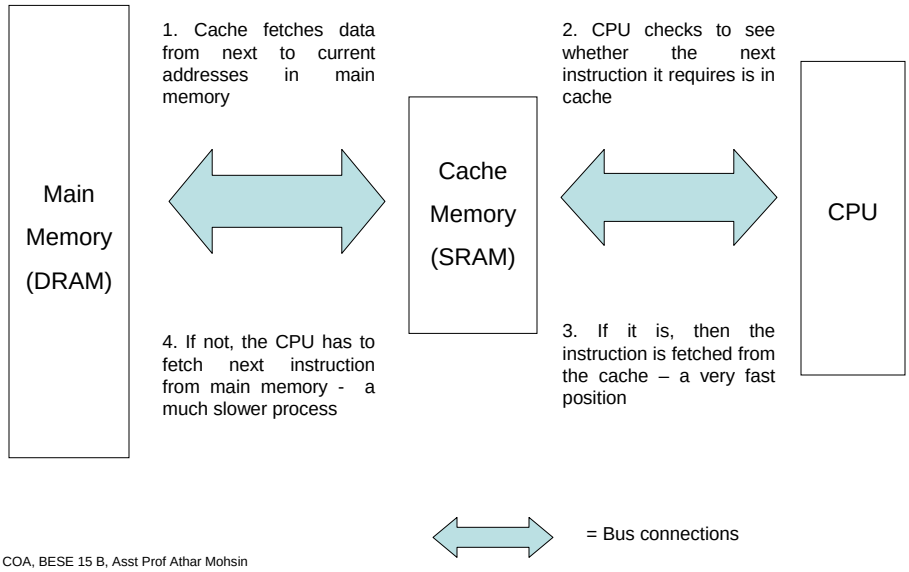
Cache Memories

Lecture 14

The operation of cache memory

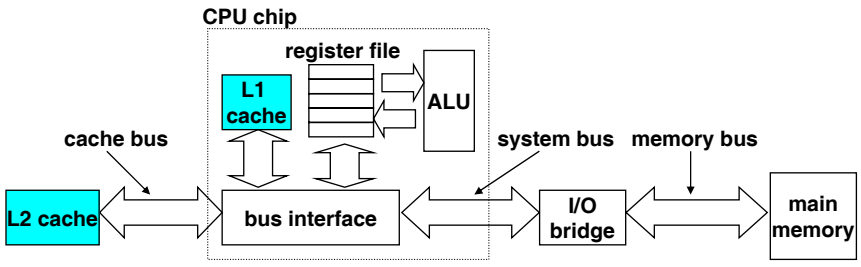


Memory Hierarchy Basis

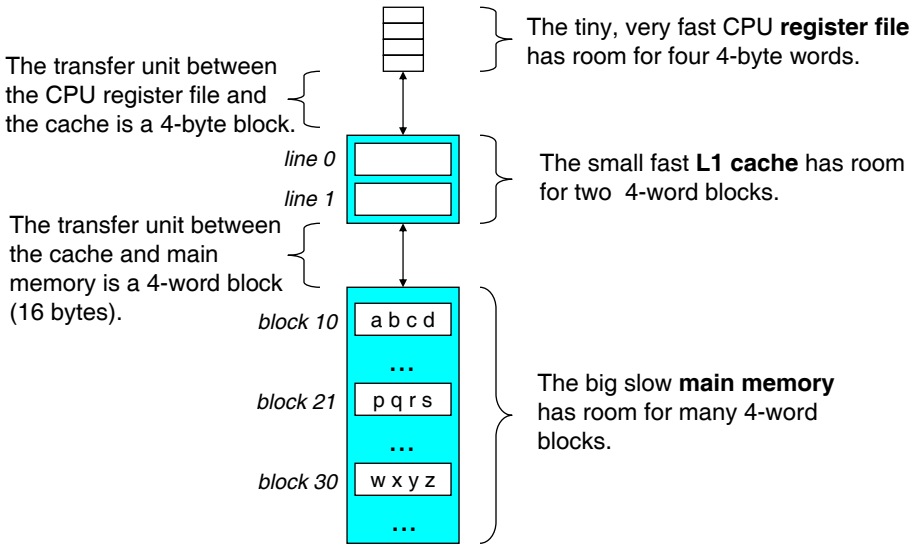
- Disk contains everything.
- When Processor needs something, bring it into all higher levels of memory.
- Cache contains copies of data in memory that are being used.
- Memory contains copies of data on disk that are being used.
- Entire idea is based on Temporal Locality

Cache Memories

- Hold frequently accessed blocks of main memory
  - CPU looks first for data in L1, then in L2, then in main memory.
- Typical bus structure:



