Seminar 3
Memory Hierarchy

Computer Organization and Components / Datorteknik och komponenter (IS1500), 9 hp
Computer Hardware Engineering / Datorteknik, grundkurs (IS1200), 7.5 hp

KTH Royal Institute of Technology

Introduction

The purpose of seminars is to enable active learning of the more theoretical tasks that are typically part of the final written exam. Seminars are optional. However, we strongly recommend that you perform these seminar exercises and attend the seminar.

Rules. You may receive up to 1 extra point on the fundamental part of the written exam if:

- you make an honest *attempt* to solve *all* the seminar exercises on your own. You may discuss the exercises with your friends, but you are not allowed to copy any solutions from anyone or anywhere. You need to have written down a potential solution on all assignments. You are not allowed to skip some exercises, you need to try to provide a solution for all exercises.
- you write down your solutions *by hand* on this exercise form. You are not allowed to hand in machine printed solutions, copies, or handwritten solution on another paper format.
- you bring your solution *personally* to the seminar and attend the whole seminar. This means that you are not allowed to hand in a solution on behalf of someone else.
- you need to have signed this form before you hand it in.
- you are not allowed to attend the seminar if you are not bringing a solution, that is, if you do not bring this form filled out with your own solutions, you cannot attend the seminar.
- you must come to the seminar on time when it starts. If you are not there from the beginning, the assistants may refuse that you participate in the seminar.

Note that the extra point is only valid on the next ordinary exam, and the following two retake exams. During the seminar, the teaching assistant or teacher will go through the solutions and you will correct the solution done by another student. You need to have received at least 50% of the total number of points to pass the seminar. In such a case, you get one extra bonus point on the exam. We recommend that you take a photo of your solutions before you hand it in.

By signing the following, I hereby guarantee that I follow the rules above.

Your name (printed): Philip Salqvist

Signature: [illegible]

Date: 2021-02-17

Exercises

1. Suppose you have a 32-bit MIPS processor. For each of the two following cache configurations, how many bits are used for the *tag* field, the *set* field (also called the *index*), and the *byte offset* field of the address?

   (a) A direct mapped cache with capacity 2048 bytes and block size 8 bytes.

   (b) A 4-way set associate cache with capacity 4096 bytes and block size 16 bytes.

   Your solution:

a) $C = 2048$ bytes , $b = 8$ bytes , index: $\frac{2048}{8} = 2^8 \Rightarrow 8$ bits

byte-offset: $8 = 2^3 \Rightarrow 3$ bits

tag: $32 - 8 - 3 = 21$ bits

b) index: $\frac{4096}{16 \cdot 4} = 64 = 2^6 \Rightarrow 6$ bits

byte-offset: $16 = 2^4 \Rightarrow 4$ bits

tag: $32 - 6 - 4 = 22$ bits

2. Assume that you have a 32-bit MIPS processor with a direct mapped data cache with the capacity 4096 bytes and a block size of 16 bytes. The cache is initially empty (all valid bits are 0). Which sets of the cache have been updated *after* that the following program has been executed? For each of the sets, specify the *set number*, the *value of the valid bit*, and the *tag value* of the data cache.

```
1 lui     $t0,0x12ff
2 lw      $t1,0x1240($t0)
3 lw      $t2,0x5aa4($t0)
4 lw      $t3,0x4248($t0)
```

Your solution:

$\frac{4096}{16} = 256 = 2^8 \Rightarrow 8$ set bits

$b = 16 = 2^4 \Rightarrow$ byte offset 4 bits

$32 - 8 - 4 = 20 \Rightarrow 20$ tag bits

2.	0001 0010 1111 1111 0001	0010 0100	0000
3.	—— 〃 —— 0101	1010 1010	0100
4.	—— 〃 —— 0100	0010 0100	1000

number 36 and 170 have been updated

Set number	valid bit	tag number
0x24	1	0x12ff4
0xaa	1	0x12ff5

3. Consider Listning 1 on the last page. Assume the code is executed on a 32-bit MIPS processor that includes a *data cache* with the following properties: Capacity =1024bytes, direct mapped, block size = 16 bytes. Assume that the cache is empty (all valid bits are zero) before you call function `sum`. Will the *data cache* utilize temporal locality or spatial locality, or both? For each of the following C function calls, what is the *data cache hit rate* when executing the function?

