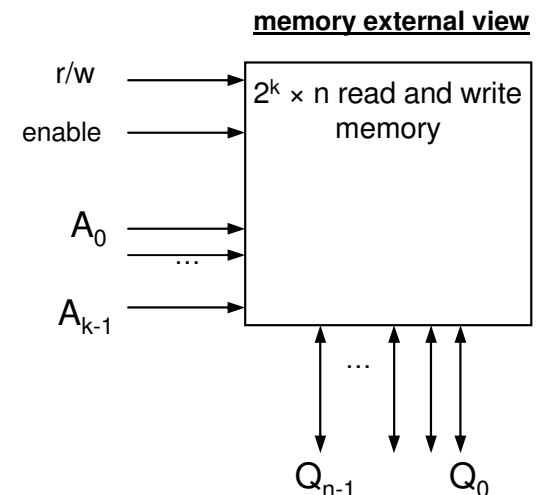
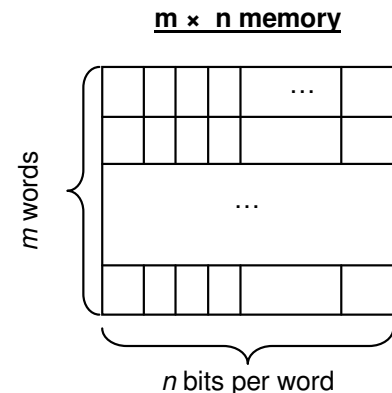


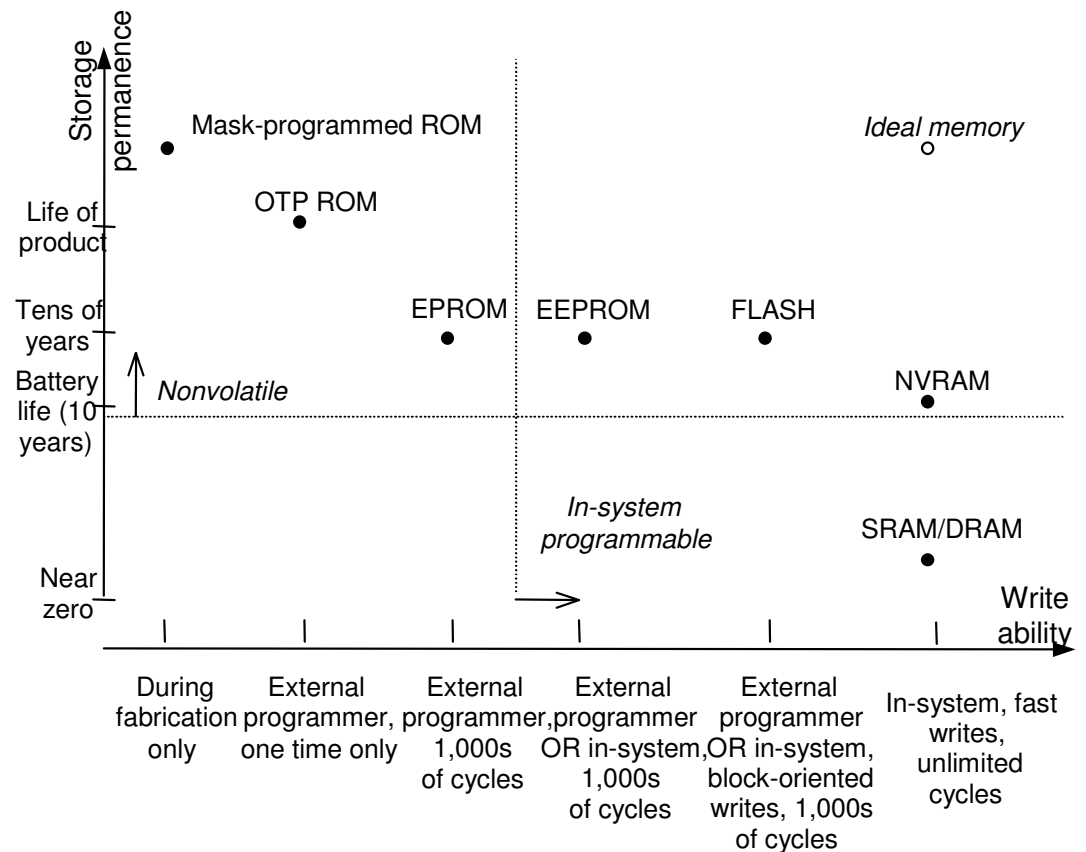
# Memory: basic concepts

- Stores large number of bits
  - $m \times n$ :  $m$  words of  $n$  bits each
  - $k = \text{Log}_2(m)$  address input signals
  - or  $m = 2^k$  words
  - e.g., 4,096 x 8 memory:
    - 32,768 bits
    - 12 address input signals
    - 8 input/output data signals
- Memory access
  - r/w: selects read or write
  - enable: read or write only when asserted
  - multiport: multiple accesses to different locations simultaneously



