CSED311 Lab6: Cache

Jae-Jun Ha
dreamline91@postech.ac.kr

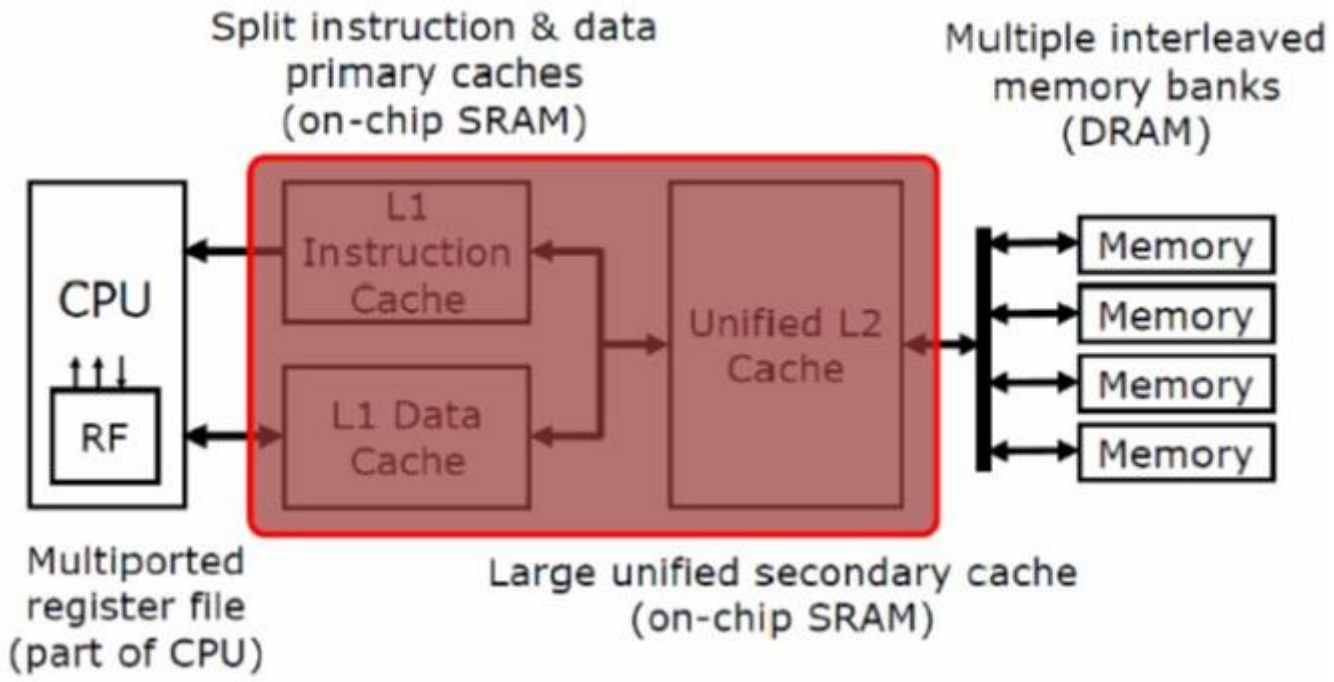
POSTECH

Objectives

- Understand cache
- Implement a direct-mapped cache on your pipeline CPU
- Evaluate the speedup achieved by using cache
  - Hit Ratio
  - Corresponding speedup (vs. no-cache CPU)

