Outline of Lecture

- **Levels of Memory Hierarchy**
- **Cache Memory**

Importance of Memory

- In the course so far, we have looked only at the processor part of a computer system:
  - ➔ CPU performance, ISA, Datapath, Pipelined execution.
- *Why do we care about memory hierarchy?*
  - ➔ In 1980: no cache in microprocessors.
  - ➔ In 1995: 2-level caches, e.g., 60% of transistors on the Alpha processor is for the cache system.
  - ➔ *150 clock cycles for a cache miss !!!*
  - ➔ Memory improvement has not kept at all with the improvement of processors.

Memory performance is extremely crucial to the overall computer performance.

Exploiting Memory Hierarchy

- **Users want large and fast memories!**

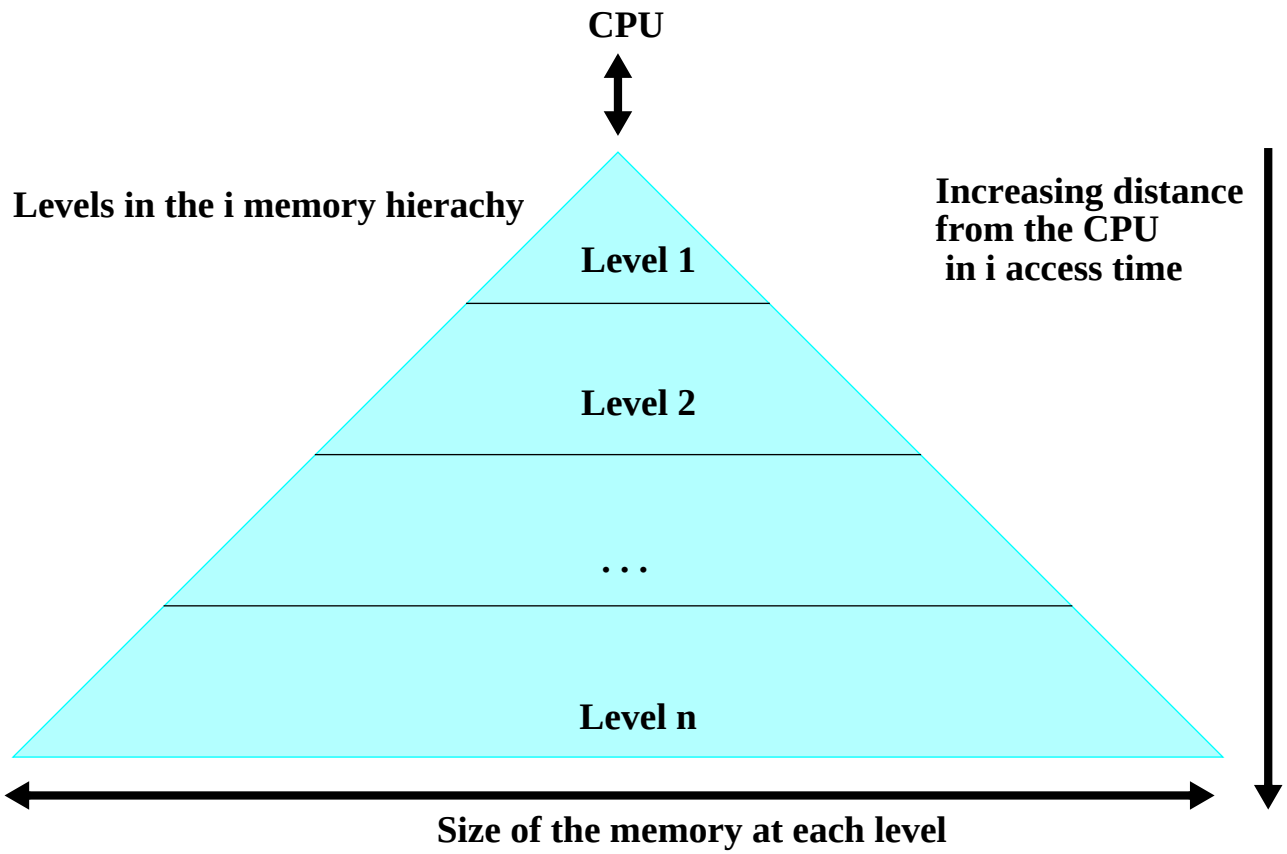
  - SRAM access times are 2 - 25 ns at cost of $100 to $250 per Mbyte.

