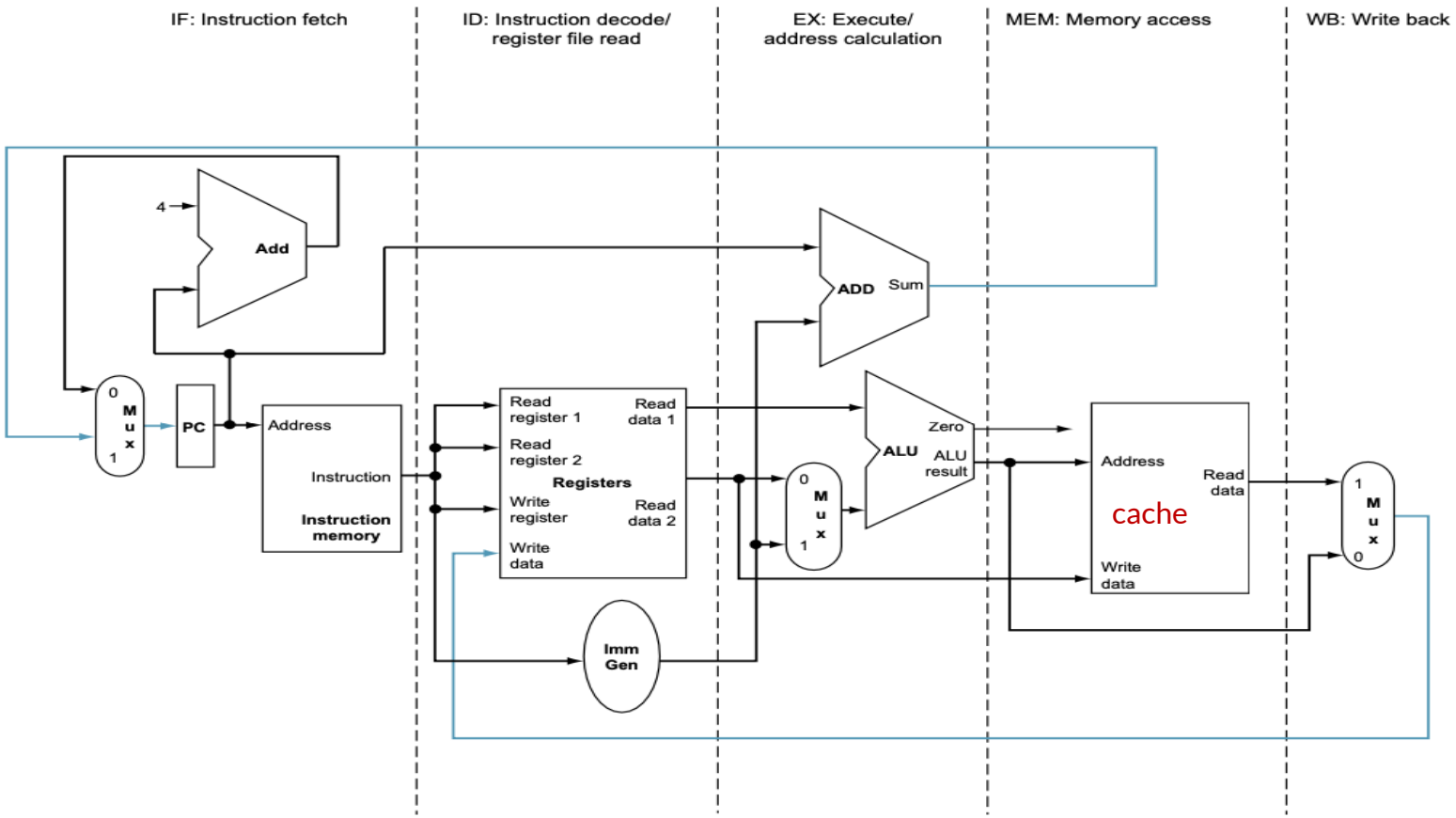
Lab 5a. Data cache

CSED 311

Sungjun Cho

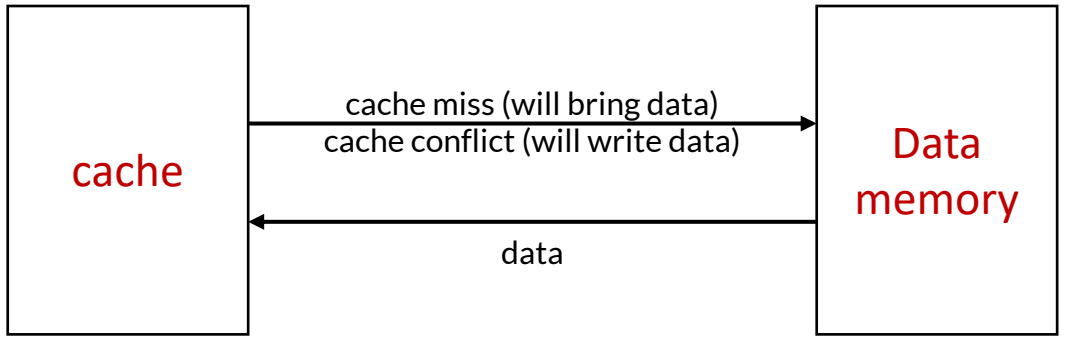
Data cache in Pipelined CPU

- Uses a blocking data cache instead of a “magic memory”