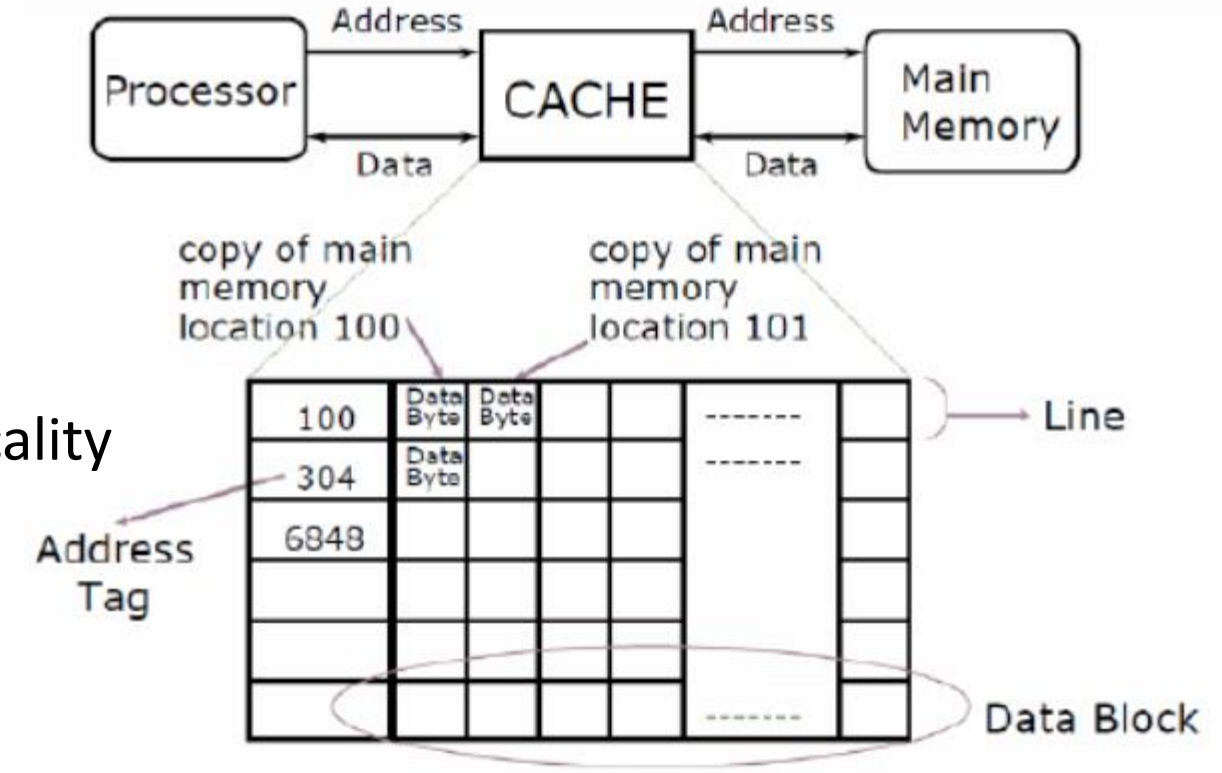
Cache

- Mitigating the gap b/w CPU and memory
  - Memory access: few hundred cycles
  - Cache access: few cycles
- Why does it work?
  - Locality!



