RiscV simulator

20198111018 성소윤

Parsing Elf file

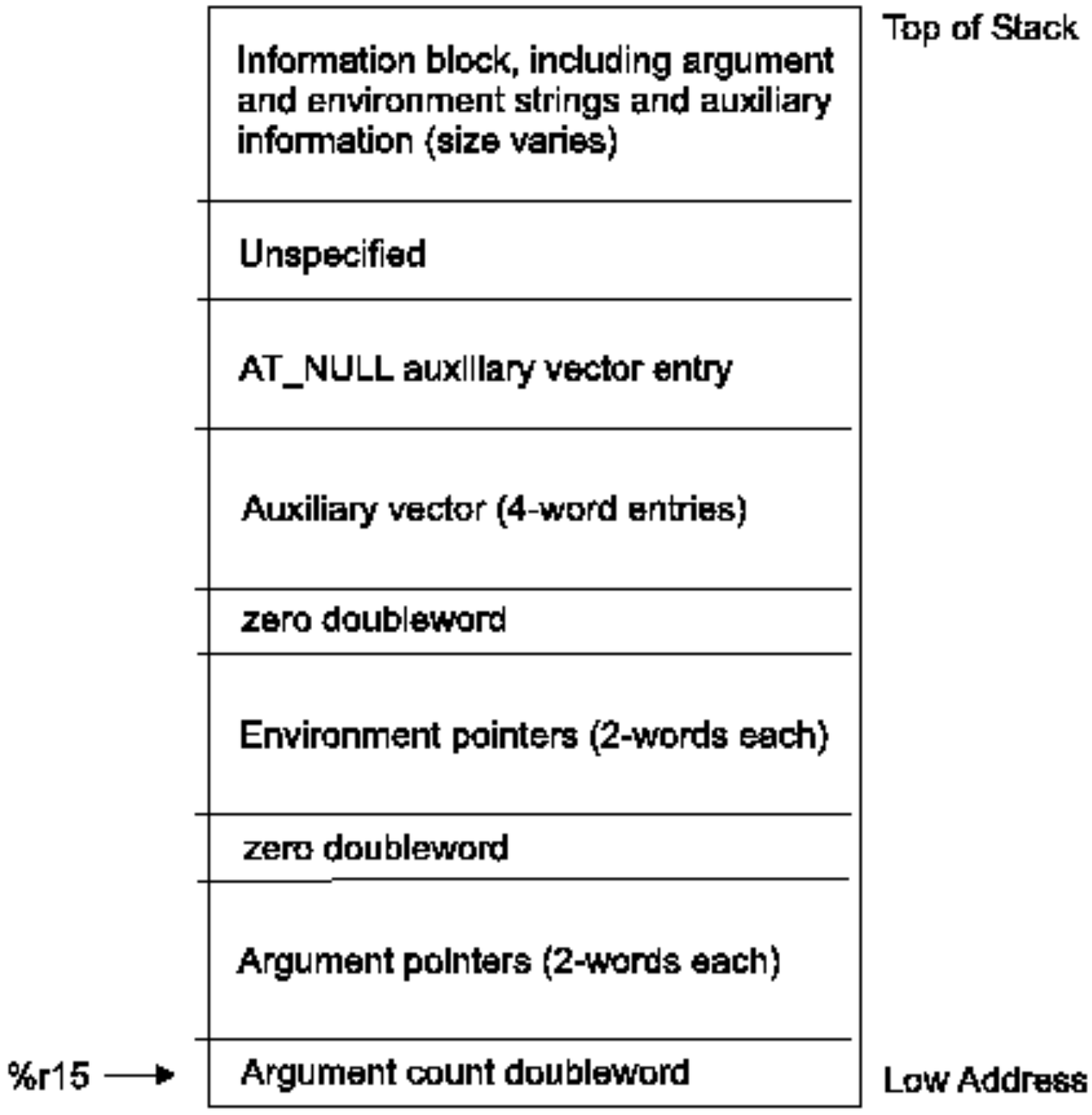
```
typedef struct {
        uint8_t  e_ident[16];
        uint16_t e_type;
        uint16_t e_machine;
        uint32_t e_version;
        uint32_t e_entry;
        uint32_t e_phoff;
        uint32_t e_shoff;
        uint32_t e_flags;
        uint16_t e_ehsize;
        uint16_t e_phentsize;
        uint16_t e_phnum;
        uint16_t e_shentsize;
        uint16_t e_shnum;
        uint16_t e_shstrndx;
} Elf32_Fhdr;
```