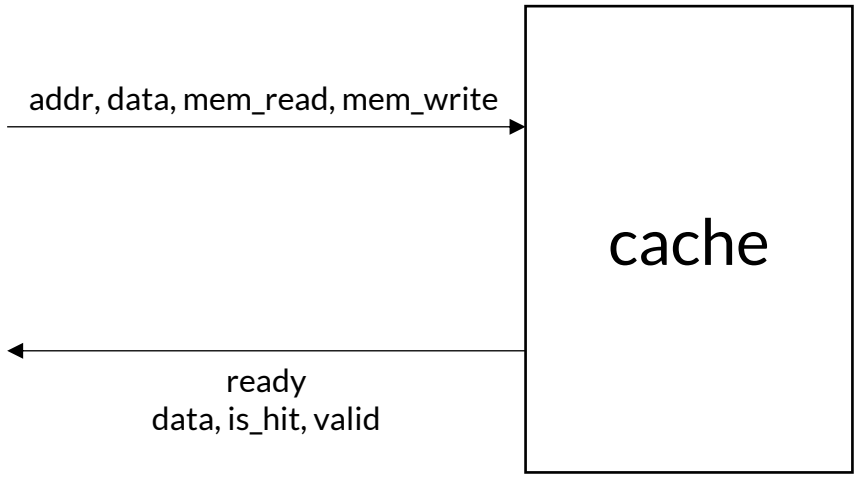


Data flow

- Cache fetches data from the memory

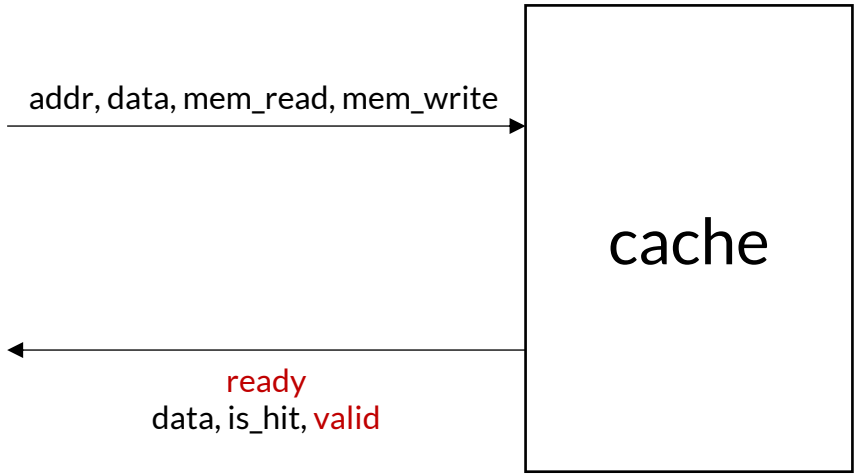


Signals to / from the cache



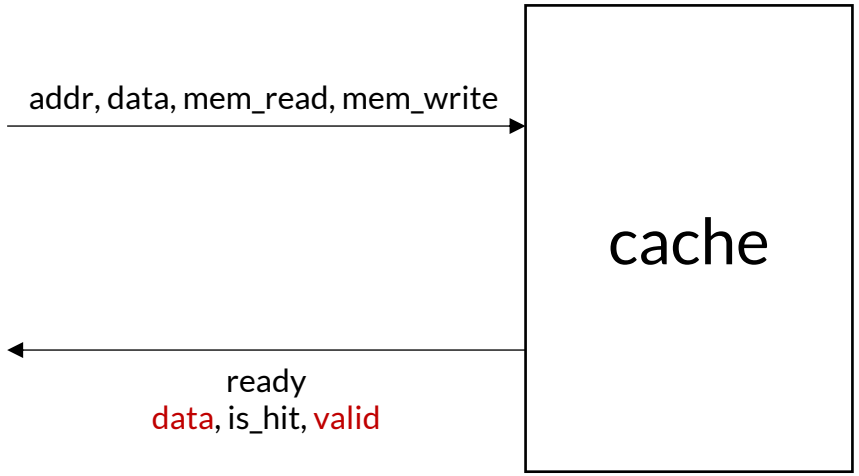
Signals from the data cache

- `ready` represents the status of the cache
  - If `ready` is true => cache is ready to accept request
  - If `ready` is false => cache is busy bring data from the memory
    - `valid` is false
    - Cache cannot accept a request currently (try later)
    - Need to stall the pipeline