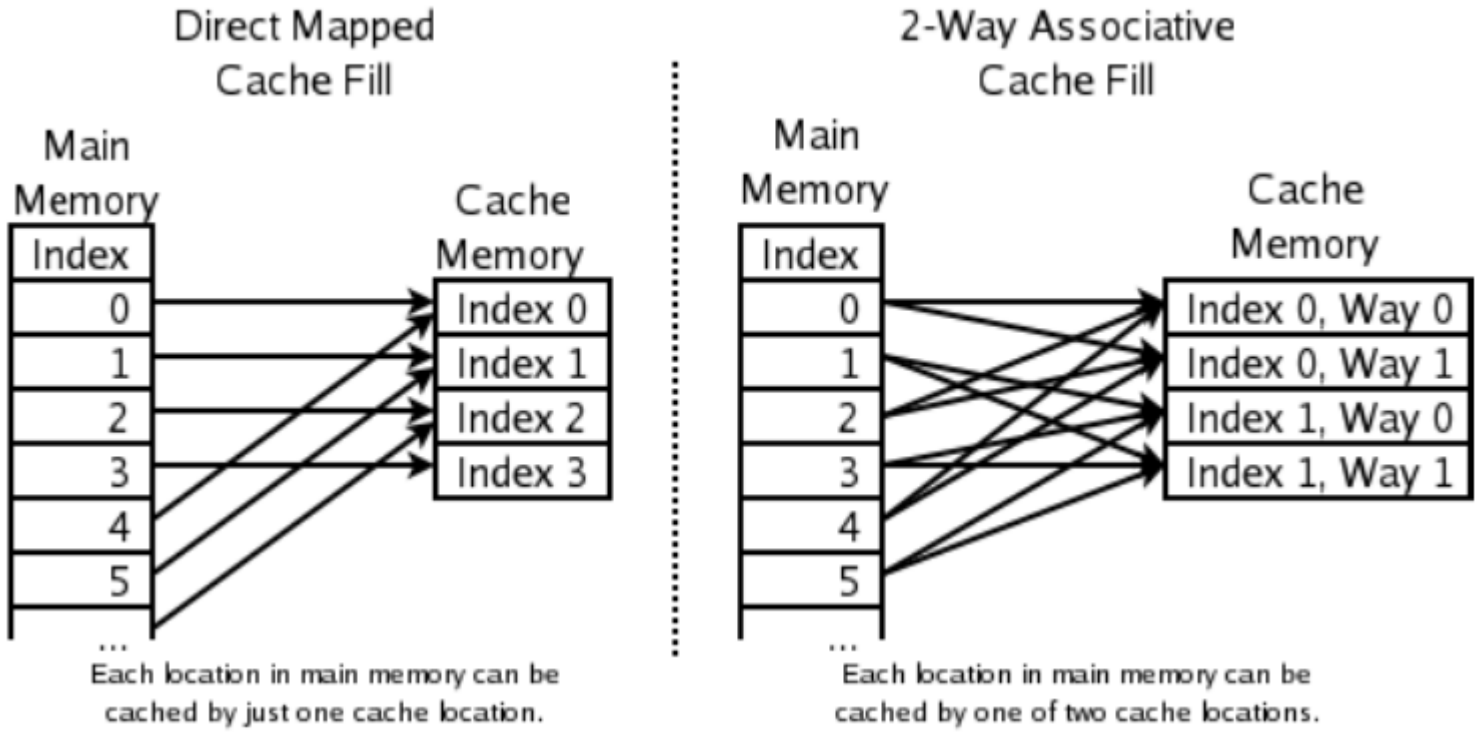
Cache: internal structure

- Tag
  - Detect address conflicts
- Data
  - Fetch by line: exploiting spatial locality



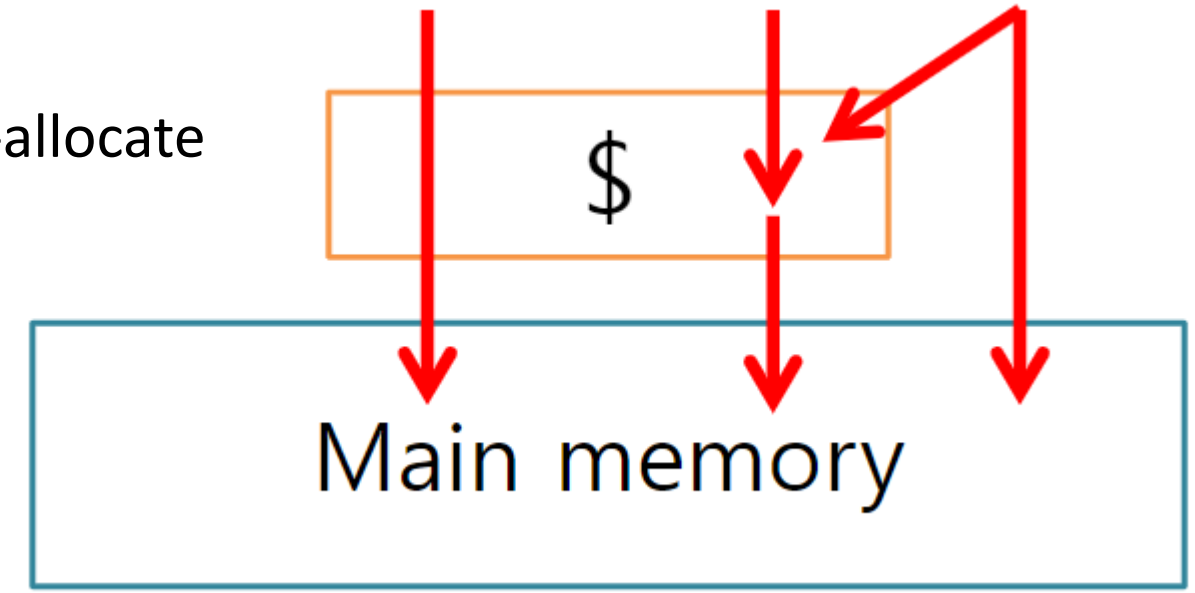
Cache: associativity

- Associativity
  - Reducing address conflicts
- Direct mapped, n-way, fully associative
  - Tradeoffs exist



Cache: other design choices

- Replacement policy
  - Random, LRU, FIFO, …
  - Each has strengths & drawbacks
- Write policy
  - Write-through, writeback, write-no-allocate
  - Related to coherency management



Assignment

- Implement a direct-mapped cache, single level cache
  - Capacity: 32 words / Line size: 4 words
  - If hit, return data in the following cycle
  - Must implement cache on your pipeline CPU

- Things to consider
  - Replacement policy, write policy
  - Unified or separate I/D cache

Assignment (cnt'd)

- Model latencies
  - Previous lab
    - One memory access fetches one word
    - One memory access takes one cycle
  - This lab
    - Need to modify previous files
    - Cache hit takes one cycle
    - One memory access should fetch four words into cache and take six cycles
    - For your baseline CPU (no cache CPU), one memory access should fetch one word and take two cycles
    - This means you need two different memory models
      - Memory for new CPU: return four words in six cycles
      - Memory for no cache CPU: return one word in two cycles

Assignment (cnt'd)

- Memory requirements
  - You can use either a 2-port RAM or a single port RAM
  - Latencies of RAM
    - Should be serialized

⇒ Different ports can handle independent requests!

Assignment

- In report, below contents will be included
  - Calculate the hit (or miss) ratio
    - Hit ratio = (# of hits) / (# of memory accesses)
  - Compare the performance
    - No cache CPU vs your CPU
    - You should use the new latencies

Lab5 Demo