CS/COE1541: Introduction to Computer Architecture

Dept. of Computer Science
University of Pittsburgh

http://www.cs.pitt.edu/~melhem/courses/1541p/index.html

Chapter 5: Exploiting the Memory Hierarchy
Lecture 1

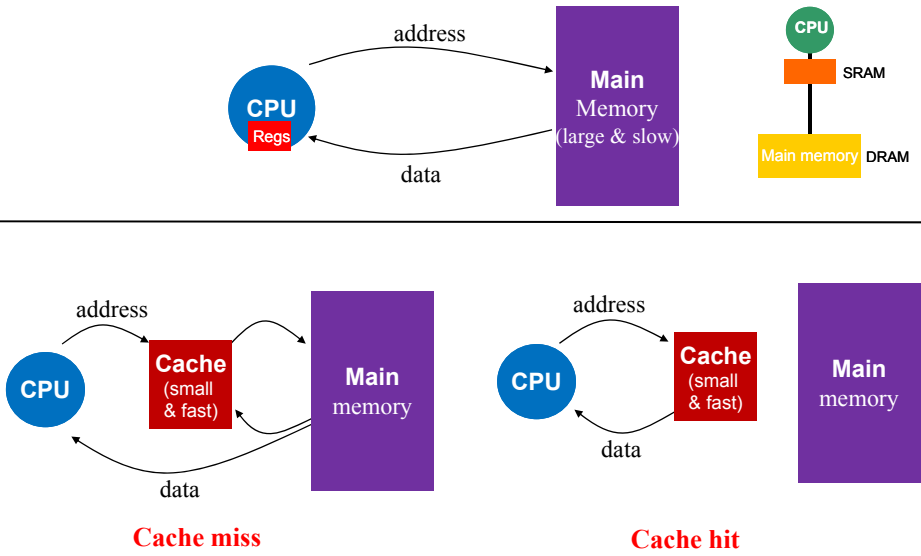
Lecturer: Rami Melhem

Memory technology and Hierarchy

Outline:

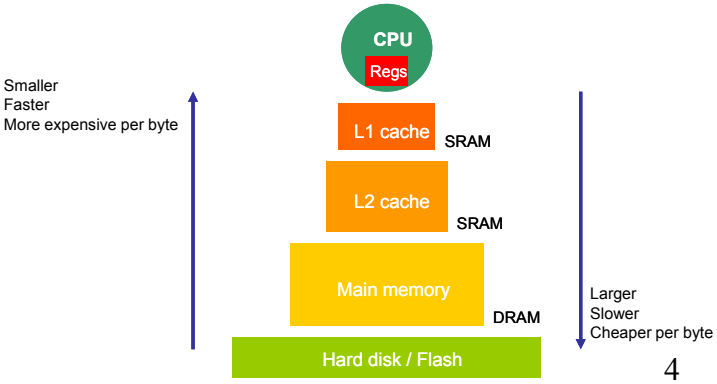
- What is caching?
- Why have a memory hierarchy
- DRAM Vs SRAM
- Disk and Flash storage
- Using SRAM to cache DRAM data

Using a cache to speed up memory operations

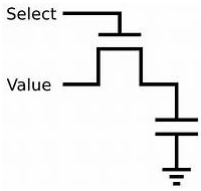


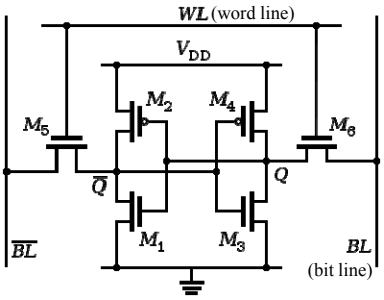
Why have a Memory Hierarchy

- **Because users want fast access to large amount of data (not possible)**
  - SRAM access times are 0.5 – 2.5 ns at cost of $500 to $1000 per Gbyte (in 2012)
  - DRAM access times are 50-70 ns at cost of $10 to $20 per Gbyte.
  - Flash access times are 5-50 us at cost of $0.75 to $1 per Gbyte
  - Disk access times are 5-20 ms at cost of $0.05 to $0.1 per Gbyte.
- **The hierarchy gives the illusion of one large and fast memory**