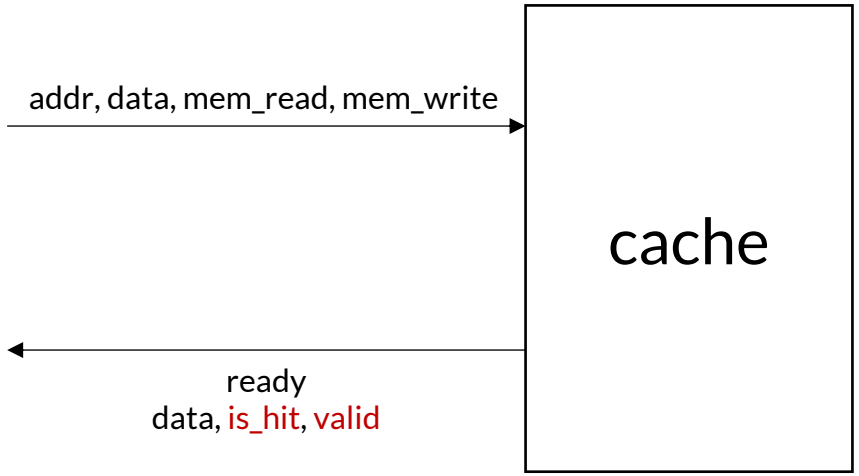
Signals from the data cache

- `data` represents data
  - Ignore `data` when `valid` is false



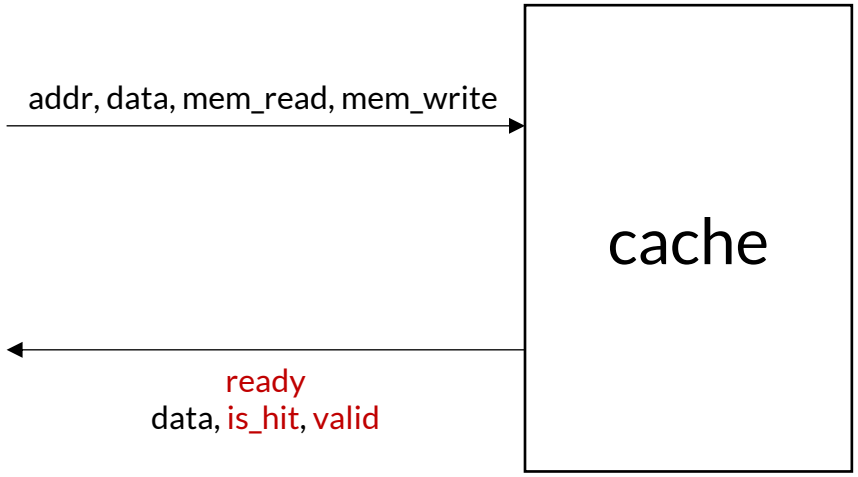
Signals from the data cache

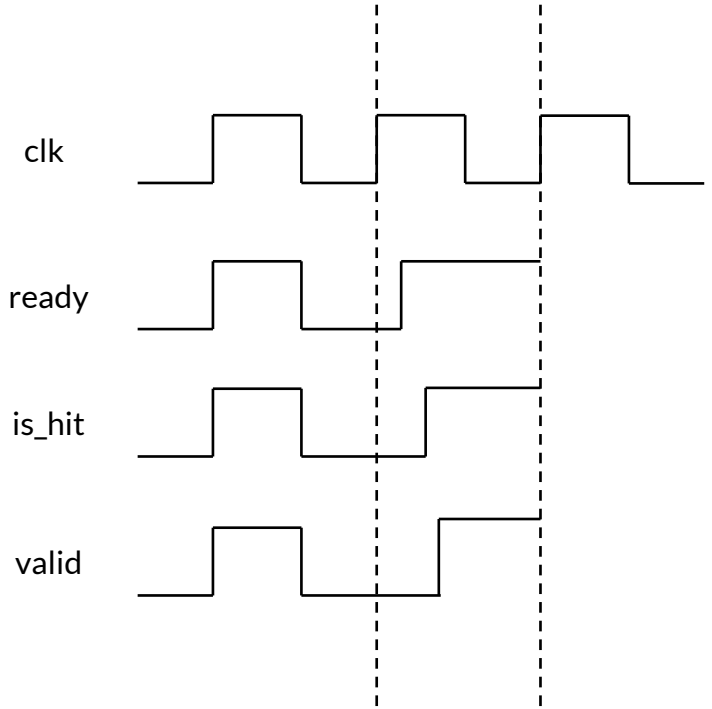