# Write ability/ storage permanence

- Traditional ROM/RAM distinctions
  - ROM
    - read only, bits stored without power
  - RAM
    - read and write, lose stored bits without power
- Traditional distinctions blurred
  - Advanced ROMs can be written to
    - e.g., EEPROM
  - Advanced RAMs can hold bits without power
    - e.g., NVRAM
- Write ability
  - Manner and speed a memory can be written
- Storage permanence
  - ability of memory to hold stored bits after they are written

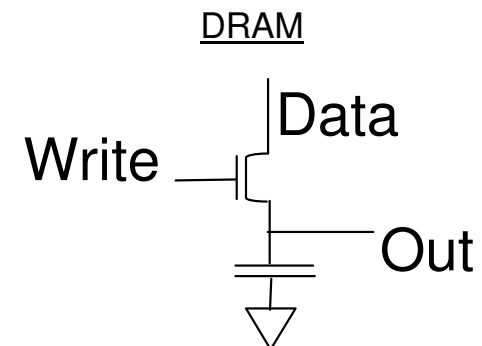
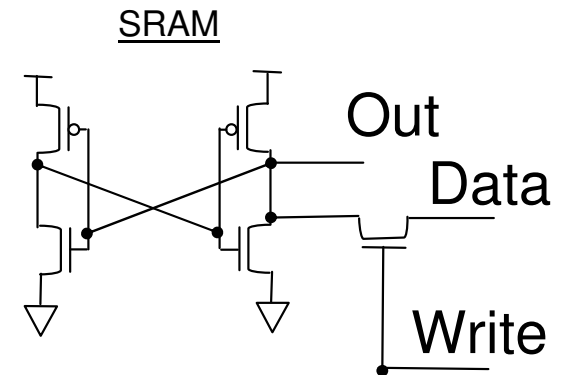


Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale).

# Basic types of RAM

- SRAM: Static RAM
  - Memory cell uses flip-flop to store bit
  - Requires 6 transistors
  - Holds data as long as power supplied
- DRAM: Dynamic RAM
  - Memory cell uses MOS transistor and capacitor to store bit
  - More compact than SRAM
  - “Refresh” required due to capacitor leak
    - word’s cells refreshed when read
  - Typical refresh rate 15.625 microsec.
  - Slower to access than SRAM

