Drake Bridgewater & Ryan Phillips
May 15, 2014
CS 472 HW 5

(9.2)

Why do computer use cache memory?

> Computers use cache memory to give the appearance of speed. When cache memory is used in conjunction with system memory and primary (secondary as the book puts it) storage the system can operate at much higher rates.

(9.3)

What is the meaning of temporal locality and spatial locality?

> Addresses are said to exhibit **spatial locality** because they are clustered within the same region of memory. Which other addresses are said to exhibit **temporal locality** because they are accessed over and over again within a short time span.

(9.4)

From first principles, derive an expression for the speedup ratio of memory system with cache (assume the hit ratio is h and the ratio of the main storage access time to cache access time to cache access time is k, where k ¡ 1). Assume that the system is an ideal system and that you don't have to worry about the effect of clock cycle times.

```
hit ratio = h
miss ratio = 1-h
main storage access time / cache access time = k
let's let k: m/c
speed up ratio = S
S is essentially the time with utilizing the cache over
           the time without using the cache
so the numerator is: hc + (h-1)m
(the hit rate * cache access time + miss rate * main memory access time)
and the denominator is just 'm'
```

```
S = hc + (h-1)m / m
```

(9.5)

For the following systems, calculate the speedup ratio S in the case t_c is the access time of the cache memory, t_m is the access time of the main store, and h is the hit ratio. Using $S = \frac{1}{1-H(1-k)}$ where $k = \frac{t_c}{t_m}$

a $t_m = 70ns,\ t_c = 7ns,\ H = 0.9$: **5.263**

b $t_m = 60ns,\ t_c = 3ns,\ H = 0.9$: **6.897**

c $t_m = 60ns,\ t_c = 3ns,\ H = 0.8$: **4.167**

d $t_m = 60ns,\ t_c = 3ns,\ H = 0.97$: **12.739**

(9.6)

For the following ideal systems, calculate the hit ratio h required to achieve the stated speedup ratio S. Using this equation$S = \frac{1}{1-H(1-k)}$ where $k = \frac{t_c}{t_m}$ we can solve it for H to get $H = \frac{1}{1-k} - \frac{1}{S(1-k)}$

a $t_m = 60ns,\ t_c = 3ns,\ S = 1.1$: **0.0957**

b $t_m = 60ns,\ t_c = 3ns,\ S = 2.0$: **0.5263**

c $t_m = 60ns,\ t_c = 3ns,\ S = 5.0$: **0.8421**

d $t_m = 60ns,\ t_c = 3ns,\ S = 15.0$: **0.9824**

(9.8)

For the following system that use a clocked microprocessor, calculate the maximum speedup ratio you could expect to see as h approaches 100%.

a $t_{cyc} = 20ns,\ t_m = 75ns,\ t_c = 15$

a $t_{cyc} = 20ns,\ t_m = 75ns,\ t_c = 25$

a $t_{cyc} = 10ns,\ t_m = 75ns,\ t_c = 15$

(9.11)

In a direct mapped cache memory system, what is the meaning of the following terms: word, line and set?

In a direct-mapped cache, the lines are arranged into units called sets, where the size of a set is the same size as the cache. The lines are then divided into words. These terms are all terms to narrow down where. Essentially like a persons address city like set, street like line, and word like house number.

(9.12)

How is data in main store mapped on to each of the following: a direct-mapped cache, a fully associative cache, and a set-associative cache?

Direct Mapped: Each location in main memory goes with one entry in the cache. This is one of the simplest types of caches to design.

Fully Associative: Has no restrictions on where data can be located. It uses a tag and a valid bit to check and retrieve the data; done in parallel.

Set-Associative: Set-associative is a combination of fully associative and direct mapped. Lines are grouped into sets. A given address is mapped into a set (like in direct-mapped), and within the the set the lines are organized like that of a fully associative scheme.

(9.17)

What is cache coherency?

> "Cache coherence is the consistency of shared resource data that ends up stored in multiple local caches. When clients in a system maintain caches of a common memory resource, problems may arise with inconsistent data. This is particularly true of CPUs in a multiprocessing system." Wikipedia says it best.

(9.22)

Why is it harder to design a data cache than an instruction cache?

> Because an instruction cache requires fewer features. For example: entries are never modified, and you don't have to worry about swapping out instructions, because the program doesn't change during the course of its execution.

(9.23)

When a CPU writes to the cache, both the item in the cache and the corresponding item in the memory must be updated. If data is not in the cache, it must be fetched from memory and loaded in the cache. If t_1 is the time taken to reloaded the cache on a miss, show that the effective average time of the memory system is given by $t_{avg} = ht_c + (1-h)t_m + (1-h)t_1$

> $t_{avg} = hits + misses$
> $hits = ht_c$
> $misses = miss\ time + time\ fetch\ memory$
> $misses = (1-h)t_m + (1-h)t_1$
> $t_{avg} = ht_c + (1-h)t_m + (1-h)t_1$

(9.26)

A system has a level 1 cache and a level 2 cache. the hit rate of the level 1 cache is 90% and the hit rate of the level 2 cache is 80%. An access to level 1 cache requires one cycle, an access to level 2 cache requires four cycles, and an access to main memory requires 50 cycles. What is the average access time?

