Introduction to Computer Architecture
Project 3

MIPS CPU Simulator + Cache Simulator

Hyungmin Cho
Department of Software
Sungkyunkwan University

Project 2 Overview

- In Project 3, you'll expand your project 2 implementation to model the **cache** behavior.
  - At the end of the execution, the simulator reports the number of total cache hits and misses.
  - We do not consider the instruction cache. Only the **data cache** is simulated
    - You need to model the cache behavior when you're processing the **`lw`** and **`sw`** instructions.
  - Need to support the same set of MIPS instructions as Proj2

Cache Specs to Simulate – Cache #1

- 1KB (1024 bytes) of total data capacity
- Direct-mapped
- “Write-through, no write allocate” policy
- 32-bit memory address
- Block size: **variable** (2nd input argument)

Cache Specs to Simulate – Cache #2

- 4KB (4096 bytes) of total data capacity
- 2-way set associative, LRU replacement policy
- “Write-back, write allocate” policy
- 32-bit memory address
- Block size: **variable** (2nd input argument)

Simulator Program Behavior

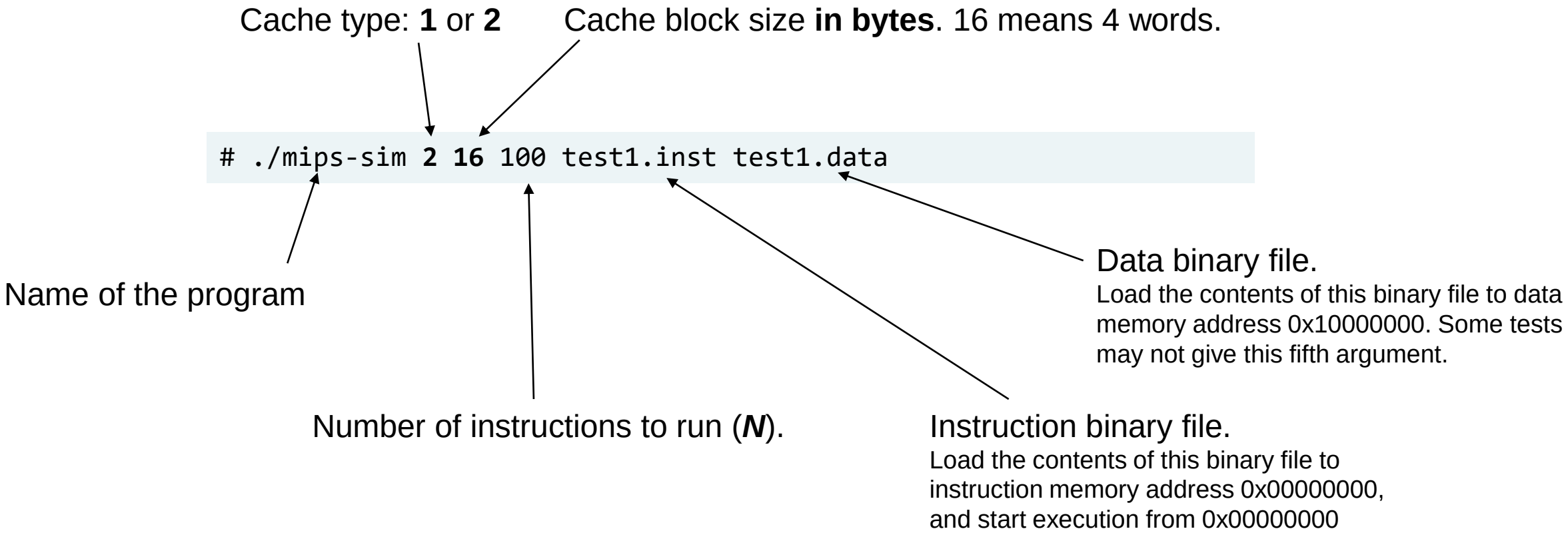
1. Your program takes 4 or 5 command-line arguments.

2. 1st argument: Cache type
   - If the 1st argument is 1, simulate Cache #1
   - If the 1st argument is 2, simulate Cache #2

3. 2nd argument: Cache block size in bytes

4. The 3rd, 4th and 5th arguments are the same as the 1st, 2nd, and 3rd arguments of Proj2
   - The 3rd argument is the number of instructions to execute (***N***)
   - The 4th argument is the instruction binary file.
   - The 5th optional argument is the data binary file.

Simulator Program Behavior



Accepted Block Sizes

- The block size must meet the following rule:
  - Block size ≥ Word size (4)
  - Block size ≤ (Cache capacity / Set associativity)
    - Maximum block size for Cache #1 is 1024
    - Maximum block size for Cache #2 is 2048
  - Block size must be 2^n

- You don’t need an error handling for invalid cache sizes inputs.
  - We will test the valid block sizes only

Outputs

1. Instructions:
   - The number of instructions executed by the simulator.
   - It can be smaller than the second argument (N) if the simulator is terminated due to an unknown instruction. In that case, print the number of executed instructions *including* the unknown instruction.
2. Total:
   - Hits + Misses
3. Hits:
   - The number of data cache hits
4. Misses:
   - The number of data cache misses

```
# ./mips-sim 1 1000 test.inst test.data
Instructions: 222
Total: 14
Hits: 10
Misses: 4
```

- These numbers DO NOT include the memory accesses to load the data file. That is, start counting hits and misses after loading the data file. Also, assume all cache lines are invalid after loading the data file.