-> entry point, phnum

```
typedef struct
{
        uint32_t p_type;
        uint32_t p_offset;
        uint32_t p_vaddr;
        uint32_t p_paddr;
        uint32_t p_filesz;
        uint32_t p_memsz;
        uint32_t p_flags;
        uint32_t p_align;
 } Elf32_Phdr;
```

-> PT_LOAD인 경우 memory에

Stack



Process header

```
{AT_ENTRY, fh.e_entry},
 {AT_PHNUM, fh.e_phnum},
 {AT_PHENT, sizeof(Elf32_Phdr)},
 {AT_PHDR, sp_phdr},
 {AT_PAGESZ, 0},
 {AT_SECURE, 0},
 {AT_RANDOM, stack_top},
 {AT_NULL, 0}
```

Arg cnt: 1
Arg val: Program name

System call (Linux)

```
void handle_syscall(RegisterFile& register_file, Memory& memory, unsigned long long clock)
{
        register_file.gpr[10] = do_syscall(register_file.gpr[10]
                , register_file.gpr[11], register_file.gpr[12], register_file.gpr[13],
                register_file.gpr[14], register_file.gpr[15], register_file.gpr[17], memory, clock);
    std::clog << "sys call num: " << std::dec << register_file.gpr[17] << std::endl;
}
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/uio.h>
#include <sys/mman.h>
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/utsname.h>
```

sys call num: 214 (brk)

sys call num: 160 (uname)

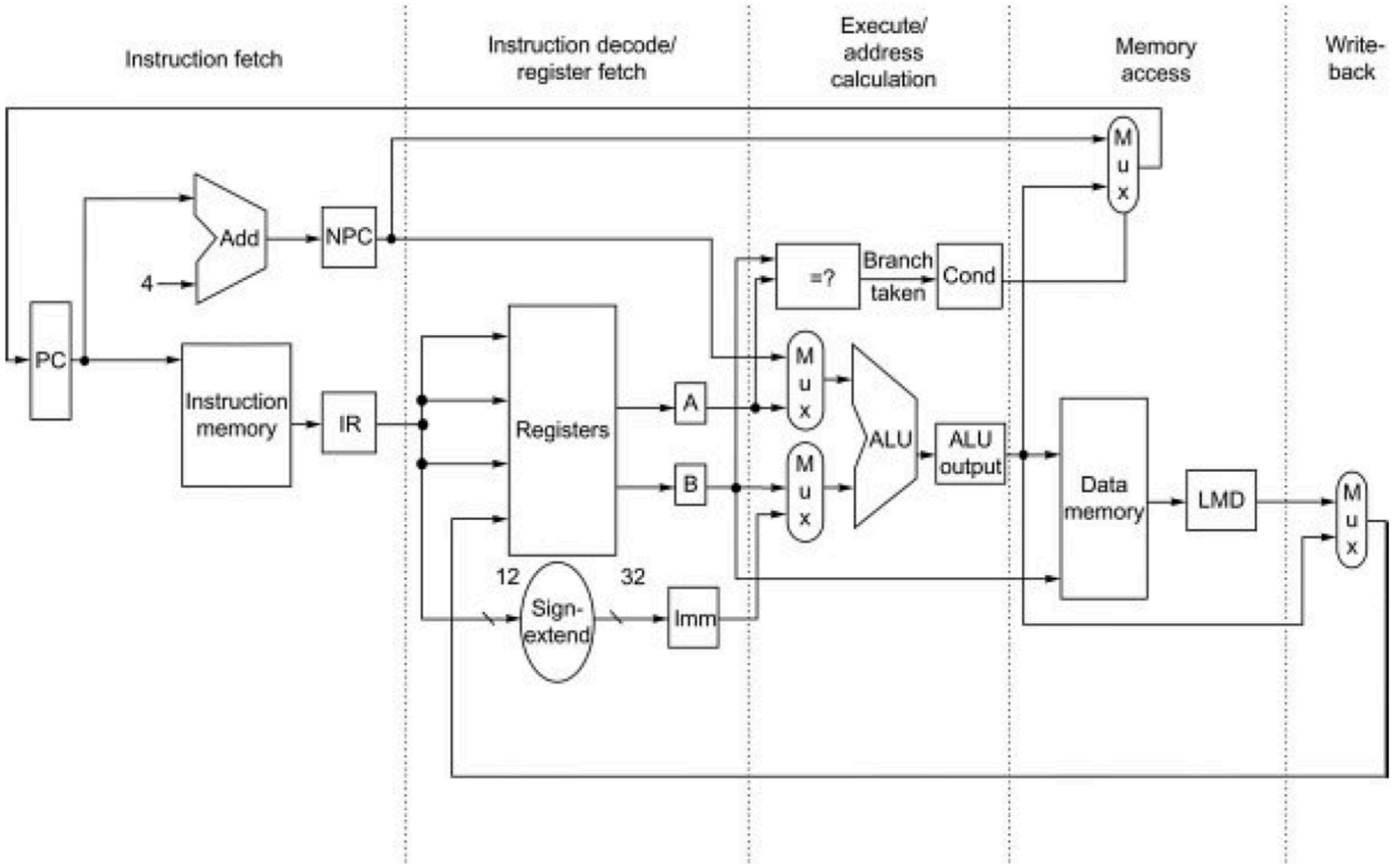
sys call num: 78 (readlinkat)

