Examples From Sections 5.3 and 5.4 - The Basics of Caches and Improving Cache Performance

Paul Scanlon and Matt Murdoch

Colorado State University, Research for Dr. Sanjay Rajopadhye, Summer 2019

---

1. Bits in a Cache

How many total bits are required for a direct-mapped cache with 16 KiB of data and 4-word blocks, assuming a 32-bit address?

From the given information, we can deduce the following:

$$\begin{cases} 16\ KiB = 4096\ words = 2^{12}\ words. \\ Block\ size = 4\ words = 2^2\ words \implies There\ are\ 1024 = (2^{10})\ blocks. \\ Each\ block\ has\ 4 \times 32 = 124\ bits\ of\ data\ plus\ a\ tag. \\ Tag\ size = (32 - 10 - 2 - 2)\ bits\ plus\ a\ valid\ bit \end{cases}$$

The equation for total cache size is:

$$2^n \times (block\ size + tag\ size + valid\ field\ size) \tag{1}$$

Plugging in the deduced information yields:

$$2^{10} \times (4 \times 32 + (31 - 10 - 2 - 2) + 1) = 2^{10} \times 147 = 147\ Kibibits \tag{2}$$

This is 18.4 KiB for a 16 KiB cache, 1.15 times as many as needed for storage in this example.

2. Mapping an Address to a Multiword Cache Block

Consider a cache with 64 blocks and a block size of 16 bytes. To what block number does byte address 1200 map?

The block is given by:

$$(Block\ address)\ modulo\ (Number\ of\ blocks\ in\ the\ cache) \tag{3}$$

The address of the block is:

$$\frac{Byte\ address}{Bytes\ per\ block} \tag{4}$$

This block address contains all addresses between:

$$[\frac{Byte\ address}{Bytes\ per\ block}] \times Bytes\ per\ block \tag{5}$$

$$and \tag{6}$$

$$[\frac{Byte\ address}{Bytes\ per\ block}] \times Bytes\ per\ block + (Bytes\ per\ block - 1) \tag{7}$$

Thus with 16 bytes per block, byte address 1200 is block address

$$[\frac{1200}{6}] = 75 \tag{8}$$

which maps to cache block number (75 *modulo* 64) = 11. This block maps all addresses between 1200 and 1215.

3. Calculating Cache Performance

Assume the miss rate of an instruction cache is 2% and the miss rate of the data cache is 4%. If a processor has a CPI of 2 without any memory stalls and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 36%.

The number of memory miss cycles for instructions in terms of the Instruction count (I) is:

$$Instruction\ miss\ cycles = I \times 2\% \times 100 = 2.00 \times I \tag{9}$$

The same for data miss cycles is:

$$Data\ miss\ cycles = I \times 36\% \times 4\% \times 100 = 1.44 \times I \tag{10}$$

Thus, The total number of memory-stall cycles in terms of instruction count (I) is:

$$2.00I + 1.44I = 3.44I \tag{11}$$

A simple ratio calculation relating the cycles per instruction, CPI, shows the gain from a perfect cache is:

$$\frac{2 + 3.44}{2} = 2.72\ times\ faster\ with\ a\ perfect\ cache. \tag{12}$$

It is noted that speeding up the processor but not the memory system makes memory stalls more severe. The same effect occurs when increasing clock speed without augmenting memory.

If, in this example, CPI is reduced from 2 to 1, representing processor augmentation, the ammount of time spent on memory stalls increases from 63% to 77%.

4. Calculating Average Memory Access Time

Find the AMAT for a processor with a 1 ns clock cycle time, a miss penalty of 20 clock cycles, a miss rate of 0.05 misses per instruction, and a cache access time (including hit detection) of 1 clock cycle. Assume that the read and write miss penalties are the same and ignore other write stalls.

The equation for average memory access time per instruction is given as:

$$AMAT = Time\ for\ a\ hit + (Miss\ rate \times Miss\ penalty) \tag{13}$$

Plugging in the information given for this example yields:

$$AMAT = 1 + (0.05 \times 20) = 2\ clock\ cycles = 2\ nano\ seconds \tag{14}$$

This is the average memory access time for this processor.

5. Misses and Associativity in Caches

Assume there are three small caches, each consisting of four one-word blocks. One cache is fully associative, a second is two-way set-associative, and the third is direct-mapped. Find the number of misses for each cache organization given the following sequence of block addresses: 0, 8, 0, 6, and 8.

Each memory block will be mapped to the cache differently for each associativity scheme. Direct mapping is a function of the size of the cache, thus incorporating a *modulo* 4 as shown in the first table of *Figure* 1. Two-way set-associative mapping uses an index an two sets, thus incorporating a *modulo* 2 as shown in the second table of *Figure* 1. Full-associative mapping is a function of the number of memory blocks needed - three are needed with a size of four, so no mapping is needed in this example for full associativity.

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

Figure 1: Mapping Logic for Direct Mapping and 2-Set Associative

With this mapping logic, the population of the cache with respect to successive accesses can be represented as a linear table. The successive state of the cache, a hit or a miss in the desired cache line at each successive access, is shown below in *Figure* 2. Blank entries indicate invalid blocks. Colored entries indicate new additions. Non-colored entries indicate old entries for a hit or no change on that line for that access. The successive accesses are read top to bottom.

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss	Memory[8]	Memory[6]		

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

Figure 2: Consecutive Cache Population for each Scheme

The results for the five accesses for each associative scheme are five misses for direct mapping, 4 misses for two-set-accociativity and three misses for full-associativity.

It is noted that cache size and associativity are not independent in determining cache performance.

The magnitude of m in m-way set associativity directly affects the number of comparators required. Thus, the choice among direct-mapped, set-associative, or fully associative mapping in any memory hierarchy will depend on the cost of a miss versus the cost of implementing associativity, both in time and in extra hardware.