

50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

The Memory Hierarchy

Natalie Agus (Fall 2018)

1 Introduction

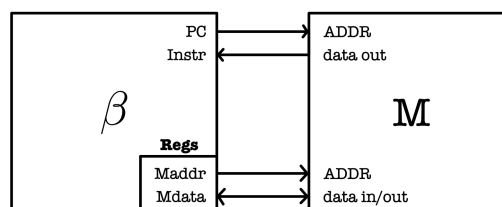


Figure 1

We have learned that β communicates with the Memory. The registers in β are expensive, so we have limited numbers of them but they are fast. The materials that make up the physical memory (RAM) is way cheaper so we can afford a bigger storage space, but it is much slower in comparison to the registers in β . The access to the memory is the bottleneck in the computer performance.

Our goal is to have a large and fast memory space.

2 Registers vs Memory vs Disk Technology

2.1 SRAM: Tech of Regs (fastest, most exp)

SRAM is one of the technology behind static bistable storage element (flip-flop). It is made up of **6-transistors and amp senses** in each cell, which is what made it expensive. Each cell stores one bit, the loop between two inverters can store a bit as long as it is plugged in (connected to a power source). Figure 2 shows the structure of one single SRAM cell and how one can READ or WRITE into a single SRAM cell. The word line is driven into HIGH voltage during both READ or WRITE, such that the cell is connected to both bit lines.

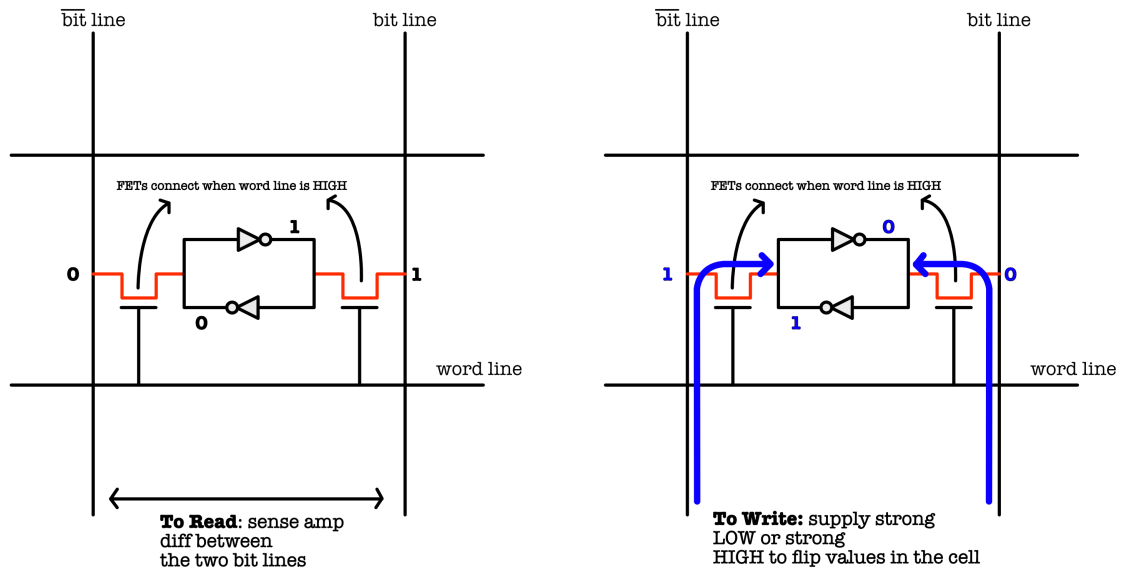
SRAM CELL

Figure 2

2.2 DRAM: Tech of Memory (slower, cheaper)

A DRAM cell is made of just a single FET and a single capacitor. Figure 4 shows how bits of address can access a specific chunk of cells in DRAM into the output. The problem with DRAM is that the capacitor will lose charge, so DRAM has to be **refreshed** very frequently to keep the data intact. This refresh cycles cause DRAM to be significantly slower than SRAM. Similar to SRAM, the word line is driven to HIGH when one wants to READ or WRITE.