- Do not print “unknown instruction” even if the simulator has stopped due to the unknown instruction.
- Do not print the register values
- Do not print the PC value

Reference Implementation

- We provide a reference implementation (without source code) in the following location.
  - `~swe3005/2022s/proj3/mips-sim`

Test Sample (1)

- `~swe3005/2022s/proj3/proj3_1.inst`
- proj3_1 tests the cache block size differences

```
lui $2, 0x1000
lw $3, 0($2)
lw $3, 4($2)
lw $3, 8($2)
lw $3, 12($2)
...
...
lw $3, 60($2)
lw $3, 64($2)
lw $3, 68($2)
...
..
lw $3, 108($2)
lw $3, 112($2)
lw $3, 116($2)
lw $3, 120($2)
lw $3, 124($2)
nop
```

Example outputs

```
./mips-sim 1 4 1000 ~swe3005/2022s/proj3/proj3_1.inst
```

```
Instructions: 35
Total: 32
Hits: 0
Misses: 32
```

```
./mips-sim 1 32 1000 ~swe3005/2022s/proj3/proj3_1.inst
```

```
Instructions: 35
Total: 32
Hits: 28
Misses: 4
```

```
./mips-sim 2 64 1000 ~swe3005/2022s/proj3/proj3_1.inst
```

```
Instructions: 35
Total: 32
Hits: 30
Misses: 2
```

Test Sample (2)

- `~swe3005/2022s/proj3/proj3_2.inst`
- proj3_2 tests the cache size difference (1 KB vs. 4 KB)

```
lui $2, 0x1000
add $3, $2, 0x400
lw $4, 0($2)
lw $4, 0($3)
lw $4, 4($2)
lw $4, 4($3)
nop
```

Example outputs

```
./mips-sim 1 64 1000 ~swe3005/2022s/proj3/proj3_2.inst
```

```
Instructions: 8
Total: 4
Hits: 0
Misses: 4
```

```
./mips-sim 2 64 1000 ~swe3005/2022s/proj3/proj3_2.inst
```

```
Instructions: 8
Total: 4
Hits: 2
Misses: 2
```

Test Sample (3)

- ~swe3005/2022s/proj3/proj3_3.inst
- proj3_3 tests the associativity of Cache #2 (2-way)

```
lui $2, 0x1000
add $3, $2, 0x400
add $4, $3, 0x400
add $5, $4, 0x400
add $6, $5, 0x400
lw $10, 0($2)
lw $10, 4($3)
lw $10, 8($4)
lw $10, 12($5)
lw $10, 16($2)
lw $10, 20($3)
lw $10, 24($4)
lw $10, 28($5)
lw $10, 32($2)
lw $10, 36($3)
lw $10, 40($4)
lw $10, 44($5)
lw $10, 48($6)
lw $10, 52($2)
lw $10, 56($3)
lw $10, 60($4)
lw $10, 56($5)
lw $10, 52($6)
nop
```

Example outputs

```
./mips-sim 1 32 1000 ~swe3005/2022s/proj3/proj3_3.inst
```

```
Instructions: 25
Total: 18
Hits: 0
Misses: 18
```

```
./mips-sim 2 32 1000 ~swe3005/2022s/proj3/proj3_3.inst
```

```
Instructions: 25
Total: 18
Hits: 6
Misses: 12
```

```
./mips-sim 2 64 1000 ~swe3005/2022s/proj3/proj3_3.inst
```

```
Instructions: 25
Total: 18
Hits: 10
Misses: 8
```

Test Sample (4)

- `~swe3005/2022s/proj3/proj3_4.inst`
- proj3_4 tests the replacement scheme of Cache #2 (2-way LRU)

```
lui $2, 0x1000
add $3, $2, 0x400
add $4, $3, 0x400
add $5, $4, 0x400
add $6, $5, 0x400
lw $10, 0($2)
lw $10, 0($3)
lw $10, 0($4)
lw $10, 0($5)
lw $10, 0($6)
lw $10, 0($3)
lui $2, 0x1000
add $2, $2, 0x40
add $3, $2, 0x400
add $4, $3, 0x400
add $5, $4, 0x400
add $6, $5, 0x400
lw $10, 0($2)
lw $10, 0($3)
lw $10, 0($4)
lw $10, 0($5)
lw $10, 0($6)
lw $10, 0($2)
lui $2, 0x1000
add $2, $2, 0x80
add $3, $2, 0x400
add $4, $3, 0x400
add $5, $4, 0x400
add $6, $5, 0x400
lw $10, 0($2)
lw $10, 0($3)
lw $10, 0($4)
lw $10, 0($5)
lw $10, 0($2)
lw $10, 0($6)
lw $10, 0($2)
lw $10, 0($3)
nop
```