   (a) `sum(0x55aa1000,10);`

   (b) `sum(0x55aa100a,30);`

Your solution:

We have spatial but not temporal locality

a) # Hits = 7, # access = 10 => Hit rate = $\frac{7}{10}$ = 0,7

b) # Hits = 3·7+1 = 22, # access = 30

Hit rate = $\frac{3\cdot 7+1}{30} = \frac{22}{30} = \frac{11}{15}$

4. Consider Listning 1 on the last page. Assume the code is executed on a 32-bit MIPS processor that includes an *instruction cache* with the following properties: Capacity = 512bytes, direct mapped, block size = 8 bytes. Assume that the cache is empty (all valid bits are zero) before you call function `sum`.

   (a) Will the *instruction cache* utilize temporal locality or spatial locality, or both?

   (b) What is the *instruct cache miss rate* when executing the function? Only count the memory accesses within the function, that is, the first instruction in the function is `xor` and the last one `jr`. The function is called with the following C statement `sum(0x6ff1000,100);` and the first instruction of `sum` is located at address `0x00400000`. Answer as a rational number.

Your solution: a) Both, spatial since we load more than one instruction when loading to cache. And temporal since we loop hundred times the same instr. will be executed 100 times.

b)

0x00400000	M		} 1
0x00400004	H	H	
0x00400008	M	H	
0x0040000C	H	H ...	} 5×100
0x00400010	M	H	
0x00400014	H	H	
0x00400018			M } 1

Miss rate = $\frac{\#\text{ misses}}{\#\text{ access}} = \frac{4}{100}$ = 0,04

5. Suppose we have a 32-bit MIPS processor, which includes a *2-way set associative data cache* with capacity 16384 bytes, 16 bytes block, and a *least recently used (LRU)* replacement policy. Assume that the cache is empty (all valid bits are 0) before the following code is executed.

```
        lw      $t1,0x1040($0)
        lw      $t2,0x2044($0)
        lw      $t3,0x3048($0)
        lw      $t4,0x1044($0)
        lw      $t5,0x504c($0)
        lw      $t6,0x3040($0)
```

For each of the six assembly instructions above, state i) the *set field* value for the accessed address, ii) the *tag field value*, and iii) if the instruction results in a cache hit or a cache miss.

Your solution:

	tag	index	byte offset	H/M
1.	0000 0000 0000 0000 000	1 0000 0100	0000	M
2.	—— 〃 —— 001	0 0000 0100	0100	M
3.	—— 〃 —— 001	1 0000 0010	1000	M
4.	—— 〃 —— 000	1 0000 0100	0100	H
5.	—— 〃 —— 010	1 0000 0100	1100	M
6.	—— 〃 —— 001	1 0000 0100	0000	M

6. For each of the following statements, state if they are *true* or *false*.

   (a) A cache using a write-back policy writes back to the main memory simultaneously as it writes to the cache.

   (b) Two important properties of a virtual memory are to give the illusion of a very large memory and to give memory protection between two concurrently running programs.

   (c) A unit called the *memory management unit (MMU)* is a common solution to perform fast translations from virtual page numbers to physical page numbers.

   (d) Modern high-performance processors do not use multi-level caches because they are too expensive and give little performance benefits.

Your solution:

a) false

b) true

c) true

d) false

The following listing is used in the seminar exercises above. You need this information to solve the exercises, but you do not have to print this page (page 5) and bring it to the seminar.

Listning 1.

```
sum:
        xor     $v0,$v0,$v0
loop:
        lb      $t0,0($a0)
        add     $v0,$v0,$t0
        addi    $a0,$a0,1
        addi    $a1,$a1,-1
        bne     $a1,$zero,loop
        jr      $ra
```