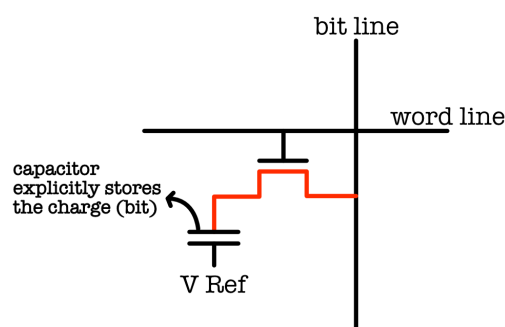
DRAM CELL

Figure 3

2.3 Decoding Address in with SRAM/DRAM cells

Billions of these SRAM or DRAM cells are assembled together to form a large memory unit. Each cell (or a group of 8 cells in byte addressing) can have specific addresses. To decode address, one can split the address into "row" and "column" address and read information out of the bitlines (the column, vertical lines in Figure 4). We often read hundreds of bits in parallel (one row contains hundreds of bits), and select which of the 32 bits or 64 bits we want using the 'column' address, depending on our device.

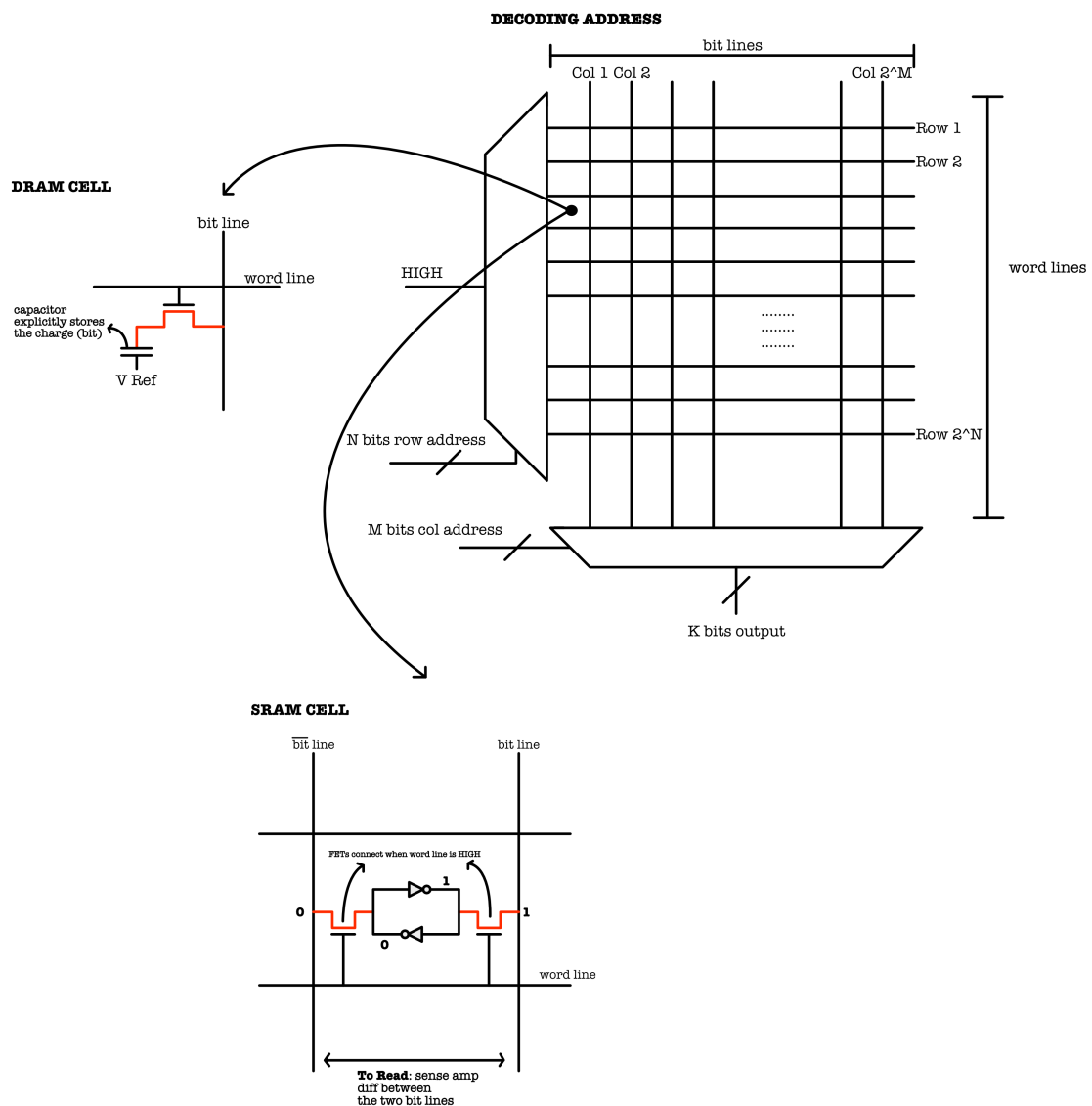


Figure 4

2.4 DISK: Tech of Secondary Storage (very slow, very cheap)

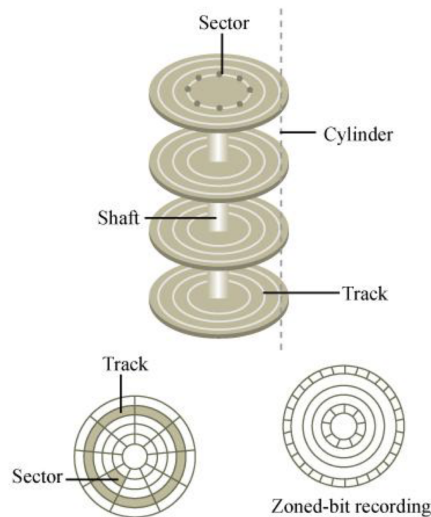


Figure by MIT OpenCourseWare.

Figure 5

A disk is typically used as secondary storage, and ideally it should not be accessed during run-time. It is made up of spinning magnetic medium. Unlike SRAM and DRAM, a disk is a **non volatile storage**, meaning that it can store information even after the power source is cut. As shown in Figure 5, it is made up of concentric circles, where each sector contains one bit. To READ or WRITE to a disk, the device has to mechanically move the head of the disk and access a particular sector.

3 Goal: Perform with SRAM speed at the cost of a DISK

The strategy to do this is to have a **memory hierarchy**:

