be a cache miss

After this access, Tag field for cache block 00010 is set to 00000

Solution . . .

Access # 2: Address = $(144)_{10}$ = $(0000000010010000)_2$

For this address, Tag = 00000, Block = 00010, Word = 010000

Since tag field for cache block 00010 is 00000 before this access, this will be a cache hit (because address tag = block tag)

Solution . . .

Access # 3: Address = $(2176)_{10}$ = $(0000100010000000)_2$

For this address, Tag = 00001, Block = 00010, Word = 000000

Since tag field for cache block 00010 is 00000 before this access, this will be a cache miss (address tag ≠ block tag)

After this access, Tag field for cache block 00010 is set to 00001

Example Problems . . .

a. How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?
b. How many lines of the address bus must be used to access 2048 bytes of memory?
c. How many of these lines will be common to all chips?
d. How many lines must be decode for chip select?
e. Specify the size of the decoders.

Example Problems . . .

$$\frac{2048}{128} = 16 \text{ chips}$$

$2048 = 2^{11}$ 11 lines to address 2048 bytes.

$128 = 2^{7}$ $\underline{\ 7\ }$ lines to address each chip

4 lines to decoder for selecting 16 chips

4×16 decoder

Example Problems . . .

On a computer with a 32-bit memory address and the length of the memory location of 1 byte is installed set-associative cache. Cache size is 16 KB, block (line) size is 16 Bytes, set associative cache is 4-way.

a) How many sets are there in cache?
b) Which bits in the memory address determine the address of the set?
c) Into which set is mapped the content of the memory address 10FFCFF(HEX)?

Solution . . .

a) How many sets are there in cache?

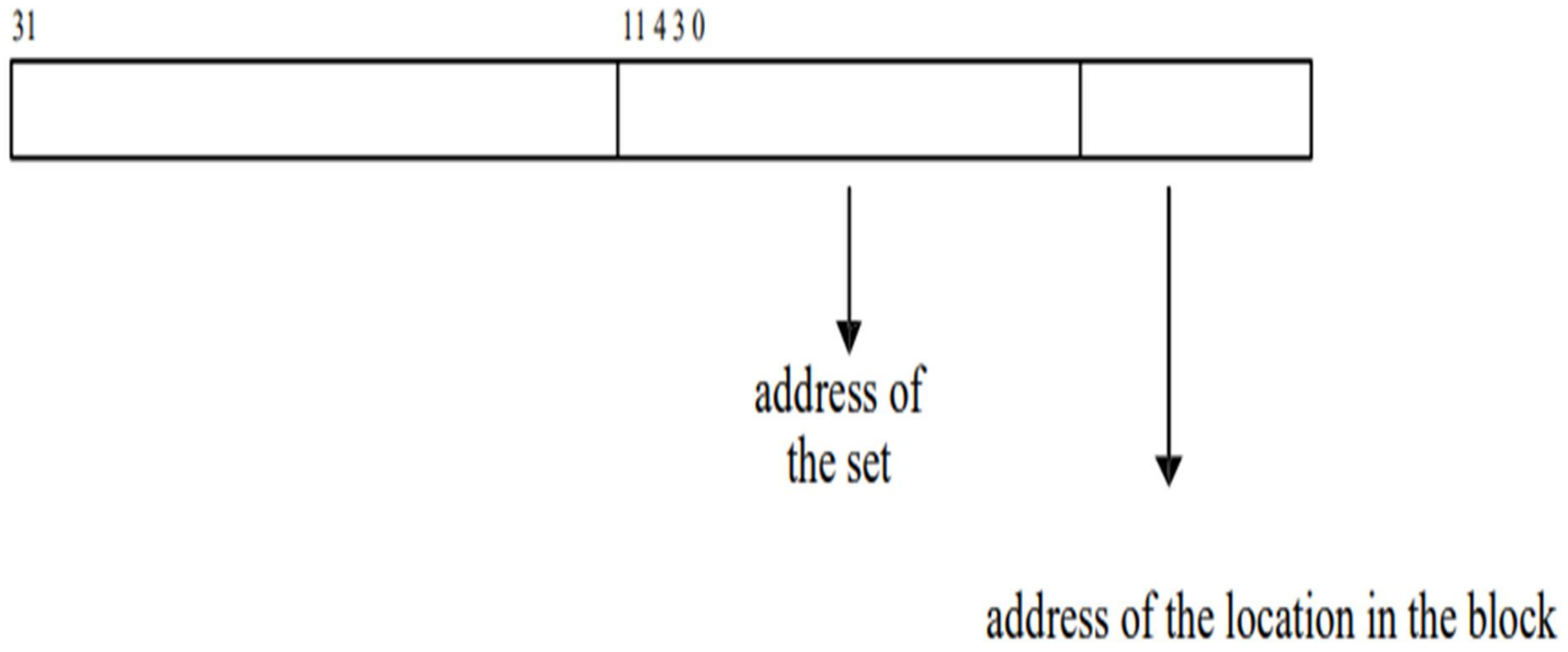
$n=32$
$M=16KB=2^4KB=2^{14}B$
$block=B=16B=2^4B$
$\underline{E=4=2^2}$