  - DRAM access times are 60 - 120 ns at cost of $5 to $10 per Mbyte.

  - Disk access times are 10 - 20 Million ns at cost of $.10 to $20 per Mbyte.

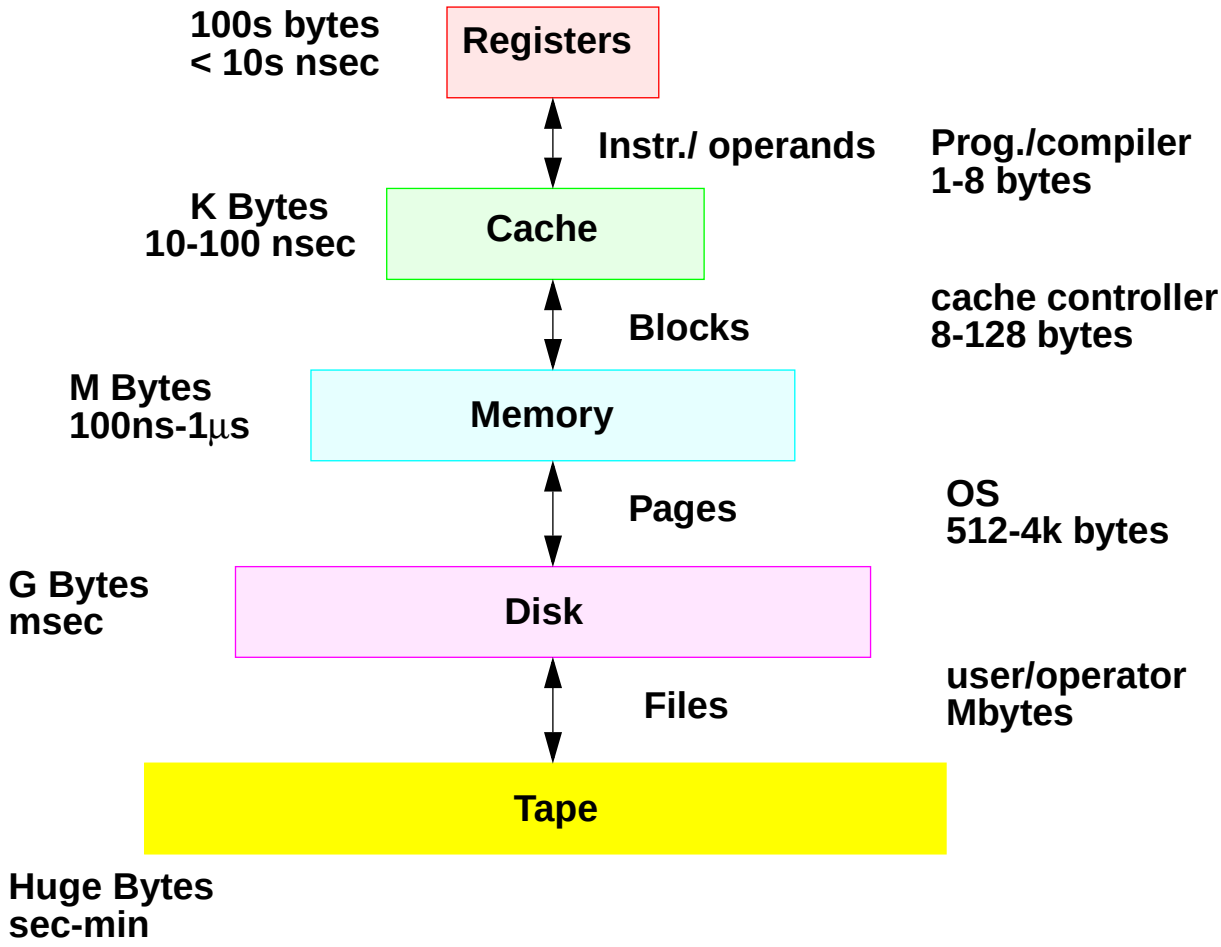
- **Try and give it to them anyway**

  - build a memory hierachy.



