Exercises 5
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
Friday 18th December, 2020

Memory Types and Concepts

1. Explain the main properties of the following memory types: *SRAM*, *DRAM*, *Flash Memory*, and *Magnetic Disks*.

2. Explain the meaning of *temporal locality* and *spatial locality*.

3. Explain at a high level the three memory levels *cache memory*, *main memory*, and *virtual memory*, and how these levels interact.

Direct Mapped Cache

4. Assume that we have a 32-bit MIPS processor with a direct mapped instruction cache. Assume further that the number of *sets* (also called rows) are 16, and the block size is 8 bytes.

   (a) Draw the hardware implementation for reading from the instruction cache.

   (b) Explain what the address fields *tag*, *set* (also called *index*), and *byte offset* mean. Compute the sizes of these fields.

   (c) Explain step-by-step what happens in the cache and how the cache content is updated when executing the program below:

```
        addi    $t0,$0,0
        addi    $t1,$0,0
loop:
        add     $t1,$t1,$t0
        addi    $t0,$t0,1
        slti    $t2,$t0,10
        bne     $t2,$0,loop
```

   The first assembly instruction starts at address 0x00400000 and the cache is initially “empty”. In particular, explain the following concepts for the execution:

   i. What can an “empty” cache actually mean? What is the meaning of valid bits? When are these bits updated?

   ii. How are the sets (rows) selected?

iii. What is the meaning of the *tag* in the cache content? How is it related to the *tag* in the address?

iv. What is the meaning of the *data* portion of the cache content?

v. Does the program have temporal locality, or spatial locality, or both?

(d) How many memory bits are needed for this cache to work correctly (including all control bits such as valid bits and tags).

(e) Compute the cache *hit rate* and the cache *miss rate* when running the complete assembly program fragment above.

5. Consider the following C program:

```
float v1[] = {33.0, 71.2, 19.1,   2.0,
              99.9, 44.1, 123.12, 77.12};
float v2[] = {66.3, 1.4,  22.7, 4.1,
              1.0,  1.1,  5.7,  99.9};
float v3[8];

void vectoradd(float* a, float* b, float* c, int len){
  int i;
  for(i=0; i<len; i++){
    c[i] = a[i] + b[i];
  }
}
```

Assume that the above program is compiled for a 32-bit MIPS processor which has a direct mapped data cache with the capacity 4096 bytes and a block size of 16 bytes. Assume further that variable `i`, parameter `len`, and the pointers `a`, `b`, and `c` are all located in registers. Moreover, we assume that the cache is initially empty and that the assembly code first loads an element via pointer `a`, followed by a load via pointer `b`, before finally writing the data to the memory using pointer `c`.

(a) We assume that `v1` is located at address `0x10008000` and that the allocation of `v2` and `v3` follows directly after in the memory with no empty space in between the declared arrays. What is then the data cache hit rate for executing the following function call:

```
vectoradd(v1,v2,v3,8);
```

Is there temporal locality or spatial locality or both?

(b) If we use the same assumptions as in (a), but now execute the following

```
vectoradd(v1,v1,v3,8);
```

what is then the data cache hit rate? Is there temporal locality or spatial locality or both?

(c) Finally, if we now assume that `v1` is located at address `0x10008000`, `v2` is located at address `0x10009000`, and `v3` is located at address `0x1000a000`, and we execute

```
vectoradd(v1,v2,v3,8);
```

what is then the data cache hit rate? Is there temporal locality or spatial locality or both? What other issue is introduced in this example?

N-way Set Associative Cache

6. Assume that we have an N-way set associative cache that has a block size of 8 bytes and the number of sets is 256. Assume that the replacement policy is *least recently used (LRU)*.

   (a) If the capacity of the cache is 8192 bytes, what is then the associativity of the cache?

   (b) Sketch the layout of the cache content for the cache if we assume that N = 2. The figure should include the ways and columns for the valid bit, the tag, and the data. What is then the capacity of the cache?

   (c) Consider now the C program that was presented in exercise 5(c) and that this program is compiled and executed with either the cache in 6(a) or 6(b). What is then the data cache hit rate for the two cases?

Other Concepts

7. Explain the difference between a *write-through policy* and a *write-back policy*.

8. What does multi-level caches mean and why are they common in modern processors?

9. Assume that you have a *virtual memory*, where the page size is 4096 bytes. The virtual memory address is 32-bits and the physical memory address is 18-bits, meaning that $2^{18} = 26244$ bytes of data can be stored in the main memory.

   - How many bits does the *page offset* consist of?
   - How many *physical pages* exist?
   - How many *virtual pages* exist?
   - Where is the mapping between virtual page numbers and physical page numbers stored? Who or what is responsible for updating this mapping?
   - Who or what is responsible for translating between virtual page numbers and physical page numbers?