sys call num: 80 (fstat)

sys call num: 64 (write)

sys call num: 93,94 (exit)

In-order 5 stage



In-order 5 stage

- stage 사이 buffer 존재

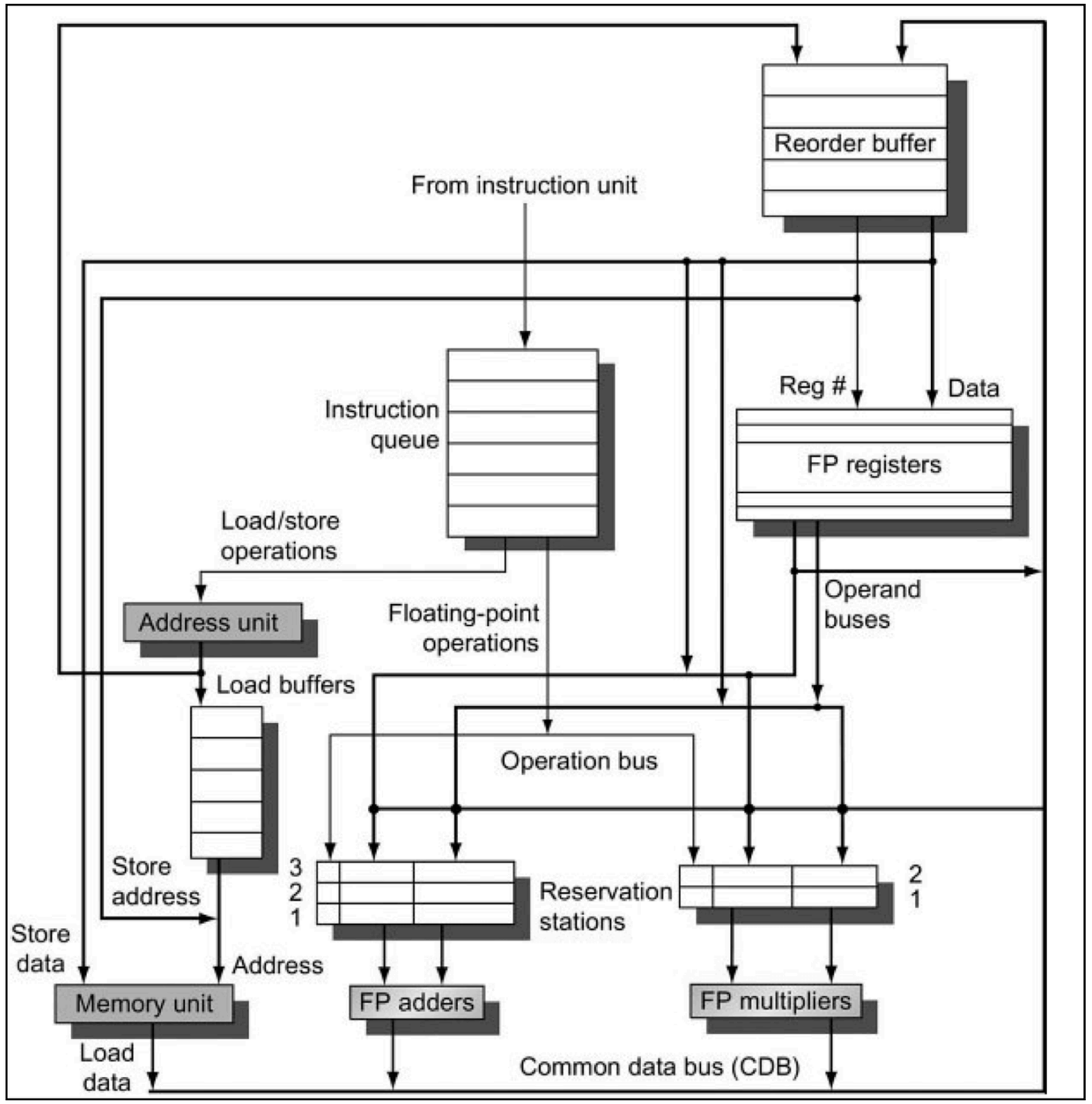
- RAW, WAW hazard 처리

  - 무한대 write port

  - 같은 레지스터에 대해서 write 할 수있으므로, 뒤의 명령어 앞의 명령어 write back 할 때까지 stall

- Fowarding

Out-Of-Order (Tomasulo)



Out-Of-Order (Tomasulo)

- Reservation Station
    - ALU, MulDiv, Address, Load Buffer
    - ROB queue
    - Instruction queue
    - Register Status

Out-Of-Order (Tomasulo)

- 5 stage
  - Fetch&decode : mem[pc] -> instruction queue
  - Issue : instruction queue -> RS & ROB
  - Execute : alu, muldiv, address, memory unit
  - Write result : CDB -> RS / ROB (store)
  - Commit : Branch / System call , write register

Out-Of-Order (Tomasulo)

- AMO instruction (LR, SC 제외)
  - Address unit
  - Memory unit
  - Commit 할 때 다시 Memory unit

Out-Of-Order (Tomasulo)

- 2 way super scalar
  - 이미 무한대 unit, write port 이기 때문에 fetch, issue 2 개씩
- 2 bit prediction
  - Fetch 할때 map[pc].taken bit 참조
  - Commit 할 때 틀리게 예측했으면 miss, taken bit 수정
- Exception handling 구현 필요

Bench mark 실행

hello_world

In-order

OOO

OOO + 2way

OOO + 2way + 2bit

```
riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 0 hello-riscv-dbg 2>out.txt
Hello, World!
[ clock ] 29993

riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 1 hello-riscv-dbg 2>out.txt
Hello, World!
[ clock ] 11694

riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 2 hello-riscv-dbg 2>out.txt
Hello, World!
[ clock ] 9689

riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 3 hello-riscv-dbg 2>out.txt
Hello, World!
[ clock ] 7566
```