DRAM Vs SRAM





- DRAM cell relies on a capacitor
  - Charged → 1 , otherwise → 0
- Slower but smaller
- Volatile (may loose charge over time)
- Need to be refreshed periodically

- SRAM cell is built from 6 transistors
- Faster but larger
- Non-volatile – keeps its value
- Does not need to be refreshed

Arranging cells in a 2-D structure

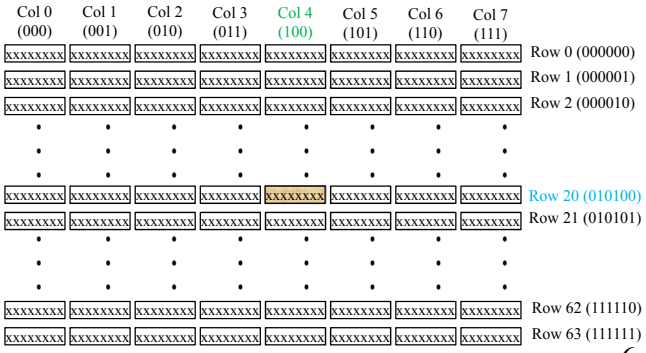
Example:

- Assume a byte-addressable memory that contains 512Bytes (a byte = 8 bits)
- Bytes are arranged into an array of 64x8 bytes (to keep an aspect ratio of 1)
- Assume row major ordering → byte *x* is in row *x*/8 and column (*x* mod 8)
- Need 9 bits to address one of the 512 bytes
- A 9-bit address is translated into a row address (6 bits) and a column address (3 bits)

Byte address 164
→ Row address 20
+ col address 4

Note:
20 = 164 / 8
4 = 164 mod 8

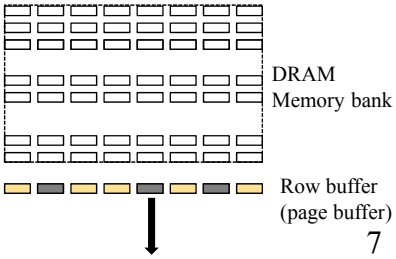
Byte address 010100100
→ Row address 010100
+ col address 100



DRAM operation

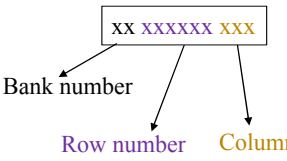
- Each memory bank has a "row buffer", which is non-volatile (SRAM registers)
- To read a byte (a similar process applies for writing):
  - The memory controller sends the row address of the byte
  - The entire row is read into the row buffer (the row is opened)
  - The memory controller sends the column address of the byte
  - The memory returns the byte to the controller (from the row buffer)
  - The memory controller sends a Pre-charge signal (close the open row)

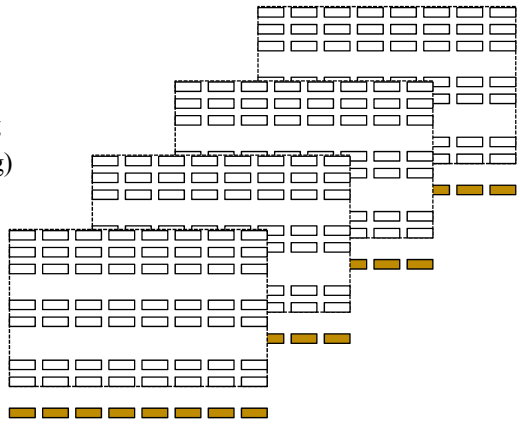
- **Closed row/page policy**: close the row after you read the data
- **Open row/page policy**: close the row only when you want to open a new row – useful if you will read again from the same row.



