Computer Organization 2024
Assignment III

Overview

- Caches enhance the processor performance by accommodating recent or frequent-used data items in the fast memories

- To reduce memory access overhead, there are two ways for hardware-software co-design
  1. Application-aware cache design
  2. Cache-architecture aware memory access

Assignment

- Part 1. (Programming)
  1. Implement FIFO data cache replacement policy on Spike
  2. Reduce memory access overhead of the given two programs

- Part 2. (Demo)
  1. Explain how the Spike cache replacement policy works
  2. Explain the design philosophy of your revised programs, e.g., the techniques/concepts you used in the revised program

Part 1. Preliminary

- Cache specifications
  - 4 ways
  - 8 sets
  - 32 bytes cacheline
  - Miss penalty = 30 cycles
  - Hit latency = 1 cycle
- Only RV64GC instruction set are allowed
- To enable data cache in Spike simulator
  - Add –dc=<set>:<ways>:<cacheline>

```
$ spike --isa=RV64GC --dc=8:4:32 $RISCV/riscv64-unknown-elf/bin/pk <your program>
```

Part 1. Implement FIFO in Spike

- Implement FIFO data cache replacement policy to Spike
- Two files to modify in Spike source code
  - *{PATH_TO_SPIKE}*/riscv/cachesim.h
  - *{PATH_TO_SPIKE}*/riscv/cachesim.cc
- Recompilation is needed after implement FIFO replacement policy
  - Please refer to HW0 on NCKU Moodle
- You can add new data structures in your implementation

Part 1. 2D Convolution

- Provided implementation is in:
  - *{PATH_TO_HW3}*/Q1/conv2d.S
- Implement Your better version in:
  - *{PATH_TO_HW3}*/Q1/conv2d-better.S
- Write in pure assembly code

```
.global conv2d
conv2d:
# a0: address of input
# a1: address of kernel
# a2: w
# a3: h
# a4: ksize
# a5: s

# prologue (you can change the prologue if you want)
    # addi sp, sp, -4
    # sw a0, 0(sp)
# start of your implementation

# end of your implementation

# epilogue (you can change th prologue if you want)
    # lw a0, 0(sp)
    # addi sp, sp, 4
    ret
```

https://riscv.org/wp-content/uploads/2015/01/riscv-calling.pdf

Part 1. 2D Convolution

- This is the C code of original version (Q1/conv2d.S)
  - You can start from improve this C code first before delved into assembly code

```
void conv2d(int *input, int *kernel, int w, int h, int ksize, int s){
    for (int j = 0; j <= w - ksize; j++){
        for (int i = 0; i <= h - ksize; i++){
            int sum = 0;
            for (int n = 0; n < ksize; n+=s){
                for (int m = 0; m < ksize; m+=s){
                    sum += input[(i + m) * w + j + n] * kernel[m * ksize + n];
                }
            }
            input[i*w + j] = sum;
        }
    }
}
```

Part 1. Matrix Multiplication

- Provided implementation is in:
  - *{PATH_TO_HW3}*/Q2/mm.c
- Implement your better version in:
  - *{PATH_TO_HW3}*/Q2/mm-better.c
- Write in C code

```
void matrix_multiplication(int *a, int *b, int *output, int i,
                           int k, int j) {
    /* Your implementation */
    return;
}
```

Files you need to change in this assignment

```
riscv-isa-sim/
└── riscv
    ├── cachesim.cc
    └── cachesim.h
```

```
CO2024HW3
├── makefile
├── Q1
│   ├── conv2d-better.S
│   ├── conv2d.c
│   ├── conv2d.S
│   └── testbench_driver.c
├── Q2
│   ├── mm-better.c
│   ├── mm.c
│   └── testbench_driver.c
├── testq1.py
└── testq2.py
```

Test Your Code

- Use `make check` for checking the algorithm correctness and the memory access overhead
- Cache miss penalty is 30 cycles in this assignment, and the cache hit latency is 1 cycle

Memory Access Cycles:

$$\text{MemCycle} = \text{cache}_{hit} \times \text{latency}_{hit} + \text{cache}_{miss} \times \text{penalty}_{miss}$$