1. Keep the most often used data at a special device made of SRAMS. We call this **cache**
2. Refer to main memory rarely
3. Refer to DISK even more rarely
4. It is possible because of **locality of reference**

4 The Locality of Reference (LOR)

The locality of reference states that:

Reference to memory location X at time t implies that reference to $X + \Delta X$ at $t + \Delta t$ becomes more probable as $\Delta X, \Delta t$ approaches zero.

Evidence that this is true in practice:

1. Local stack frame grows nearby to one another
2. Program instructions are near one another
3. Data (e.g: arrays) are also nearby one another)
4. Hence, memory reference patterns exhibit locality of reference

5 The Memory Hierarchy

In order to fulfil the goal in Section 3, the memory hierarchy in a computer is arranged as follows:

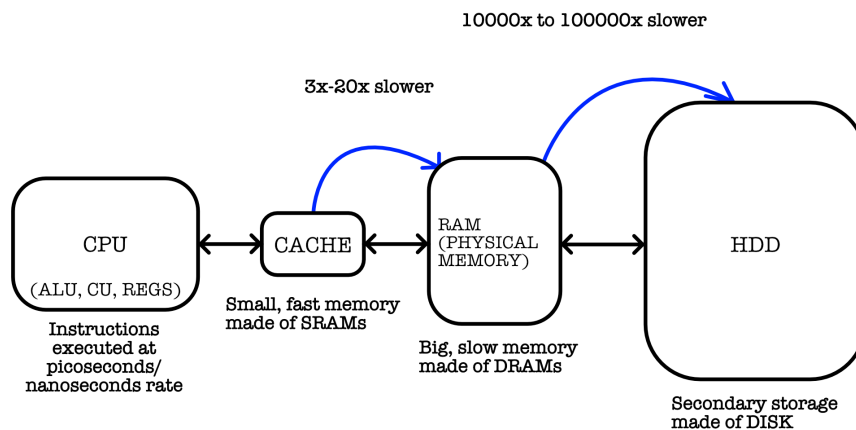


Figure 6

6 The Cache Idea

The cache contains temporary copies of selected memory locations A and its content $\text{Mem}[A]$. The cache principles work as follows:

1. Look for requested info (from CPU) in cache
2. If the info is in the cache, return the content to CPU
3. Otherwise (cache miss), look for the info in the physical memory, and subsequently, the disk

The average time-taken t_{ave} for info look-ups follows the formula:


$$t_{\text{ave}} = \alpha t_c + (1 - \alpha)(t_c + t_m) = t_c + (1 - \alpha)t_m,$$

where α is hit ratio (the amount of information found in cache), t_c is cache access time and t_m is memory access time.