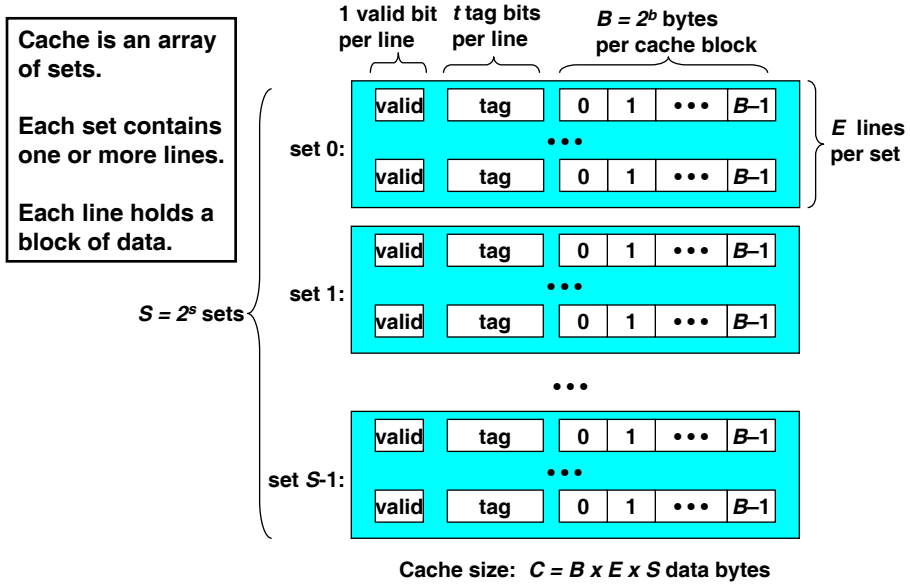
Inserting an L1 Cache Between the CPU and Main Memory



Cache Terminologies

- Cache Line/Block
  - Cache memory is subdivided into cache lines
  - Cache Lines / Blocks:
    - The smallest unit of memory than can be transferred between the main memory and the cache.
- Tag / Index
  - Every address field consists of two primary parts:
    - A dynamic (tag) which contains the higher address bits, and
    - A static (index) which contains the lower address bits
      - The Dynamic Tag may be modified during run-time while the Static Tag one is fixed.

General Org of a Cache Memory



Cache Terminologies

- Valid / Dirty bit
  - a valid bit is needed to indicate whether or not the slot holds a line that belongs to the program being executed
  - a dirty bit keeps track of whether or not a line has been modified while it is in the cache.
- Cache Hit:
  - A request to read from memory, which can satisfy from the cache without using the main memory.
- Cache Miss:
  - A request to read from memory, which cannot be satisfied from the cache, for which the main memory has to be consulted.
    - Hit Ratio: fraction of memory accesses found in upper level
    - Miss Rate: *1 - hit rate*

Cache Operation

- General scenario:
  - processor needs certain block
  - search cache for block
  - if hit send to processor
  - if miss, bring in block from lower level (possibly replacing current block in cache) and send to processor.

The Cache

- Cache generally contains a “tag array” and “the data array”
  - The tag array contains the addresses of the data contained in the cache
  - The data array contains the data itself
- Dividing the cache into tag and data array reduces the access time as
  - Tags array contains fewer bits than data array – less time
  - Once the tag array has been accesses
    - Its output must be compared to the address of the memory reference to determine a hit

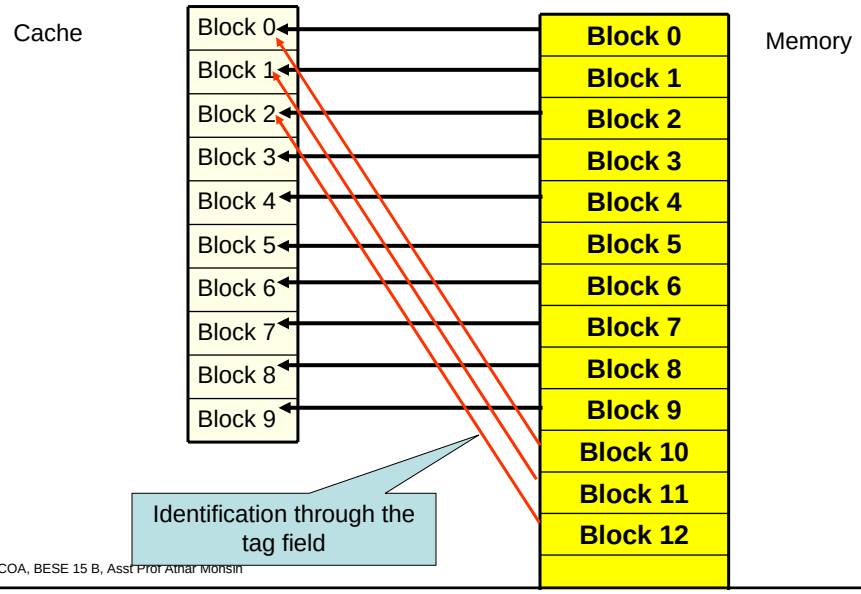
Describing the Cache

- Capacity
  - It is simply the amount of data that can be stored in the cache
    - A 32 KB can store 32 KB of the data
- Line length
  - It is the cache's block size
    - The size of the data that are brought into and thrown out of the cache in response to a miss
      - When a cache with 32 byte cache lines has a cache miss, it brings a 32 byte block of data containing the address of miss into cache, evicting a 32 byte to make room
- Associatively
  - It determine how many locations within the cache may contains a given memory address