$M=S\times E\times B$

$S=M\div(E\times B)=2^{14}\div(2^2\times 2^4)=2^{14-6}=2^8=256$ sets

Solution . . .

b) Which bits in the memory address determine the address of the set?



Address bits of set: 4-11.

Solution . . .

1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

```
31                   11      4 3  0
00000001000011111111110011111111
                     set address (4-11)
```

Address belongs to set 207.

Example Problems . . .

Consider two alternate caches, each with 4 sectors holding 1 block per sector and one 32-bit word per block. One cache is direct mapped and the other is fully associative with LRU replacement policy. The machine is byte addressed on word boundaries and uses write allocation with write back.

1. What would the overall miss ratio be for the following address stream on the direct mapped cache?

Solution . . .

read	0x00	M
read	0x04	M
write	0x08	M
read	0x10	M
read	0x08	H
write	0x00	M

Miss ratio = 5/6 = 0.8333

Example Problem . . .

Consider a cache with 64 blocks and a block size of 16 bytes (4 words). What block number does byte address 1200 map to?

First of all, we know the entry into the cache is given by

Where the block address is given by $\frac{Byte\ Address}{Number\ of\ bytes\ per\ block}$.

So, the block address is $\frac{1200}{16} = 75$.

This corresponds to block number $75\ \%\ 64 = 11$. This block maps all addresses between 1200 and 1215.

Example-Problem

Assume a direct-mapped cache with 4 blocks and 8 bytes per block. How is the physical address portioned?

Tag bits	Index bits	Offset bits
27 [31-5]	2 [4-3]	3 [2-0]

Fill in the appropriate information for the following memory references:

Address	Tag	Index	Offset
4	0	0	4
8	0	1	0
12	0	1	4
20	0	2	4
67	2	0	3

Example-Problem . . .

Assume a 2-way set-associative cache with 64 sets and 4 words per block.

How is the physical address partitioned?

Tag bits	Index bits	Offset bits
22	6	4

Fill in the appropriate information for the following memory references:

Address	Tag	Index	Offset
300	0	18	12
304	0	19	0
1216	1	12	0
4404	4	19	4
4408	4	19	8

Multi-Level Cache Performance

Consider a system with a two-level cache having the following characteristics. The system does not use “shared” bits in its caches.

L1 cache

- Physically addressed, byte addressed
- Split I/D
- 8 KB combined, evenly split -- 4KB each
- Direct mapped
- 2 blocks per sector
- 1 word per block (8 bytes/word)
- Write-through
- No write allocation
- L1 hit time is 1 clock
- L1 average local miss rate is 0.15

L2 cache

- Virtually addressed, byte addressed
- Unified
- 160 KB
- 5-way set associative; LRU replacement
- 2 blocks per sector
- 1 word per block (8 bytes/word)
- Write-back
- Write allocate
- L2 hit time is 5 clocks (after L1 miss)
- L2 average local miss rate is 0.05

Solution . . .

Compute the following elements that would be necessary to determine how to configure the cache memory array:

- Total number of bits within each L1 cache block: 65 bits
  - 8 bytes per block + 1 valid bit = 65 bits
- Total number of bits within each L1 cache sector: 158 bits
  - Tag is 40 bits - 12 bits (4 KB) = 28 bits, and is the only sector overhead since it is direct mapped 28 + 65 + 65 = 158 bits
- Total number of within each L2 set: 800 bits
  - 8 bytes per block + 1 valid bit + 1 dirty bit = 66 bits cache is 32 KB * 5 in size ; tag is 40 bits - 15 bits (32 KB) = 25 bits tag per sector + 3 bits LRU counter per sector = 28 overhead bits per sector (one sector could also be only 37 bits per the trick discussed in class... but that is optional) Set size = ((66 * 2) + 28) * 5) = 800 bits per set