Chapter 6

Memory System Hierarchy: Role of Memory System (cont.)

Example:

A computer system has 512K words of MM and a direct mapped cache. The block size is 64 words and the tag field of the MM address is 6 bit wide.

a) If the same cache were set associative mapped, the MM address will have a 9- bit tag field. How many words are there in the cache? How many blocks are there in the cache set?

b) If the same cache were fully associative mapped, determine the MM address tag field

Solution:

a) $MM\ Size = 512\ K\ words\ \equiv 2^{19} word = \ 2^{n}$

$MM\ Address = n = 19\ bit$

$Block\ \ Size = 64\ words\ \equiv 2^{6} word = \ 2^{w}$

$word\ offset = w = 6\ bit$

Solution (cont.):

$\text{DM } cache\ tag = 6\ bit$

$Thus, No. of\ bits\ needed\ for\ cache\ lines\ c = n - t - w$

$$c = 19 - 6 - 6 = 7\ bit$$

$$\therefore\ the\ total\ No. of\ cache\ lies = 2^c = 2^7 = 128\ lines$$

$$line\ size\ = Block\ size = 64\ words$$

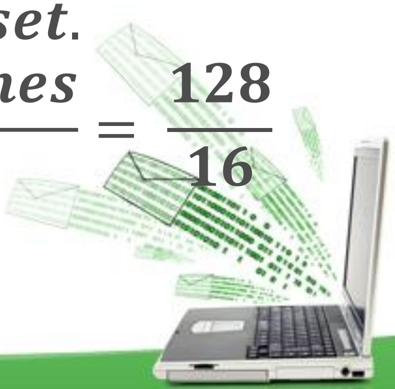
$Thus, cache\ size = Total\ No. of\ lines\ * line\ size = 2^7 * 2^6 = 2^{13} = 8\ K\ word$

$for\ SA\ cache, the\ tag = 9\ bit,\ w = 6\ bit.$

$Thus, the\ No. of\ bits\ needed\ for\ the\ sets = s = n - t - w = 19 - 9 - 6 = 4\ bits.$

$$The\ No. of\ sets\ in\ SA\ cache\ = 2^s = 2^4 = 16\ set.$$

$The\ total\ No. of\ cache \frac{line}{set} = \frac{Total\ No. of\ cache\ lines}{No. of\ sets} = \frac{128}{16} = 8\ lines/set$



Q2: How is a block found if it is in the upper level? (Block identification)

Caches have an address tag on each line that gives the MM block address.

The tag of every cache line that might contain the desired information is checked to see if it matches the MM block address generated from the CPU. As a rule, all possible tags are searched in parallel because speed is critical.

Q3: Which block should be replaced on a miss? (Block replacement)

* Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced.

* For direct mapping there is only one possible line for any particular block and no choice is possible.

* For the associative and set-associative techniques a replacement algorithm is needed

The replacement algorithms are:

1) **LRU** (Least Recently Used) - Replace that block in the set that has been in the cache longest with no reference to it. Because of its simplicity of implementation, LRU is the most popular replacement algorithm.

2) **FIFO** (First In- First Out) - Replace that block in the set that has been in the cache longest.

3) LFU (Least Frequently Used) - Replace that block in the set that used the fewest times.

4) Random - A randomly selected line form cache is replaced

Q4: What happens on a write? (Write strategy)

In case of (cache write hit), there are two basic options when writing to the cache:

A. Write through: **The data is written to both the cache line and to the block in the MM memory. Simplest technique**

A. Write back: **The result is written only to the line in the cache. The modified cache line is written to MM only when it is replaced. Minimizes memory writes but needs complex circuitry.**

Case2: when a cache Write miss occur

Since the data are not needed on a write, there are two options on a cache write miss:

A. **Write allocate:** The block is allocated on a write miss, followed by a **write hit actions**. In this natural option, write misses act like read misses.

B. **No-write allocate:** This apparently unusual alternative is write misses do not affect the cache. Instead, the block is modified only in the MM.

EXAMPLE 3

Assume a fully associative write back cache with many cache entries that starts empty. Below is a sequence of 7 instructions (the address is in parentheses):

```
MOV AX, Mem[100];      read
MOV BX, Mem[100];      read
MOV Mem[200], DX;      write
MOV CX, Mem[200];      read
MOV DX, Mem[100];      read
MOV Mem[200], BX;      write
MOV Mem[300], DX;      write
```

What are the number of hits and misses with using no-write allocate versus write allocate?