- `is_hit` represents whether a cache hit occurs
  - If a cache miss occurs, stall the pipeline
    - Even if `ready` is true



Signals from the data cache

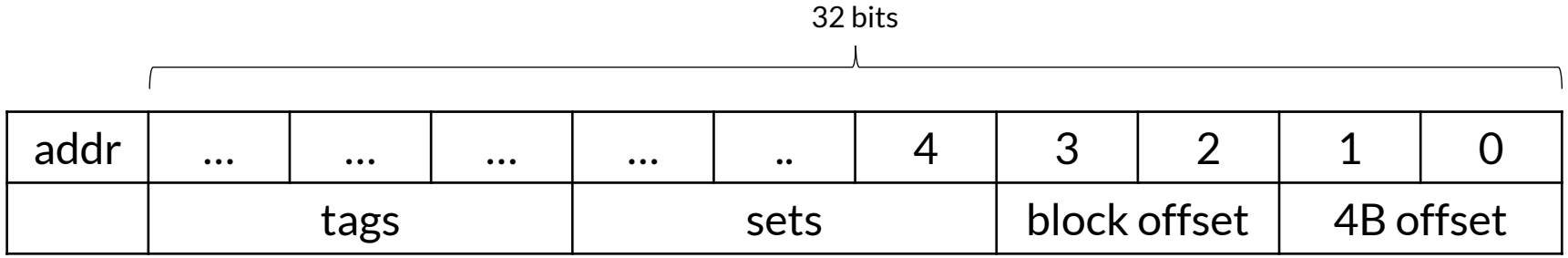
- Pipeline is not stalled only when:
  - `ready`, `is_hit` and `valid` are true





