Today's class

- Memory Hierarchy
  - Sections C.1 – C.3
  - Cache performance overview, design criteria for memory (cache) hierarchy

- Learning outcomes
  - Students will get acquainted with the block placement, identification, replacement and write strategy problems
  - Understand the performance equations
The principle of locality

- Programs access a small proportion of their address space at any time

- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables

- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - e.g., sequential instruction access, array data
Exploiting the locality

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from the disk to smaller DRAM memory
  - i.e. the main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory is attached to CPU
CPU&Cache performance

- **Execution time**
  - \( \text{Execution\_time} = ( \text{CPU\_cycles} + \text{memory\_stall\_cycles} ) \times \text{cycle\_period} \)

- **Memory stall cycles**
  - \( \text{Memory\_stall\_cycles} = \text{number\_of\_misses} \times \text{miss\_penalty} \)
    - \( = \text{IC} \times (\text{misses} / \text{instruction}) \times \text{miss\_penalty} \)
    - \( = \text{IC} \times (\text{memory\_accesses} / \text{instruction}) \times \text{miss\_rate} \times \text{miss\_penalty} \)

- **Average memory access time**
  - \( \text{AMAT} = \text{cache\_access\_time} \div \text{memory\_access\_time} \times \text{cache\_miss\_ratio} \)
AMAT and processor performance

- CPU time

  - $\text{CPU\_time} = \text{IC} \times ( \text{CPI\_execution} + \frac{\text{memory\_stall\_cycles/\text{instruction}}}{\text{instruction}} ) \times \text{cycle\_period}$
  
  - $= \text{IC} \times ( \text{CPI\_execution} + \text{miss\_rate} \times \frac{\text{memory\_accesses}}{\text{instruction}} ) \times \text{cycle\_period}$
Cache Performance examples
Memory hierarchy & cache design
The four memory hierarchy questions

The data in memory is organized in logical blocks
- (pages for main/virtual memory)

1. Block placement:
   Where can a block of memory be placed in the upper level?

2. Block identification/search:
   How is a block found in the upper level?

3. Block replacement:
   What block should be removed when the upper level fills up?

4. Write strategy:
   What happens on a data write?
Where can a block be placed?

- **Direct mapping**
  - \( \text{block\_location} = \text{Block\_address} \mod \text{cache\_capacity\_in\_blocks} \)

- **Fully associative**
  - \( \text{block\_location} \) can be anywhere in the cache

- **Set-associative**
  - \( \text{block\_location} = \text{Block\_address} \mod \text{cache\_capacity\_in\_blocks} \)
Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)
    - #Blocks is a power of 2
    - Use low-order address bits
What block should be replaced?

- Random
  - Select a block at random
- Least-recently used (LRU)
  - Reverse use of locality:
    - If the near future depends on the present, then the present depended more on the near past than the distant past. Therefore, older, unused pages are less likely to be used.
- First-in, first-out (FIFO)
  - LRU is hard to implement; FIFO removes the oldest page instead of the least-recently used.
What happens on a write?

- **Write through**
  - Write the information to the main memory as soon as the cache copy is changed
- **Write back**
  - Write the information to the main memory only when the block is to be replaced
- **Dirty-bit technique**
  - Reduces the copies to the main memory
- **On write miss?**
  - *Write allocate*: bring the block to cache
  - *No write allocate*: write directly to main memory
Cache Performance