Solution

Instruction		Write Allocate		Write No Allocate	
		Hit	Miss	Hit	miss
MOV AX, Mem[100];	read				
MOV BX, Mem[100];	read				
MOV Mem[200], DX;	write				
MOV Mem[200], CX;	write				
MOV DX, Mem[100];	read				
MOV Mem[200], BX;	write				
MOV Mem[300], DX;	write				

Cache Performance:

A better measure of memory-hierarchy performance is the average memory access time (AMAT):

AMAT (*clock cycle*) = Hit time + Miss rate × Miss penalty

Where:

Hit time: is the time to hit the data requested by CPU in the cache

(*usually takes 1 clock cycle*)

Miss penalty: In case of a cache miss, the time needed to retrieve the first word of the missed MM block from the MM memory to the cache and CPU, it represents the miss penalty (sometimes is called latency) .

Example: A typical computer system has a MM of 512Kword and a fully associative cache memory of 16Kword. The cache block size is 32words and the cache miss penalty is 100 clock cycle.

1) What is the size of the tag?

2) If the cache hit time is 1 clock cycle, what is the hit rate would be required to achieve an AMAT equal to 4.25 clock cycle?

Solution:

1) What is the size of the tag?

$$\text{MM size} = 512Kword = 2^{19}word \equiv 2^{n}$$

$$\therefore n = 19\ bit\ \ : MM\ address\ lines$$

$$MM\ block\ size = Cache\ line = 32\ word = 2^{5} = 2^{w}$$

$$\therefore word\ ofset\ \ (w) = 5\ bit$$

$$cache\ size = 16kword = \ 2^{14}$$

$$cache\ lines: C = \frac{Cache\ size}{block\ size} = \frac{16kword}{32\ word} = 512\ words \equiv 2^{c}$$

$$No.of\ bits\ needed\ to\ address\ cahe\ lines: c = 9bit$$

$$cache\ tag\ : t = n - c - w = 19 - 9 - 5 = 5bit$$

2) If the cache hit time is 1 clock cycle, what is the hit rate would be required to achieve an AMAT equal to 4.25 clock cycle?

$$AMAT = hit\ time + miss\ rate\ * miss\ penalety$$

$$4.25 = 1 + \text{miss rate} * 100$$

$$miss\ rate = \frac{4.25 - 1}{100} = 0.0325$$

$$hit\ rate = 1 - miss\ rate = 1 - 0.0325 = 0.9675$$

A **split cache** is a cache that consists of two *physically separate* parts, where one part:

- Instruction cache (I- cache), is dedicated for holding instructions and,
- Data cache (D- cache), is dedicated for holding data (i.e., instruction memory operands).

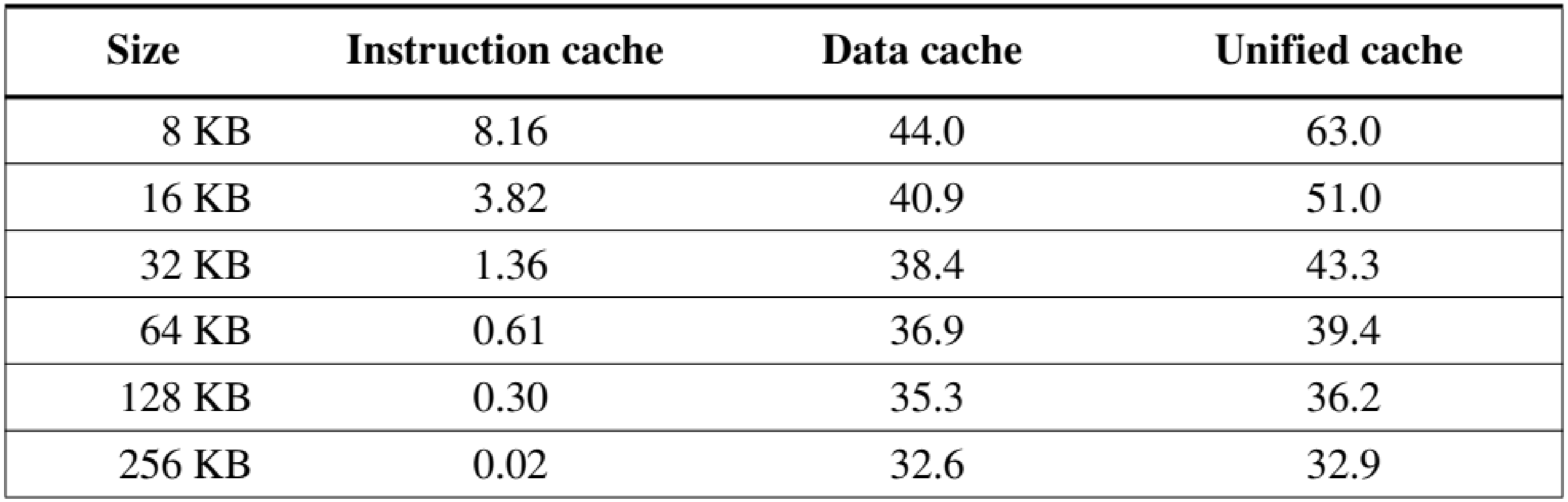
Both of the I- cache and D- cache are *logically* considered to be a single cache, described as a split cache, because both are hardware-managed caches for the same physical address space at the same level of the memory hierarchy.