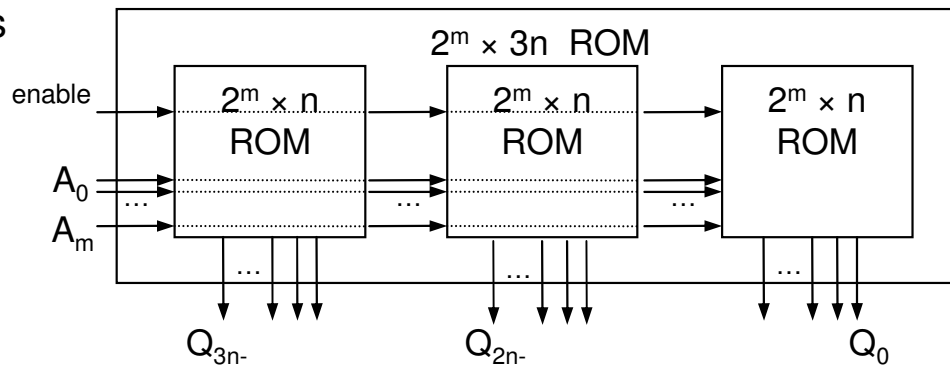
## memory cell internals



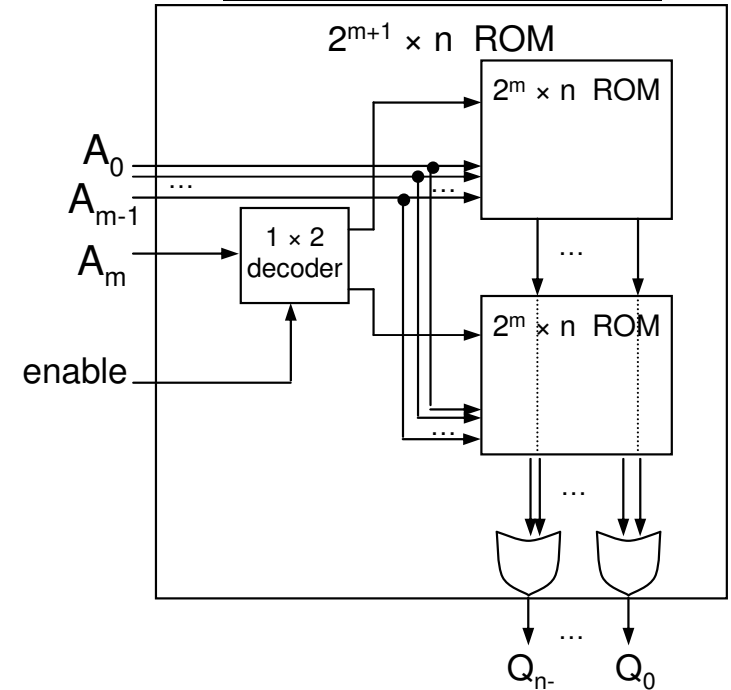
# Composing memory

- Memory size needed often differs from size of readily available memories
- When available memory is larger, simply ignore unneeded high-order address bits and higher data lines
- When available memory is smaller, compose several smaller memories into one larger memory
  - Connect side-by-side to increase width of words
  - Connect top to bottom to increase number of words
    - added high-order address line selects smaller memory containing desired word using a decoder
  - Combine techniques to increase number and width of words

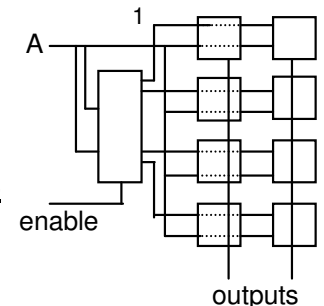
**Increase width of words**



**Increase number of words**

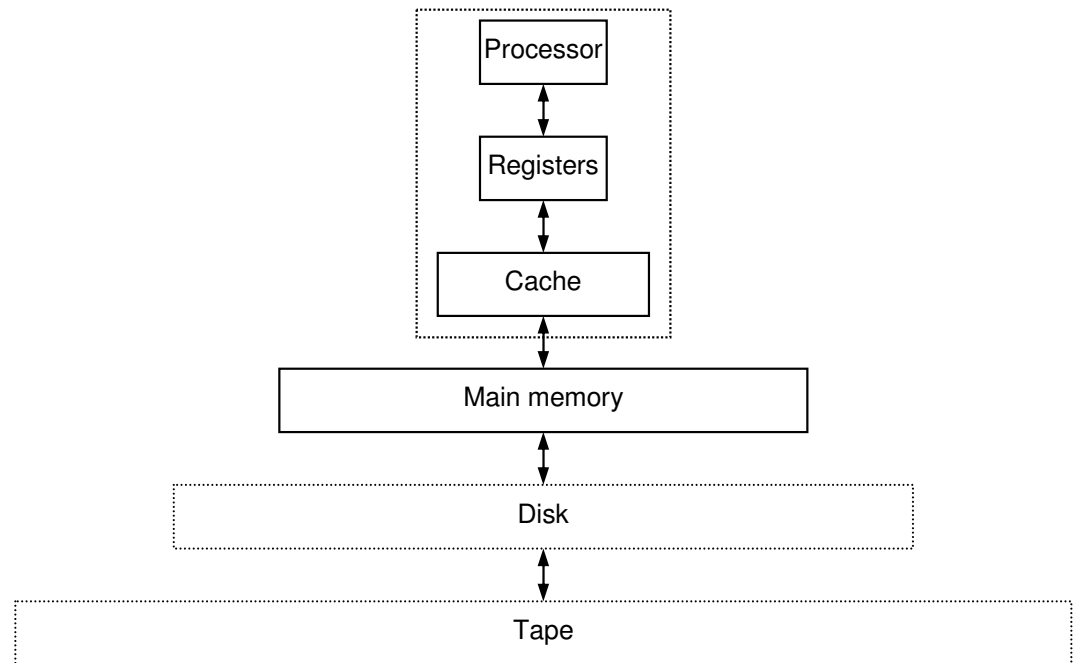


**Increase number and width of words**



# Memory hierarchy

- Want inexpensive, fast memory
- Main memory
  - Large, inexpensive, slow memory stores entire program and data
- Cache
  - Small, expensive, fast memory stores copy of likely accessed parts of larger memory
  - Can be multiple levels of cache