Bench mark 실행 Qsort

```
riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 0 qsort-riscv-dbg 2>out.txt
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,2
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,7
103,104,105,106,107,108,109,110,111,112,113,114,115,116
0,141,142,143,144,145,146,147,148,149,150,151,152,153,1
178,179,180,181,182,183,184,185,186,187,188,189,190,191
5,216,217,218,219,220,221,222,223,224,225,226,227,228,2
253,254,255,256,257,258,259,260,261,262,263,264,265,266
0,291,292,293,294,295,296,297,298,299,300,301,302,303,3
328,329,330,331,332,333,334,335,336,337,338,339,340,341
5,366,367,368,369,370,371,372,373,374,375,376,377,378,3
403,404,405,406,407,408,409,410,411,412,413,414,415,416
0,SUCCESS

[ clock ] 96462008

riscv_5stage_simulator git/master*  35s
❯ ./riscv_simulator.out 1 qsort-riscv-dbg 2>out.txt
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,2
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,7
103,104,105,106,107,108,109,110,111,112,113,114,115,116
0,141,142,143,144,145,146,147,148,149,150,151,152,153,1
178,179,180,181,182,183,184,185,186,187,188,189,190,191
5,216,217,218,219,220,221,222,223,224,225,226,227,228,2
253,254,255,256,257,258,259,260,261,262,263,264,265,266
0,291,292,293,294,295,296,297,298,299,300,301,302,303,3
328,329,330,331,332,333,334,335,336,337,338,339,340,341
5,366,367,368,369,370,371,372,373,374,375,376,377,378,3
403,404,405,406,407,408,409,410,411,412,413,414,415,416
0,SUCCESS

[ clock ] 52492619
```

```
riscv_5stage_simulator git/master*  90s
❯ ./riscv_simulator.out 2 qsort-riscv-dbg 2>out.txt
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71
103,104,105,106,107,108,109,110,111,112,113,114,115,1
0,141,142,143,144,145,146,147,148,149,150,151,152,153
178,179,180,181,182,183,184,185,186,187,188,189,190,1
5,216,217,218,219,220,221,222,223,224,225,226,227,228
253,254,255,256,257,258,259,260,261,262,263,264,265,2
0,291,292,293,294,295,296,297,298,299,300,301,302,303
328,329,330,331,332,333,334,335,336,337,338,339,340,3
5,366,367,368,369,370,371,372,373,374,375,376,377,378
403,404,405,406,407,408,409,410,411,412,413,414,415,4
0,SUCCESS

[ clock ] 45511715

riscv_5stage_simulator git/master*  139s
❯ ./riscv_simulator.out 3 qsort-riscv-dbg 2>out.txt
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71
103,104,105,106,107,108,109,110,111,112,113,114,115,1
0,141,142,143,144,145,146,147,148,149,150,151,152,153
178,179,180,181,182,183,184,185,186,187,188,189,190,1
5,216,217,218,219,220,221,222,223,224,225,226,227,228
253,254,255,256,257,258,259,260,261,262,263,264,265,2
0,291,292,293,294,295,296,297,298,299,300,301,302,303
328,329,330,331,332,333,334,335,336,337,338,339,340,3
5,366,367,368,369,370,371,372,373,374,375,376,377,378
403,404,405,406,407,408,409,410,411,412,413,414,415,4
0,SUCCESS

[ clock ] 26288921
```

Bench mark 실행

Matrix multiply

In-order

```
riscv_5stage_simulator git/master*
❯ ./riscv_simulator.out 0 bench-matrix 2>out1.txt
Multiply Finished
Value is Correct.
-3283071

[ clock ] 42435729
```

OOO

```
riscv_5stage_simulator git/master*  15s
❯ ./riscv_simulator.out 1 bench-matrix 2>out1.txt
Multiply Finished
Value is Correct.
-3283071

[ clock ] 15304939
```

OOO + 2way

```
riscv_5stage_simulator git/master*  38s
❯ ./riscv_simulator.out 2 bench-matrix 2>out1.txt
Multiply Finished
Value is Correct.
-3283071

[ clock ] 12805252
```

OOO + 2way + 2bit

```
riscv_5stage_simulator git/master*  78s
❯ ./riscv_simulator.out 3 bench-matrix 2>out1.txt
Multiply Finished
Value is Correct.
-3283071

[ clock ] 5121269
```