If we take a sample of 100 accesses to memory/cache. 90% of these call will be caught by L1 cache and will require 1 cycle to complete, leaving 10 accesses which 80% will be caught by L2 cache requiring 4 cycles, lastly if the CPU misses on L1 and L2 then it will have to fetch from main memory requiring 50 cycles.

$$100_{total\_access} = (100 * 90\%_{hits} * 1_{cycles}) + (10 * 80\%_{hits} * 4_{cycles}) + (2 * 100\%_{hits} * 50_{cycles})$$

$$(90 + 22 + 100)/100 = 2.3\ Avg.\ Cycles$$

(9.28)

In the context of multilevel caches, what is the difference between a local miss rate and global miss rate?

Local miss rate is the rate of misses that occur for a specific cache. The global miss rate takes all of the caches into account.

(9.35)

A 64-bit processor has a 8-MB, four-way set-associative cache with 32-byte lines. How is the address arranged in terms of set, line and offset bits?

set: 2-bits, line: 16-bits, word: 2-bits

(9.41)

What are the fundamental differences between cache memory (as found in a CPU) and cache memory found in a hard disk drive?

Typically the memory found in a hard disk is stored on a mechanical medium where as cache memory is stored using electrical gates. As for the cache memory found in a hard drive it need to be large enough that when DRAM controller is trying to write to the hard disk the DRAM controller is not having to wait.

(9.42)

What are the differences between write-back and write-through caches, and what are the implications for system performance?

With a write-through cache, write is done synchronously both to the cache and to the backing store. With a write-back cache, writing is initially done only to the cache. The write to the backing store is postponed until the cache blocks containing the data are about to be modified/replaced by new content. (Wikipedia.org).

A write through cache will be slower because it's not finished until it has written to the main store, which is much slower than the cache. But a write through cache is also simpler to implement because it doesn't need to keep track of all of the locations stored in cache, but not in the main store (like what a write-back cache requires).

(9.43)

A computer with a 32-bit address architecture has a memory management system with single-level 4KB page tables. How much memory space must be devoted to the page tables?

To determine the location within a page, 4KB, you will need 12 bits. With the remaining bits you can use to determine which page to access; this leads us to have a total of 20 remaining bits, each specifying different pages of 4KB. Resulting in a total memory space of 4GB devoted to page tables.

(9.45)

Look at the table in the book to complete. A computer runs with the characteristics in the following table and determine the average number of cycles per instruction?

Arithmatic operations, 70%, 1 cycle
Conditional operations, 15%, 2 cycles
Load, 10%, 2 cycles
Store, 5%, 2 cycles
Hit rate, 95%
Cost of a cache miss (read), 10 cycles
Write-through time, 5 cycles (writes to memory are not buffered)

$$Avg\_Cycles = (70\% * 1 + 15\% * 2 + 10\% * 2 + 5\% * (2 + 5_{wToMemory}))(.05 * 10_{miss\ cost})(x) + (70\% * 1 + 15\% * 2 + 10\% * 2 + 5\% * (2 + 5_{wToMemory}))(.95)(X)$$

(9.46)

Consider the following code that accesses three values in memory scalar integers x and s , and an integer vector $y[i]$. what is the memory latency in clock cycles for a trip round the loop (after the first iteration)? Assume that the array is not cached and each new access to the array results in a miss. the system has both L1 and L2 caches. the access time of the L1 cache is 2 cycles, L2 cache is 6 cycles, and the main memory has access time of 50 cycles. in this case all memory and cache memory accesses take place in parallel

```
for (i=0; i<100; i++)
{
    x=y[i];
    s=s+x;
}
```

If all memory and cache read/writes take place in parallel, we are bound by the slowest, which is the main memory access time. We can safely assume that i will be cached, but we need to read y[i], write x, read s, read x, write s. This is five operations per loop, each which will require 50 cycles, giving a total of 250 cycles. But after the first iteration, we can assume that s and x are cached, meaning that we only need to retrieve y[i] from main memory. This yields 50+50+2+2+50 = 154 cycles, much better than the initial 200.

(9.57)

A compter with a 24-bit address bus has a main memory of size 16MB and a cache size of 64KB. The word length is two bytes. What is the address format for the direct-mapped cache with a line size of 32 words? What is the address format for a fully associative cache with a line size of 32 words? What is the address format for a four-way set associative cache with line size of 16 words?

For a direct-mapped cache set: 8-bits, line: 6-bits, word: 5-bits
For a fully associative cache line: 6-bits, word: 10-bits
For a four-way set associative cache set: 2-bits, line: 6-bits, word: 8-bits