Data cache design

- The size of a cache line (block) is 16 bytes



# sets	# ways	Block offset 0	Block offset 1	Block offset 2	Block offset 4
Set 0	Way 0	4B	4B	4B	4B
	Way 1				
	…				

Data cache design

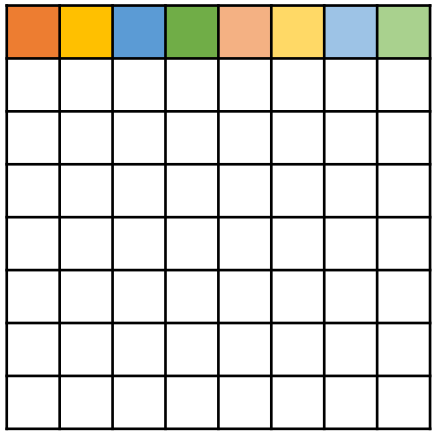
- Asynchronous read:
  - `valid`, `data`, `is_hit`
- Synchronous write:
  - Write data (both from CPU and memory) into the cache line
- Write-allocate
  - Read data from the memory if a write miss occurs
- Write-back

Data cache design

- Replacement policy
  - FIFO
- Structure
  - Direct-mapped or set-associative (not fully-associative)
  - Cache size is 256B for data
    - You are free to define # of ways and sets
- Each cache line should have:
  - Valid bit
  - Dirty bit
  - Bits for replacement
  - ...

Matrix data layout

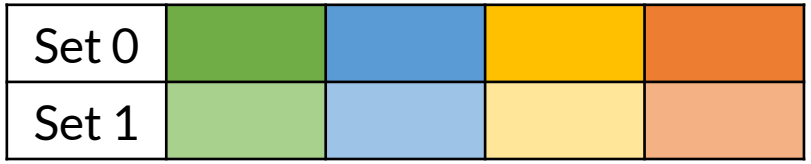
- Memory layout of the matrix (row-major order)
  - Assume each element of the matrix is 4B
  - Assume cache line size is 16B