Multiple banks per memory chip

- Assume 4 banks per chip, 512 bytes per bank → total 2048 bytes
- 2048 bytes → the address of each byte is 11 bits
  - 2 bits to select a bank
  - 6 bits to select a row
  - 3 bits to select a column
- Example of a possible ordering (bank, then row major ordering)

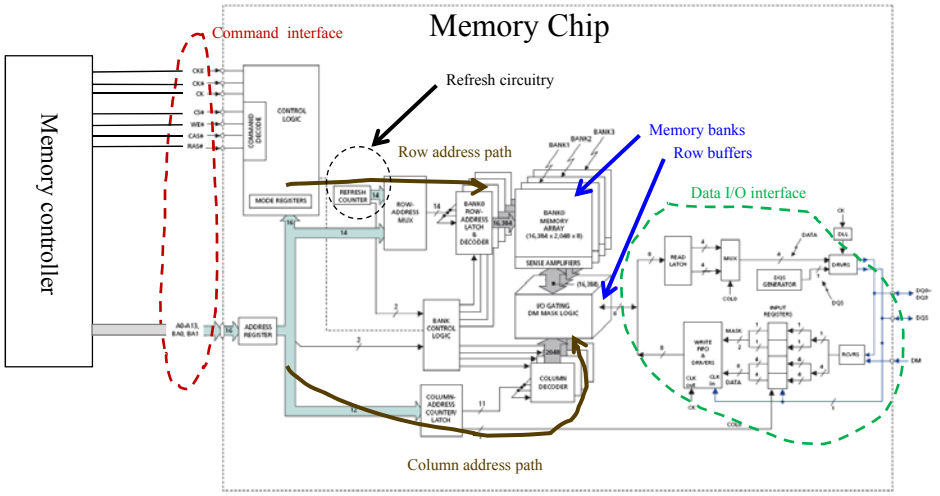




- Other ways of ordering

xx xxx xxxxxx	Bank then column major
xxxxxx xx xxx	Row then bank major

DRAM internals (not in textbook)

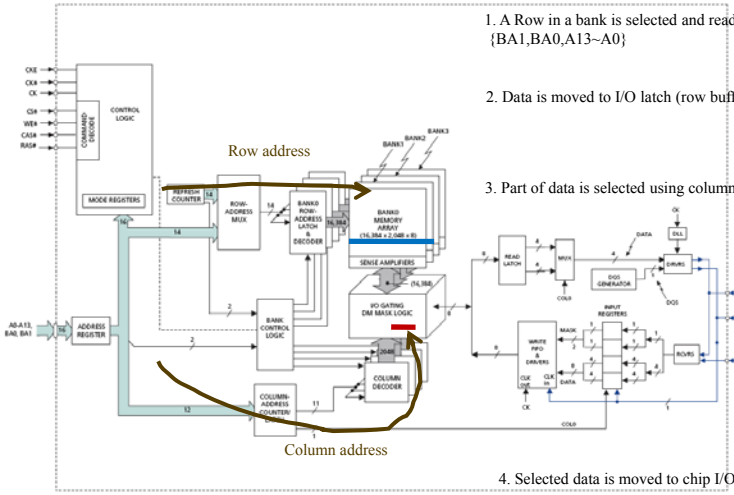


EX: a 128 Mbytes chip (2^{27}) → 4 banks, each having 32 Mbytes organized as 2^{14} =16K rows and 2^{11} =2K columns