# Cache

---

- **Usually designed with SRAM**
    - faster but more expensive than DRAM
  - **Usually on same chip as processor**
    - space limited, so much smaller than off-chip main memory
    - faster access ( 1 cycle vs. several cycles for main memory)
  - **Cache operation:**
    - Request for main memory access (read or write)
    - First, check cache for copy
      - cache hit
        - copy is in cache, quick access
      - cache miss
        - copy not in cache, read address and possibly its neighbors into cache
  - **Several cache design choices**
    - cache mapping, replacement policies, and write techniques
-

# Cache mapping

---

- Problem:
  - Cache memory is much smaller than main memory
- Solution:
  - First: Make the cache very wide (like 4 or 8 times as wide as the main memory)
  - Second: Use PART of the main memory address as the address in the cache
  - Third: Store the rest of the main memory address with the data

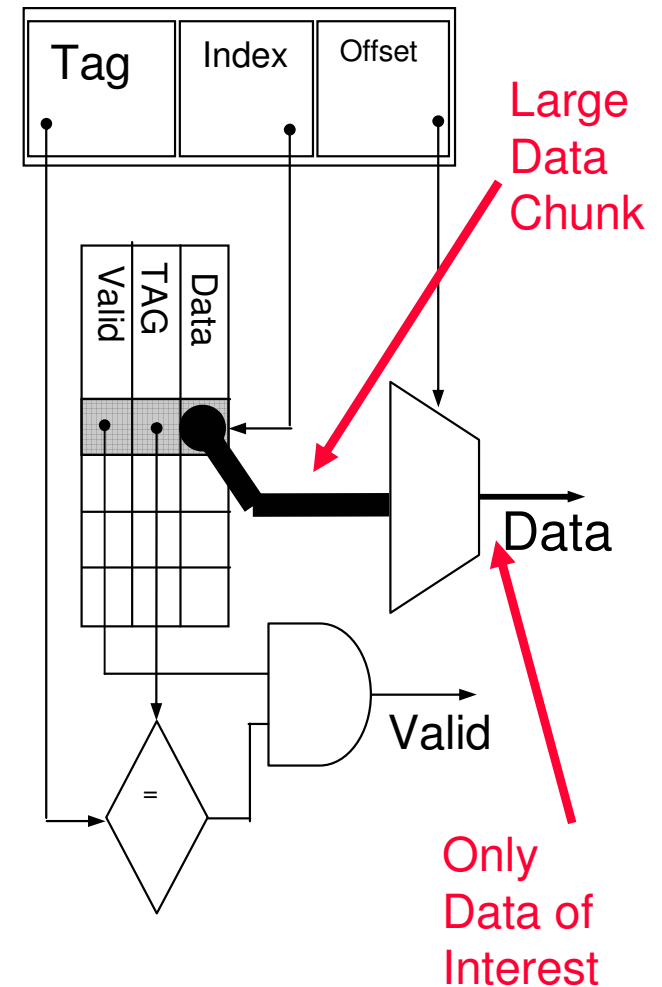
# Cache mapping

---

- Three basic techniques:
  - Direct mapping
  - Fully associative mapping
  - Set-associative mapping
- Caches partitioned into indivisible blocks or lines of adjacent memory addresses
  - usually 4 or 8 addresses per line