matrix

address	data
0x00	
0x04	
0x08	
0x0c	
0x10	
0x14	
0x18	
0x1c	

memory

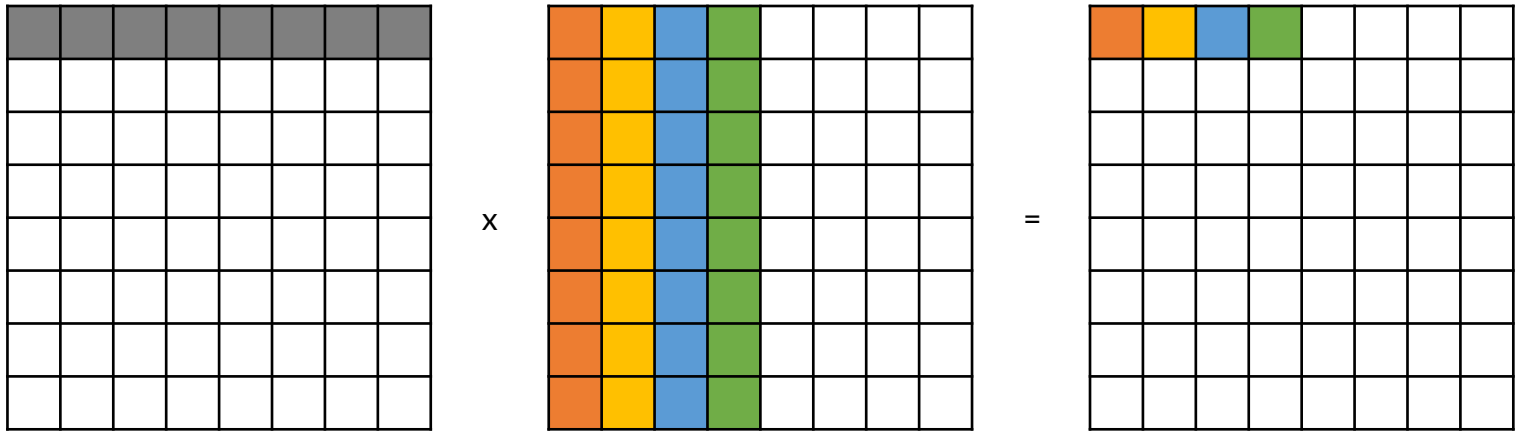


cache

Matrix multiplication

- Naïve implementation
  - Is this cache-friendly? No. Why?

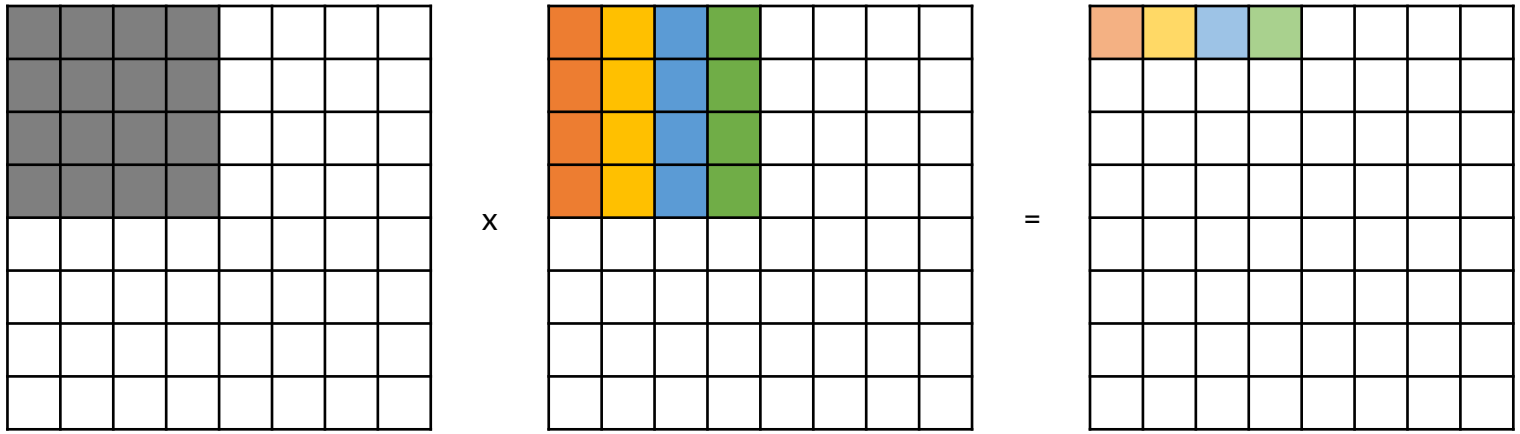
```
// matrix op
for (int m = 0; m < M; ++m) {
  for (int n = 0; n < N; ++n) {
    for (int k = 0; k < K; ++k) {
      c[m][n] += a[m][k] + b[k][n];
    }
  }
}
```



Matrix multiplication

- Tiled implementation
  - Is this cache-friendly? If yes, why?