Memory Hierarchy

- The memory is organized into many levels



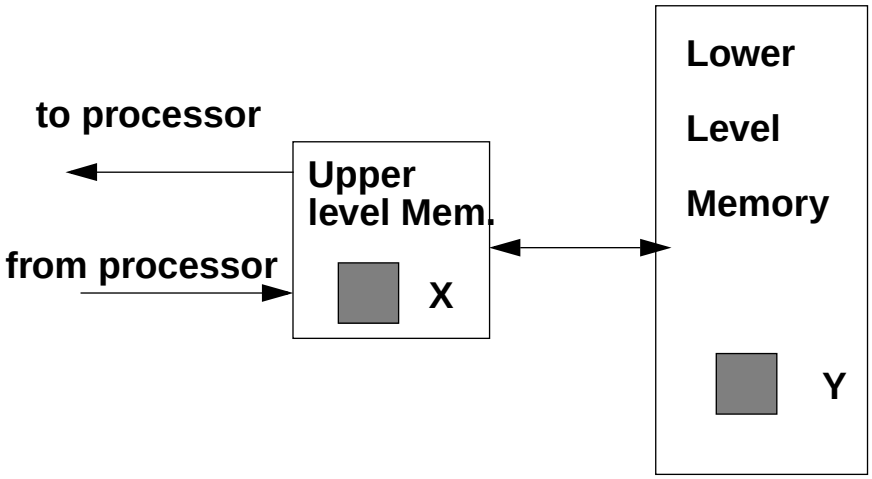
The Principle of Locality

- The principle of locality:
  - ➔ A program access a relatively small portion of the address space at any instant (instructions and data).
- Two different types of locality:
  - ➔ Temporal locality (locality in time): If an item (instruction or data) is referenced, it will tend to be referenced again soon.
  - ➔ Spatial locality (locality in space): If an item (instruction or data) is referenced, items whose addresses are close by tend to be referenced soon.

Memory Terminology

- **Block:** minimum unit of data

- **Hit:** data appears in some block in the upper level (example: Block X)



➔ **Hit Rate:** the fraction of memory accesses found in the upper level (e.g., in the cache).

➔ **Hit Time:** Time to access the upper level which consists of access time + Time to determine hit/miss.