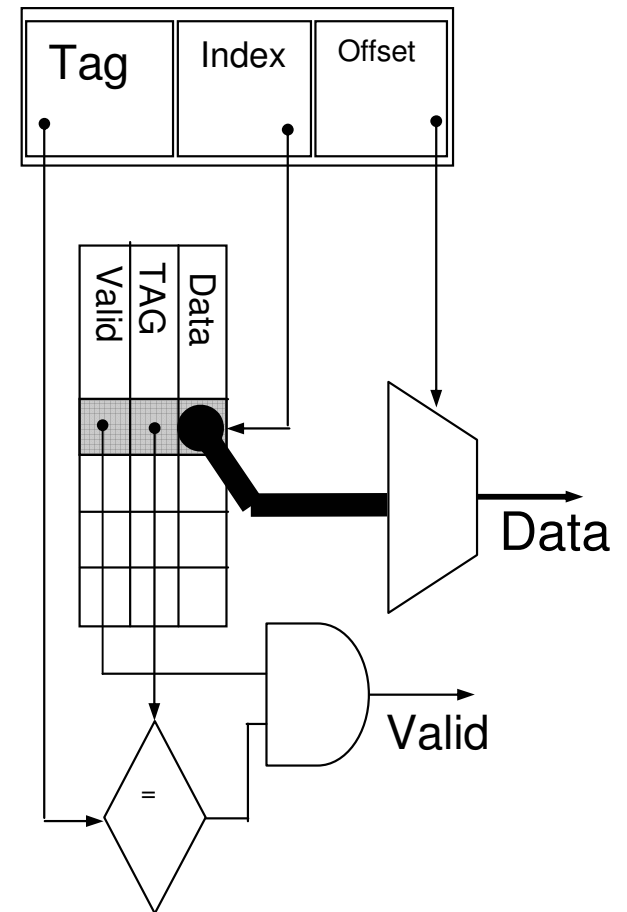
# Direct mapping

- Main memory address divided into 2 fields
  - Index
    - cache address
    - number of bits determined by cache size
  - Tag
    - compared with tag stored in cache at address indicated by index
- Valid bit
  - indicates whether data in slot has been loaded from memory
- Offset
  - used to find particular word in cache line



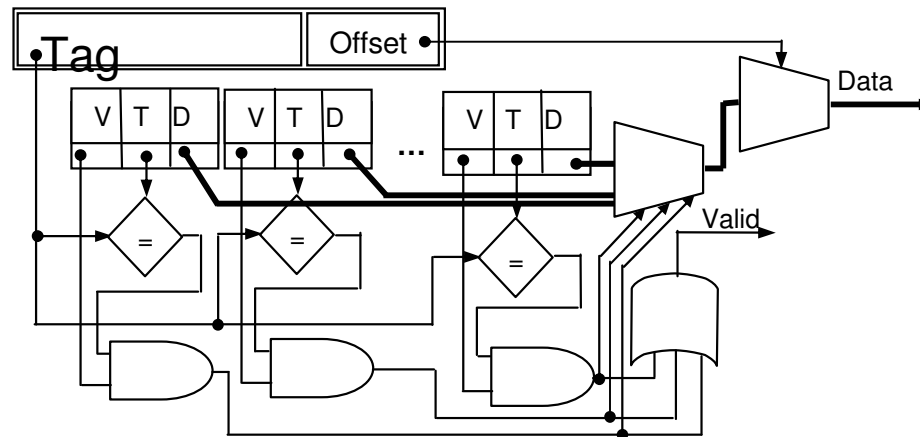
# ICE!

- If you have the value “0xAAAA” stored at main memory location “0x3f0158b4” what will the **TAG**, **Index**, and **Offset** be if you have a cache with the following specifications:
- Eight data values per line
- 1024 lines (addresses)