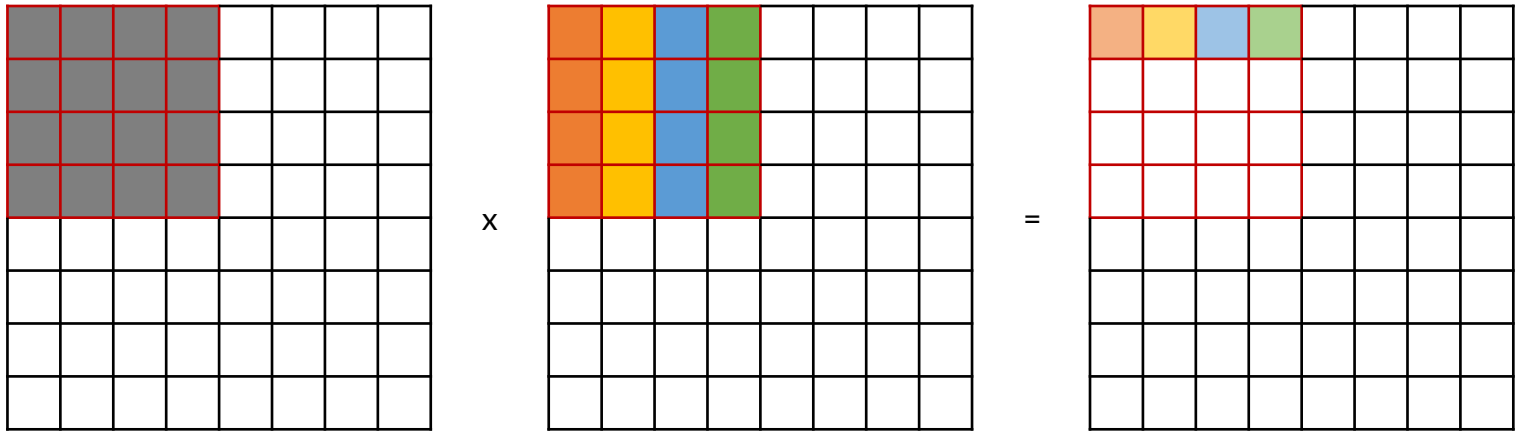
```
// matrix op
for (int tile_m = 0; tile_m < M; tile_m += TILE) {
  for (int tile_n = 0; tile_n < N; tile_n += TILE) {
    for (int tile_k = 0; tile_k < K; tile_k += TILE) {
      for (int m = tile_m; m < tile_m + TILE; ++m) {
        for (int n = tile_n; n < tile_n + TILE; ++n) {
          for (int k = tile_k; k < tile_k + TILE; ++k) {
            c[m][n] += a[m][k] + b[k][n];
          }
        }
      }
    }
  }
}
```



Matrix multiplication

- Tiled implementation
  - Is this cache-friendly? If yes, why?
  - Reuse data (in the cache) as much as possible within each tile
    - The tile size is set by considering cache line size

```
// matrix op
for (int tile_m = 0; tile_m < M; tile_m += TILE) {
  for (int tile_n = 0; tile_n < N; tile_n += TILE) {
    for (int tile_k = 0; tile_k < K; tile_k += TILE) {
      for (int m = tile_m; m < tile_m + TILE; ++m) {
        for (int n = tile_n; n < tile_n + TILE; ++n) {
          for (int k = tile_k; k < tile_k + TILE; ++k) {
            c[m][n] += a[m][k] + b[k][n];
          }
        }
      }
    }
  }
}
```



Submission

- Implementation (Deadline: 5/31 9:00 am)
  - Blocking data cache
    - Direct-mapped (no extra credit)
    - N-way associative cache (Full extra credit + 3)
  - You need to follow the rules described in lab_guide.pdf

- Report (Deadline: 5/31 23:59)
  - The design of the cache
    - Direct-mapped or associative cache
  - Analyze cache hit ratio
    - If you implement associative cache, compare it with direct-mapped cache
    - naive_matmul vs opt_matmul
    - why is the cache hit ratio different between two matmul algorithms?
    - what happens to the cache hit ratio if you change the # of sets and # of ways?

Submission

- Implementation file format
  - .zip file name: Lab4-3a_{team_num}_{student1_id}_{student2_id}.zip
  - Contents of the zip file (only *.v):
    - All files except top.v, Memory.v, and RegisterFile.v
    - Note that top.v, Memory.v, and RegisterFile.v must not be modified
- Report file format
  - Lab4-3a_{team_num}_{student1_id}_{student2_id}.pdf