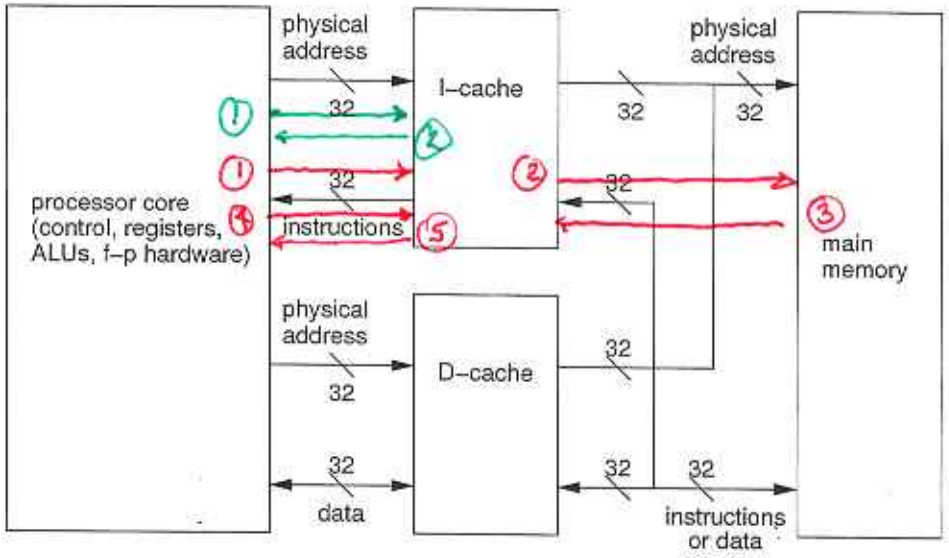
**A simple computer with one level of cache, and no address translation.** The system shown on page 1 is quite complicated. It makes sense to start studying memory organization by looking at a simpler system, such as the one shown below. Such a memory organization is what you would see in many 32-bit embedded computer systems.




– cache hit!
– cache miss!

Author: Steve Norman.
Electronic copies of documents for this course can be found at
https://people.ucalgary.ca/~norman/encm369winter2019/

Monday, March 18, 2019

## Caches small compared to main memory

A typical new laptop → 4 GB main memory (DRAM)
3 MB cache (SRAM)

- cache capacity is about 0.075% of main memory.

Too small to be useful? (lots of cache misses?)

Nope!

## Locality of reference

→ a property most programs have

Two types...

### 1. Temporal locality

Related to time. If a memory word is accessed, chances are that it will happen again soon.

E.g.,
- stack slots
- frequently-accessed global variables
- instructions in loops
- instructions in frequently-called procedures.

### 2. Spatial locality

Nearby words might be accessed

E.g.,
- sequential instruction flow
- current stack frame
- sequential access to array elements.

High locality of reference yields high cache hit rates (95-99%).

# Simple example of cache organization

The direct-mapped cache.

Definitions:

**Direct-mapped cache:** a main memory word with a given address maps to one-place in the cache.

**Byte offset:** part on an address to select a byte within a word (2 bits for 4-byte words)

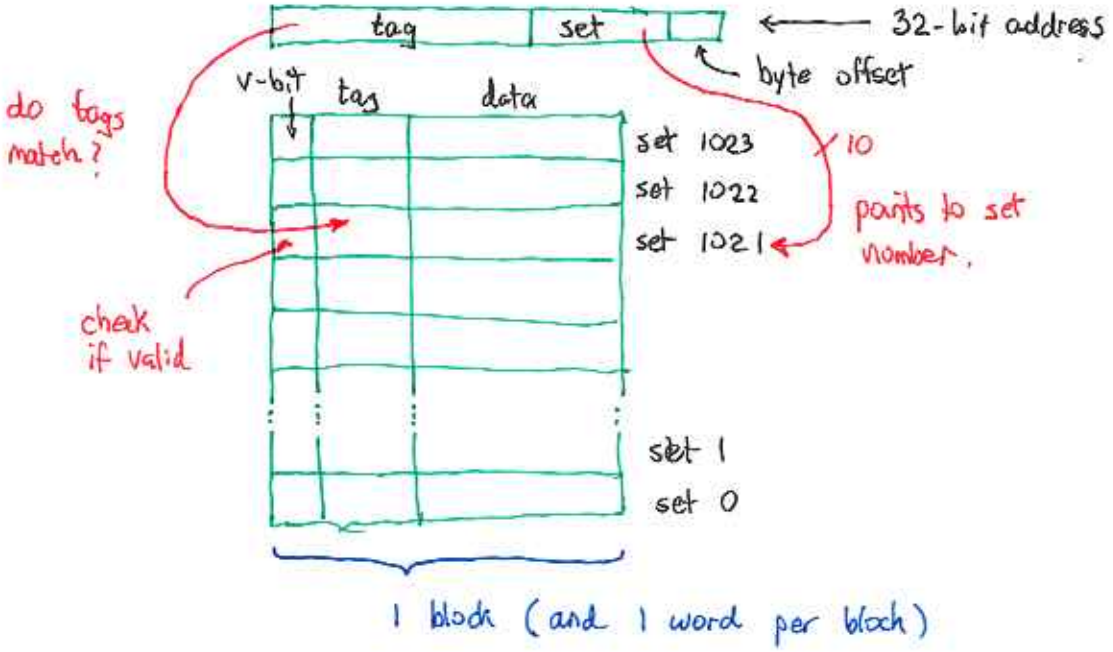
**Set:** part of an address used to decide which set (or "row") in the cache to search for a memory word.

**Tag:** part of an address to compare with tag bits for words previously copied.

**Valid bit: (or v-bit)** 1 if a cache entry is valid, 0 otherwise (starts as all zeros)

**Example:** a direct-mapped cache with 1024 sets.

Here's a simplified view of a direct-mapped cache with one-word blocks.



- This cache has 1024 sets × (32 + 20 + 1) bits per set.

Suppose we have an I-cache and processor tries to fetch an instruction at 0x0040_3570.

Address broken down as:

| 0000_0000_0100_0000_0011 | 0101_0111_00 | 00 |
|---|---|---|
| 20-bit tag<br>0x00403 | 10-bit set<br>0x15c, $348_{10}$ | 2-bit byte offset (not used in word searches) |

Line number (set) 348 in cache is checked:

- If v = 1 and tag = 0x00403, cache hit!
  - → data word from set 348 given to the processor.
- If v = 0 or tag ≠ 0x00403, cache miss.
  - → processor stalls; memory word at 0x0040_3570 read into cache set 348, v-bit set to 1, tag set to 0x00403.
  - → fetch retried.

(Note this simple cache takes advantage of temporal locality, not spatial)

Example: (same cache with 1024 sets)

Monitor D-cache activity (assume initially empty)

```
        addi  $t0, $zero, 5
loop:   beq   $t0, $zero, done
        lw    $t1, 0x4($zero)
        lw    $t2, 0xc($zero)
        lw    $t3, 0x8($zero)
        addi  $t0, $t0, -1
        j     loop
done:
```

first time through, 3 cache misses.

Address : 0x0000_0004