7 Types of Caches

7.1 Fully Associative Cache (FA)

Characteristics of FA caches:

1. TAG contains the all bits of address A 
2. DATA contains all bits of content at A : $\text{Mem}[A]$
3. **Expensive**, made up of SRAMS and lots of other hardwares (comparator circuit at each row)
4. **Real fast** because it does PARALLEL lookup. FA cache is the **gold standard** on how well a cache should perform
5. **Flexible** because memory address + content can be stored on any TAG-DATA row.
6. However one **needs replacement strategy** to decide which of the cache line to write to when cache is full

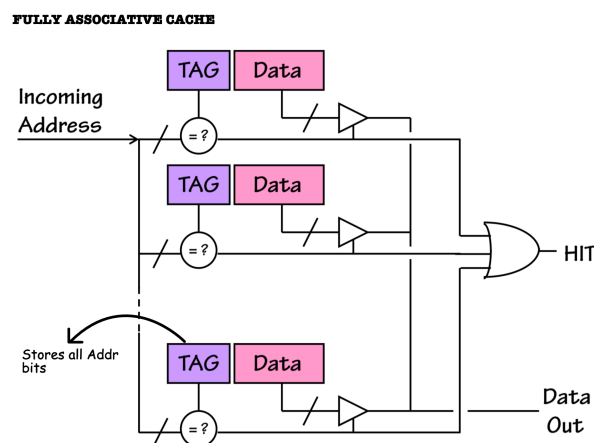


Figure 7

7.2 Direct Mapped Cache (DM)

Characteristics of DM caches:

1. TAG contains T-upper bits of address A ,
2. DATA contains all bits of content at A : $\text{Mem}[A]$. The lower K -bits of A decides which 'row' of DM cache we are looking for. A unique combination of K -bits of A is mapped to **exactly** one of the rows of DM cache, hence making it inflexible.
3. Although it is also made of SRAMS, it is cheaper than FA caches because it is made up of less hardware (only one comparator circuit per DM cache),
4. There is NO PARALLELISM, but **fast mapping** between address and cache line index
5. Hence, DM cache suffers contention (collision problem) in mapping, meaning that two different addresses can be mapped to the same location of both has the same K -lower bits. Selecting the K -lower bits for mapping is better than selecting the T upper bits **due to locality of reference**, but does not completely eliminate contention.

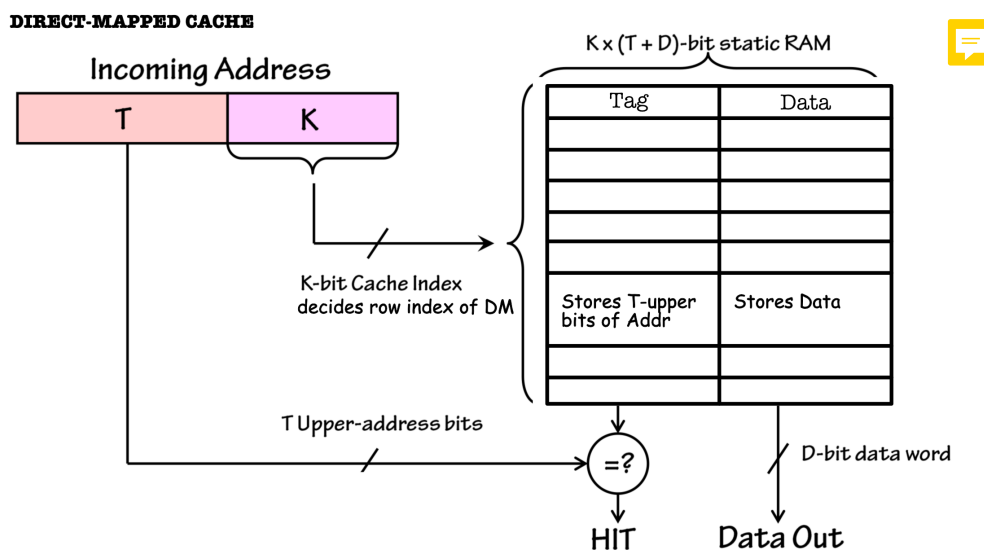


Figure 8