- Instruction fetch requests are handled only by the instruction cache and
- memory operand read and write requests are handled only by the data cache.

- A cache that is not split is called a unified cache.

E X A M P L E 5:

Which has the lower miss rate: a 16-KB instruction cache with a 16-KB data cache or a 32-KB unified cache?
Use the miss rates in Figure 1 to help calculate the correct answer, assuming 47% of the instructions are data transfer instructions. Assume a hit takes 1 clock cycle and the miss penalty is 100 clock cycles. A load or store hit takes 1 extra clock cycle on a unified cache if there is only one cache port to satisfy two simultaneous requests. Using the pipelining terminology of the previous chapter, the unified cache leads to a structural hazard. What is the average memory access time in each case? Assume write-through caches with a write buffer and ignore stalls due to the write buffer.



Size	Instruction cache	Data cache	Unified cache
8 KB	8.16	44.0	63.0
16 KB	3.82	40.9	51.0
32 KB	1.36	38.4	43.3
64 KB	0.61	36.9	39.4
128 KB	0.30	35.3	36.2
256 KB	0.02	32.6	32.9

Figure 1: Miss per 1000 instructions for instruction, data, and unified caches of different sizes. The percentage of instruction references is about 78%. The data are for two way associative caches with 64-byte blocks for the same computer

ANSWER

First let's convert misses per 1000 instructions into miss rates. Solving the general formula is from above, miss rate is

$$\text{Miss rate} = \frac{\dfrac{\text{Misses}}{\text{1000 Instructions}} / 1000}{\dfrac{\text{Memory accesses}}{\text{Instruction}}}$$

Since every instruction access has exactly 1 memory access to fetch the instruction, the instruction miss rate is:

$$\text{Miss rate}_{\text{16 KB Instruction}} = \frac{3.82/1000}{1.00} = 0.004$$

Since 47% of the instructions are data transfers, the data miss rate is:

$$\text{Miss rate}_{\text{16 KB Data}} = \frac{40.9/1000}{0.47} = 0.087$$

The unified miss rate needs to account for instruction and data accesses:

$$\text{Miss rate}_{\text{32 KB Unified}} = \frac{43.3/1000}{1.00 + 0.47} = 0.029$$

As stated above, about 78% of the memory accesses are instruction references. Thus, the overall miss rate for the split caches is

$$(78\% \times 0.004) + (22\% \times 0.087) = 0.022$$

Thus, a 32-KB unified cache has a higher effective miss rate than two16- KB caches.

The average memory access time formula can be divided into instruction and data accesses:

AMAT= % instructions × (Hit time + Instruction miss rate × Miss penalty) +
% data × (Hit time + Data miss rate × Miss penalty)

Therefore, the time for each organization is

$$\text{Average memory access time}_{\text{split}}$$
$$= 78\% \times (1 + 0.004 \times 100) + 22\% \times (1 + 0.087 \times 100)$$
$$= (78\% \times 1.38) + (22\% \times 9.70) \;=\; 1.078 + 2.134 \;=\; 3.21$$

$$\text{Average memory access time}_{\text{unified}}$$
$$= 78\% \times (1 + 0.029 \times 100) + 22\% \times (1 + 1 + 0.029 \times 100)$$
$$= (78\% \times 3.95) + (22\% \times 4.95) \;=\; 3.080 + 1.089 \;=\; 4.17$$

Hence, the split caches in this example—which offer two memory ports per clock cycle, thereby avoiding the structural hazard—also have a better average memory access time (AMAT) than the unified cache.