Improved ratio:

$$\text{MemCycle}_{Ori} / \text{MemCycle}_{Imp}$$

```
=============== Q1 ===============
Original version
D$ Bytes Read:            9388618
D$ Bytes Written:         2756484
D$ Read Accesses:         2005180
D$ Write Accesses:        478403
D$ Read Misses:           95201
D$ Write Misses:          41069
D$ Writebacks:            108920
D$ Miss Rate:             5.487%

Memory access overhead =  6435413 (cpu cycle)
---------------------------------
Improved version
D$ Bytes Read:            9388618
D$ Bytes Written:         2756484
D$ Read Accesses:         2005180
D$ Write Accesses:        478403
D$ Read Misses:           34629
D$ Write Misses:          41069
D$ Writebacks:            52533
D$ Miss Rate:             3.048%

Memory access overhead =  4678825 (cpu cycle)

Improved ratio:  1.3754335757374982
Output Correctness:  Pass
---------------------------------
```

Test Your Code

- Use `make diffq1` to check the ouput:
  - This will compare your 2D convolution output with correct answer
- Use `make diffq2` to check the output:
  - This will compare your matrix multiplication output with correct answer

(Example) make diffq1

```
diff --git a/Q1/ans.output b/Q1/stu.output
old mode 100755
new mode 100644
index 90986cc..d05a4e9
--- a/Q1/ans.output
+++ b/Q1/stu.output
@@ -15 +15 @@
-842 -395 54 504 954 378 -199-1 248 -329 119 -456 -6 443 891 3
6 123 573 -2 446 -131 316 -261 -837 -387 -961 -512 -65 382 829
2 67 514 -63 -640 -192 -767 -317 -892 -444 3 450 -127 321 -253
442 5 -572 -124 -698 -248 201 648 71 518 -58 391 -183 -758 -31
03 -54 396 845 269 716 139 586 10 -564 -114 -689 -242 -819 -37
65 914 337 784 207 -369 80 -494 -45 -621 -174 273 720 144 594
5 -172 275 -301 149 -425 -1000 -553 -106 341 789 214 664 89 -4
4 344 -231 -805 -356 -932 -485 -38 409 857 283 -291 158 -419 2
162 -736 -287 -864 -417 30 478 -97 353 -222 226 -351 -928 -481
7 -219 228 675 98 546 -28 422 -153 -730 -283 -860 -412 37 487
6 743 167 616 42 -533 -85 -15 376
make: *** [makefile:17: diffq1] Error 1
```

Part 2. Demo

- Demo session on
  - June 20-21, 2024 (all days)
- Demo session at
  - Room 65704 (Advanced Systems Research Lab)
- We will download the code you submit before June 19, 2024 23:59 in demo section. You don't need to prepare PC in demo session
- Please register your preferred time on the Google Sheet on NCKU Moodle

Scoring Criteria

- Part 1. (Programming) 40%
  1. Implement FIFO data cache replacement policy on Spike (10%)
  2. Reduce the memory access overhead of 2D convolution (15%)
     - Improved > 1.3 (15/15)
     - Improved > 1 (10/15)
  3. Reduce the memory access overhead of matrix multiplication (15%)
     - Improved > 1.8 (15/15)
     - Improved > 1 (10/15)

Note: If you cannot implement FIFO replacement policy, you can still work on other questions

Scoring Criteria

- Part 2. (Demo) 80%
  1. Explain how cache mechanism work in Spike simulator (10%)
  2. Explain how you reduce overhead on 2D convolution (25%)
  3. Explain how you reduce overhead on matrix multiplication (25%)
  4. Bonus: Does your method can apply on any matrix size? Why or Why not? (20%)

Submission

- Compress your source files into a **zip** file
- Submit your zip file to NCKU Moodle
- The files organization of your zip file should be as follow
  - NOTE: Change all StudentID to your student ID, e.g. F12345678_HW3.zip

Incorrect format will lose 10 points

```
StudentID_HW3.zip
└── StudentID_HW3
    ├── cachesim.cc
    ├── cachesim.h
    ├── Q1
    │   └── conv2d-better.S
    └── Q2
        └── mm-better.c
```

Submission

- Office hours: Every Friday 10 a.m. to 12 p.m.
  - At room 65704 (Advanced Systems Research Lab)
- If you have any problem, please contact TA via Lab's email
  - asrlab@csie.ncku.edu.tw
  - Email subject starts with "[Comp2024]"
- Deadline: 2024/06/19 23:59
- Late submission are limited to one week after deadline, and the score for late submissions will be reduced to 70%

QA

Thanks.