Example outputs

```
./mips-sim 1 32 1000 ~swe3005/2022s/proj3/proj3_4.inst
```

```
Instructions: 39
Total: 20
Hits: 0
Misses: 20
```

```
./mips-sim 2 32 1000 ~swe3005/2022s/proj3/proj3_4.inst
```

```
Instructions: 39
Total: 20
Hits: 4
Misses: 16
```

Test Sample (5)

- `~swe3005/2022s/proj3/proj3_5.inst`
- proj3_5 tests the write policy Cache #2 (Write allocate)

```
lui $2, 0x1000
addi $3, $zero, 1234
sw $3, 0($2)
lw $4, 4($2)
nop
```

Example outputs

```
./mips-sim 1 32 1000 ~swe3005/2022s/proj3/proj3_5.inst
```

```
Instructions: 6
Total: 2
Hits: 0
Misses: 2
```

```
./mips-sim 2 32 1000 ~swe3005/2022s/proj3/proj3_5.inst
```

```
Instructions: 6
Total: 2
Hits: 1
Misses: 1
```

Test Sample (6)

- ~swe3005/2022s/proj3/proj3_6.inst
- proj3_6 corresponds to the following C code

```
int main ()  {
  int* p = (int*) 0x10000000;
  int val = 0;

  for(int i = 0; i < 32; i++) {
    p[i*4] = p[i*16];
  }

  return 0;
}
```

Example outputs

```
./mips-sim 1 64 1000 ~swe3005/2022s/proj3/proj3_6.inst
```

```
Instructions: 761
Total: 294
Hits: 244
Misses: 50
```

```
./mips-sim 2 32 1000 ~swe3005/2022s/proj3/proj3_6.inst
```

```
Instructions: 761
Total: 294
Hits: 253
Misses: 41
```

Test Sample (7)

- `~swe3005/2022s/proj3/proj3_7.inst, ~swe3005/2022s/proj3/proj3_7_8.data`
- proj3_7 and proj3_8 are sorting programs. Proj3_7 implements the quicksort algorithm
- `proj3_7_8.data` contains 2,048 numbers (8,192 bytes)

Example outputs

```
./mips-sim 2 64 10000000 ~swe3005/2022s/proj3/proj3_7.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1137603
Total: 434271
Hits: 433776
Misses: 495
```

```
./mips-sim 2 128 10000000 ~swe3005/2022s/proj3/proj3_7.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1137603
Total: 434271
Hits: 433815
Misses: 456
```

```
./mips-sim 2 256 10000000 ~swe3005/2022s/proj3/proj3_7.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1137603
Total: 434271
Hits: 433598
Misses: 673
```

Test Sample (8)

- `~swe3005/2022s/proj3/proj3_8.inst, ~swe3005/2022s/proj3/proj3_7_8.data`
- proj3_7 and proj3_8 are sorting programs. Proj3_8 implements the radix sort algorithm
- `proj3_7_8.data` contains 2,048 numbers (8,192 bytes)

Example outputs

```
./mips-sim 2 64 10000000 ~swe3005/2022s/proj3/proj3_8.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1067581
Total: 402343
Hits: 396477
Misses: 5866
```

```
./mips-sim 2 128 10000000 ~swe3005/2022s/proj3/proj3_8.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1067581
Total: 402343
Hits: 396732
Misses: 5611
```

```
./mips-sim 2 256 10000000 ~swe3005/2022s/proj3/proj3_8.inst ~swe3005/2022s/proj3/proj3_7_8.data
```

```
Instructions: 1067581
Total: 402343
Hits: 395711
Misses: 6632
```

Project Environment

- We will use the department's ~~In~~-Ui-Ye-Ji cluster
  - `swui.skku.edu`
  - `swye.skku.edu`
  - `swji.skku.edu`
  - ssh port: 1398

- You'll need a similar Makefile as proj2
  - Same executable file name (i.e., **`mips-sim`**)

- If you have a problem with the account, send an e-mail to the server admin
  - inuiyeji-skku@googlegroups.com
  - Do not send an email that is not related to the account itself!
  - If you're not sure, ask the TAs first.

Submission

- Clear the build directory
  - Do not leave any executable or object file in the submission
  - `make clean`

- Use the `submit` program
  - `~swe3005/bin/submit project_id path_to_submit`
  - If you want to submit the ‘project_3’ directory…
    - **`~swe3005/bin/submit proj3 project_3`**

```
Submitted Files for proj2:
File Name                                    File Size       Time
------------------------------------------------------------------------------
proj3-2020123456-Sep.05.17.22.388048074        268490                  Thu Sep  5 17:22:49 2020
```

- Verify the submission
  - **`~swe3005/bin/check-submission proj3`**

Project 3 Due Date

- 2022 June 6th (Monday), 23:59:59

- No late submission