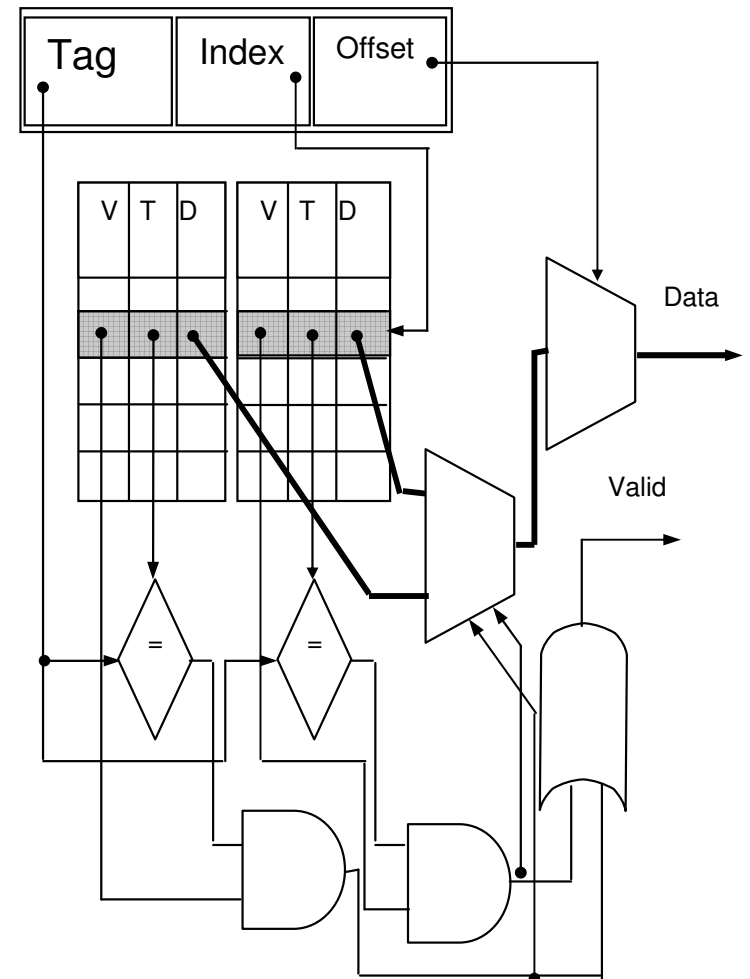
# Fully associative mapping

- Complete main memory address stored in each cache address
- All addresses stored in cache simultaneously compared with desired address
- Valid bit and offset same as direct mapping



# Set-associative mapping

- Compromise between direct mapping and fully associative mapping
- Index same as in direct mapping
- But, each cache address contains content and tags of 2 or more memory address locations
- Tags of that **set** simultaneously compared as in fully associative mapping
- Cache with set size N called N-way set-associative
  - 2-way, 4-way, 8-way are common



# Cache-replacement policy

---

- Technique for choosing which block to replace
  - when fully associative cache is full
  - when set-associative cache's line is full
  - Direct mapped cache has no choice
- Random
  - replace block chosen at random
- LRU: least-recently used
  - replace block not accessed for longest time
- FIFO: first-in-first-out
  - push block onto queue when accessed
  - choose block to replace by popping queue

# Cache write techniques

---

- When written, data cache must update main memory
- Write-through
  - write to main memory whenever cache is written to
  - easiest to implement
  - processor must wait for slower main memory write
  - potential for unnecessary writes
- Write-back
  - main memory only written when “dirty” block replaced
  - extra dirty bit for each block set when cache block written to
  - reduces number of slow main memory writes

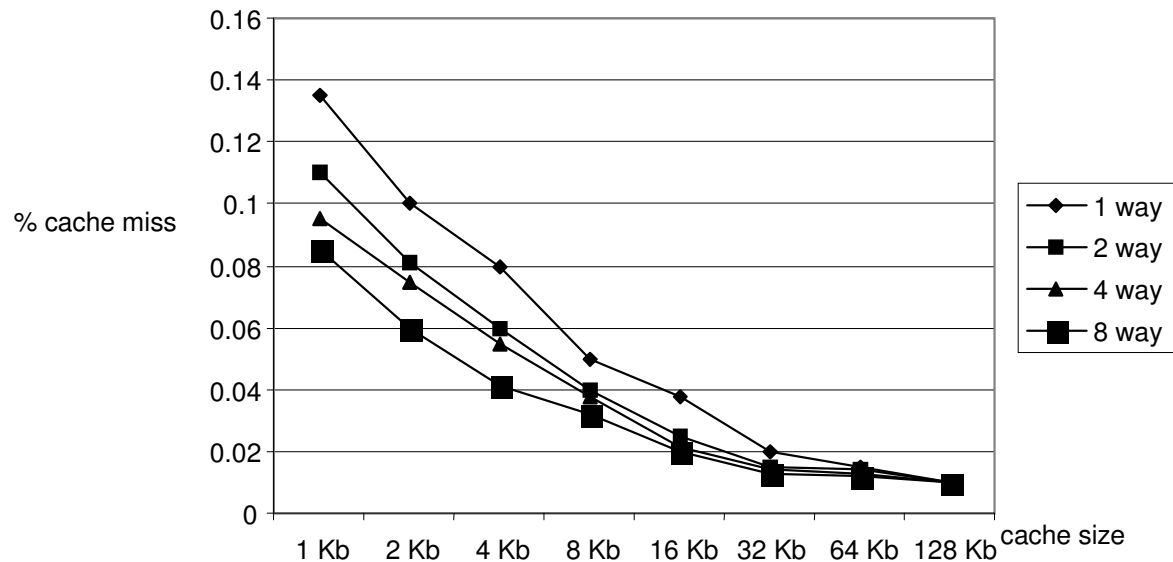
# Cache impact on system performance

---

- Most important parameters in terms of performance:
  - Total size of cache
    - total number of data bytes cache can hold
    - tag, valid and other house keeping bits not included in total
  - Degree of associativity
  - Data block size

# Cache performance trade-offs

- Improving cache hit rate without increasing size
  - Increase line size
  - Change set-associativity



# To Do Before TUESDAY...

---

- Homework 4 (should be on web tonight)