Memory Terminology

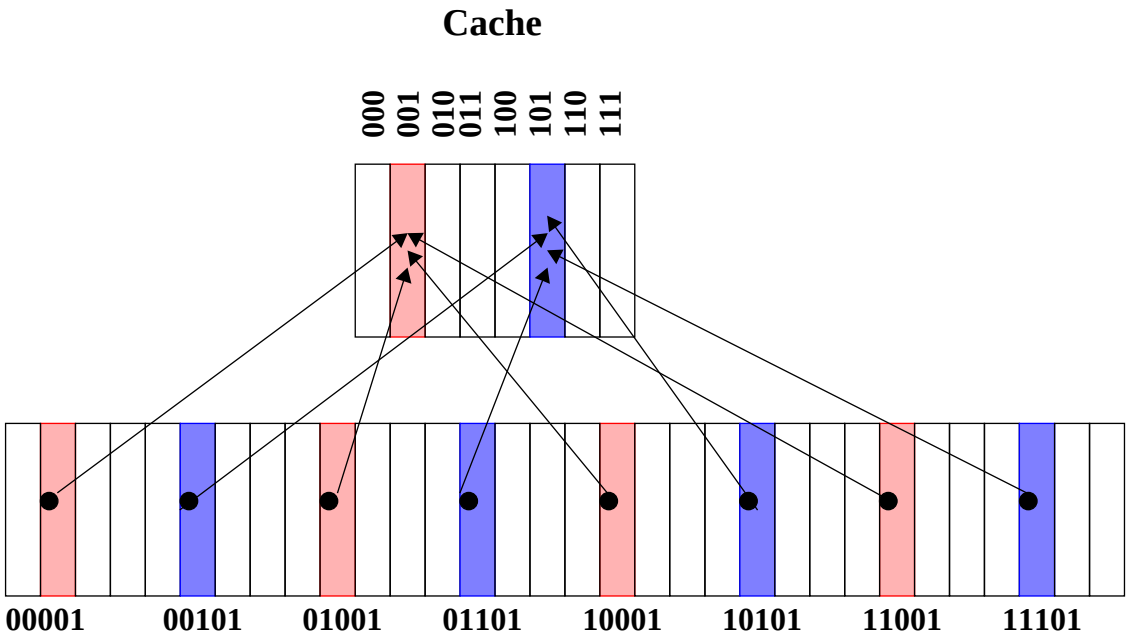
- **Miss:** data needs to be retrieved from a block in the lower level (example: Block Y)
  - ➔ **Miss Rate:** 1 - Hit Rate.
  - ➔ **Miss Penalty:** Time to replace a block in the upper level + time to deliver the block to the processor.
- **Hit Time << Miss Penalty**

$$Memory\ \ stall\ \ cycles\ =\ instruction\ \ count \times Memory\ \ references$$

$$per\ \ instruction \times Miss\ \ rate \times Miss\ \ penalty$$

Questions for Cache Memory

➔ ***Direct mapped*: Each block has only one place to appear in the cache. The mapping is (*block address MOD number of blocks in cache*).**



✣ **It is easy to locate blocks in the cache (only one possibility)**

- **Certain blocks cannot be simultaneously present in the cache (they can only have the same cache location)**

➔ ***Fully associative***: **A block can be placed anywhere in the cache.**

✣ **No restriction on the placement of blocks. any combination of blocks can be simultaneously present on the cache.**