Direct Cache Memory

- The simplest cache mapping scheme is *direct mapped cache*.
  - Each memory address can only be stored in one location in the cache
  - Direct mapped cache consisting of *N* blocks of cache
    - block *X* of main memory maps to block *Y* of cache.
      - e.g: if we have 10 blocks of cache, block 7 of cache may hold blocks 7, 17, 27, 37, . . . of main memory.
  - Once a block of memory is copied into cache, a *valid* bit is set for the cache block
    - to let the system know that the block contains valid data.

Direct Mapped Cache



Direct Mapped Cache Memory

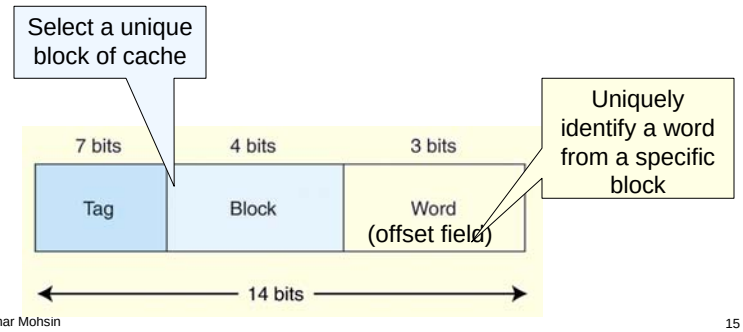
- The diagram shows of what cache looks like.

Block	Tag	Data	Valid
0	00000000	words A, B, C,...	1
1	11110101	words L, M, N,...	1
2	----------		0
3	----------		0

- Block 0 contains
  - multiple words from main memory,
  - identified with the tag 00000000
  - Block 1 contains words identified with the tag 11110101.
- The other two blocks are not valid.

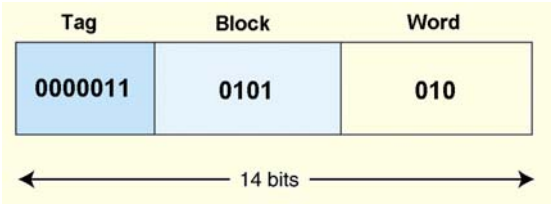
Direct Mapped Cache Memory

- The size of each field into which a memory address is divided depends on the size of the cache.
  - Suppose memory consists of 2^{14} words, while cache has $16 = 2^4$ blocks, and each block holds $8=2^3$ words.
    - Thus memory is divided into $2^{14} / 2^3 = 2^{11}$ blocks.
- For our field sizes, we need 4 bits for the block, 3 bits for the word, and the tag is what's left over:



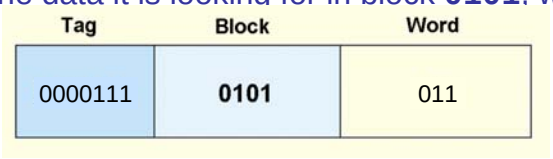
Cache Memory

- As an example, suppose a program generates the address **1AAh**.
  - In 14-bit binary, this number is: **00**0001**1 0101 010**.
    - The first 7 bits of this address go in the tag field,
    - the next 4 bits go in the block field, and
    - the final 3 bits indicate the word within the block.



Cache Memory

- If subsequently the program generates the address **1AB**, **0000011 0101 011**
  - it will find the data it is looking for in block **0101**, word **011**.



- if the program generates the address, **3ABh**, instead
  - 3ABh = 0000111 0101 011
  - the block loaded for address **1AA** would be evicted from the cache, and
  - replaced by the blocks associated with the **3AB** reference.

Example

- Suppose a computer using direct mapped cache has 2^{20} words of main memory, and a cache of 32 blocks, where each cache block contains 16 words.
  - a. How many blocks of main memory are there?
  - b. What is the format of a memory address as seen by the cache, i.e., what are the sizes of the tag, block, and word fields?
  - c. To which cache block will the memory reference 0DB63h map?
- *Ans.*
  - a. $2^{20}/2^4 = 2^{16}$
  - b. 20 bit addresses with 11 bits in the tag field, 5 in the block field, and 4 in the word field
  - c. 0DB63h = 00001100101-10110 - 0111, which implies Block 22