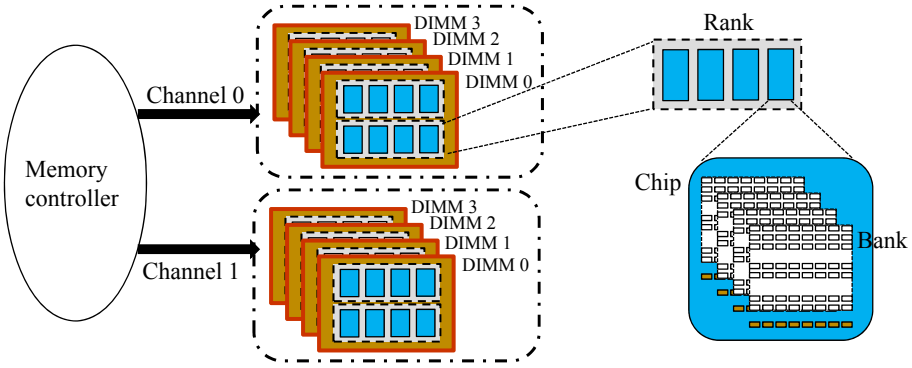
Bank address

2 bits	14 row address bits	11 column address bits

DRAM internal

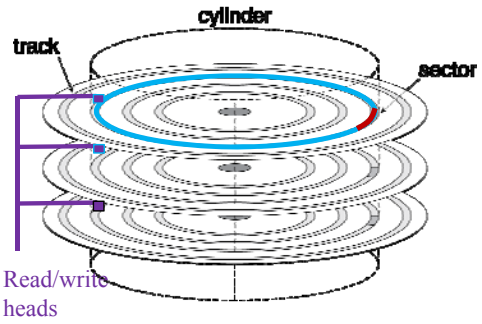


DRAM Organization



- A memory chip is organized internally as a number of banks (1-8 usually).
- Multiple banks can execute different commands at the same time
- A Rank consists of multiple (parallel) chips contributing to the same transaction. For example, each of 4 chips can provide a byte for a total of 32 data bits (read or written).
- A DIMM (Dual Inline Memory Module) consists of 1 – 4 ranks (2 in the figure) mounted on a single printed-circuit board.
- A Channel supports multiple DIMMS (4 in the above figure)

Hard disk storage



Example:

- One or more magnetic surfaces
- 10,000 – 50,000 track per surface
- 100 - 500 sectors per track,
- 512 Bytes (4Kbits) per sector,
- A bit is stored in a small area of the magnetic surface
- One magnetization direction → 1
- The opposite direction → 0
- One read/write head per surface
  - Can be positioned over any track
- Disks are continuously rotating

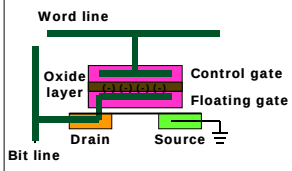
To access data:

– seek: position head over the proper track (each surface has a head)
– rotational latency: wait for desired sector (on average half a rotation)
– transfer: read or write the data from a sector

Flash Storage

- Non-volatile semiconductor storage
  - 100X – 1000X faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)
- Flash bits wears out after 1000's of accesses
  - Not suitable for main memory (too many write operations)
  - Wear levelling: remap data to less used blocks

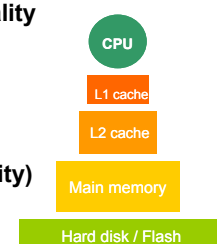
NOR flash:
Lower density
Random access
More reliable
Slower erase
Faster random read
Used for instruction memory



NAND flash:
Higher density
Page access
Less reliable (needs ECC)
Faster erase
Faster streaming read
Used for streamed data

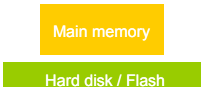
Locality

- **Having a memory hierarchy makes sense because of "locality of reference".**
- **If an item is referenced,**
  - it will tend to be referenced again soon (**temporal locality)**
  - nearby items will tend to be referenced soon (**spatial locality)**



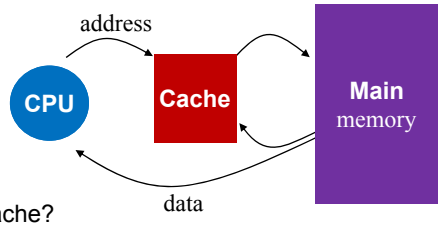
Can you think of examples of locality in code and data?

- **Consider any two consecutive levels (upper, lower)**
  - **block:** minimum unit of data transferred between levels
  - **hit:** data requested is in the upper level
  - **miss:** data requested is not in the upper level
  - **hit rate:** percentage of hits (sometimes called hit ratio)
  - **miss rate:** percentage of miss (sometimes called miss ratio)
  - **hit time:** the time to satisfy request in case of a hit
  - **miss penalty:** the time to satisfy a request in case of a miss



The Basics of Caches

- We will consider the caching of data from main memory into an SRAM cache
- Until specified otherwise, we will assume that the block size is one word



- **Will discuss**
  - where do we put an item in the cache?
  - how do we identify the items in the cache?

- **Two solutions**
  - put item anywhere in cache (associative cache)
  - associate specific locations to specific items (direct mapped cache)