- **Quite costly (hardware and time) to search for a block in the cache.**

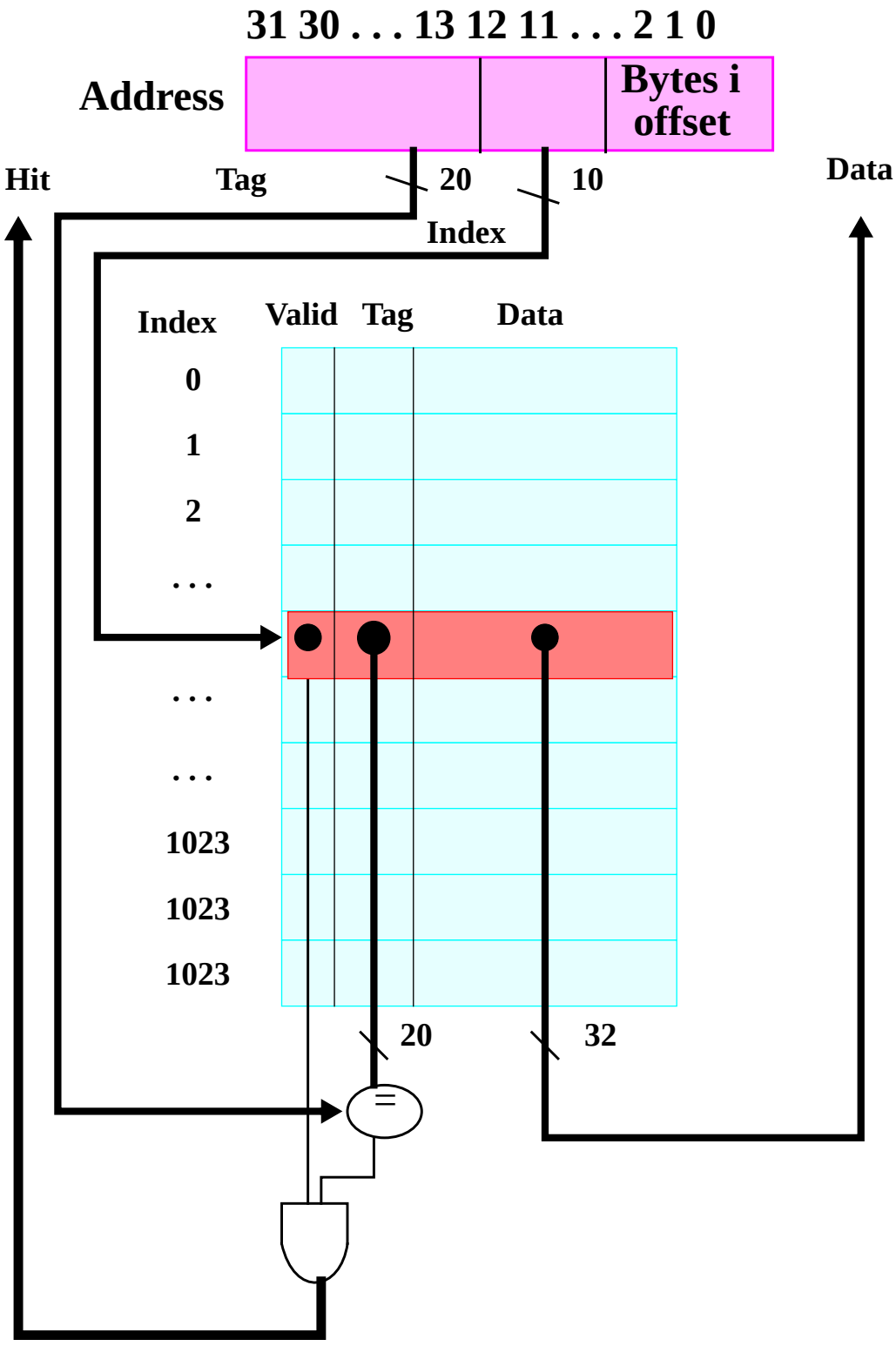
➔ ***Set associative***: **Each block has only a certain number of places to appear in the cache. The mapping is (*block address MOD number of sets in cache*).**

✣ **A very good compromise between direct mapped and fully associative caches.**

Most microprocessors use direct mapped, 2 way set associate, or 4-way set associative caches.

How a block is found if it is in the cache? (*block identification*)

- Each block in the cache has an address ***tag*** that gives the block address.
- We further need an ***index*** to point to the appropriate location within the cache.
- We also need a ***valid*** bit to indicate whether the item in the cache is valid or not.
- ***Example***: Direct mapped cache:
  - ➔ Cache index selects a location from cache
  - ➔ The tag is compared to that in the cache location.
  - ➔ Data is selected based on the tag result.


31 30 . . . 13 12 11 . . . 2 1 0
Address
Bytes i offset
Hit
Tag
20
10
Data
Index
Index
Valid
Tag
Data
0
1
2
. . .
. . .
. . .
1023
1023
1023
20
32
=

Which block should be replaced on a cache miss? (*block replacement*)

- When a miss occurs, the cache controller must select a block to be replaced with the desired data (with direct mapped there is only one choice).

  ➔ ***Random:*** To spread allocation uniformly, candidate blocks are randomly selected.

  ➔ ***Least-recently used (LSU):*** The block replaced is the one that has been unused for the longest time.

What happens on a write? (*The write strategy*)

- **There are two basic options when writing to the cache:**

  - ***Write through (or store through):*** **The information is written to both the block in the cache *and* to the block in the lower-level memory.**

  - ***Write back (store back):*** **The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.**

Further Reading

Chapter 6. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface*. Morgan Kaufman (page 540-555).**