EE312 <Lab 3: Single-cycle CPU Lab>

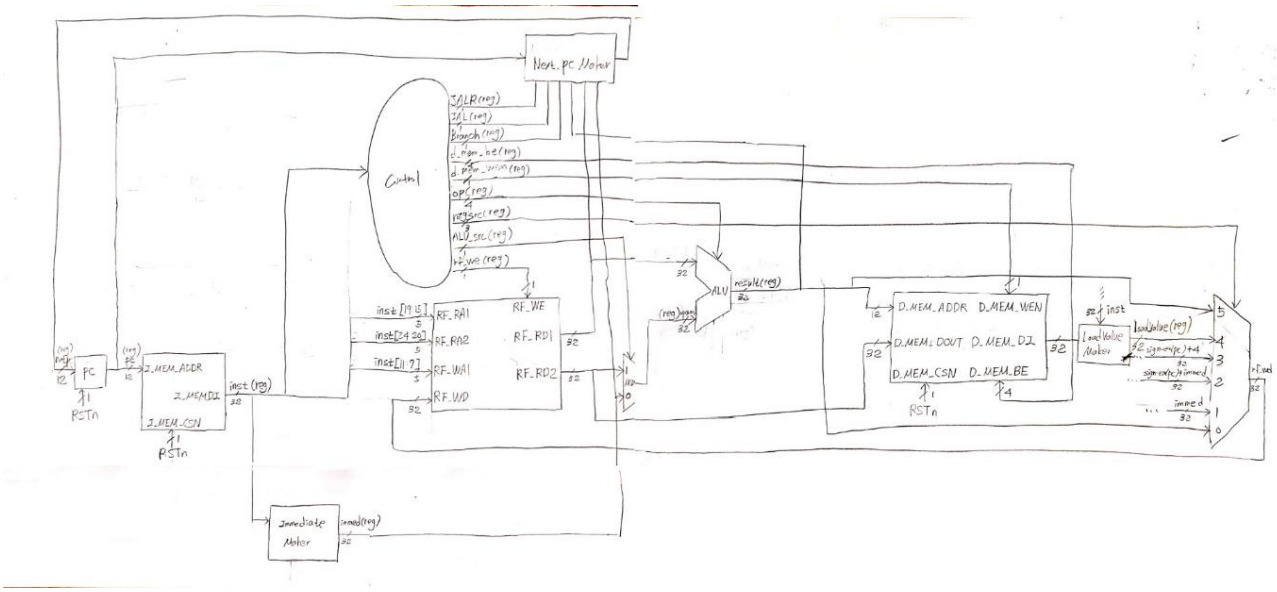
20140727 이동현 20170616 정희진

1. Introduction

Lab 3는 verilog를 이용하여 single-cycle CPU를 설계해보는 lab이다. single-cycle CPU는 매 clock cycle마다 하나의 instruction이 발생하는 CPU를 말한다. 다만 이번 lab에서 설계할 single-cycle CPU는 수업시간에 배운 MIPS version single-cycle CPU가 아닌 RISC-V version single-cycle CPU이다. 즉 lab3는 RISC-V version single-cycle CPU를 베릴로그로 구현하는 과정에서 modern computer architecture에 대해 보다 구체적으로 이해하는 것을 목적으로 한다.

2. Design

RISCV_TOP.v 파일에 코드를 쓰기 전에 다음과 같이 그림을 그려 디자인을 하였다.



3. Implementation

Instruction fetch, Register operand fetch, Memory operand fetch 과정은 이미 template 파일에 주어졌다. control part에 의해 정해지는 변수 값들은 다음과 같이 설정해주었다.

Reg_src = 3'b000 (1,2)
" = 3'b001 (LUI)
" = 3'b010 (AUIPC)
" = 3'b011 (4)
" = 3'b100 (6)
" = 3'b101 (others)

load_value = {{24{D_MEM_DI[7]}}, D_MEM_DI[7:0]} (LB)
" = {{16{D_MEM_DI[15]}}, D_MEM_DI[14:0]} (LH)
" = {{24{1'b0}}, D_MEM_DI[7:0]} (LBU)
" = {{16{1'b0}}, D_MEM_DI[15:0]} (LHU)
" : D_MEM_DI (others)

nxt_pc = pc + immed[12:0] (JAL, (Branch && result[0]==1))
= (RF_RD1 + immed[12:0]) & 0xfffe (JALR)
= PC+4 (others)

rf_we = 0 (5,7)
d_mem_wen 0 (7)
d_mem_be = 4'b1111 (SW)
" = 4'b0011 (SH)
" = 4'b0001 (SB)
" = 4'b0000 (others)

immed = {{21{inst[31]}}, inst[30:20]} (2(except SRAI, SRLI, SLLI), 6, JALR)
" = {{20{inst[31]}}, inst[7], inst[30:25], inst[11:8], 1'b0} (5)
" = {{21{inst[31]}}, inst[30:25], inst[11:8], inst[7]} (7)
" = {inst[31:12], {12{1'b0}}} (3)
" = {{12{inst[31]}}, inst[19:12], inst[20], inst[30:21], 1'b0} (JAL)
" = {{28{inst[24]}}, inst[23:20]} (SRAI, SRLI, SLLI)
" = {{32{1'b0}}} (others)

Branch = 1 (5)
op = 4'b0000 (ADD, ADDI, 6, 7)
" = 4'b0001 (SUB)
" = 4'b0010 (AND, ANDI)
" = 4'b0011 (OR, ORI)
" = 4'b0100 (XOR, XORI)
" = 4'b0101 (SLT, SLTI, BLT)
" = 4'b0110 (SLTU, SLTIU, BLTU)
" = 4'b0111 (SRA, SRAI)
" = 4'b1000 (SRL, SRLI)
" = 4'b1001 (SLL, SLLI)
" = 4'b1010 (BEQ)
" = 4'b1011 (BNE)
" = 4'b1100 (BGE)
" = 4'b1101 (BGEU)
" = 4'b1110 (others)

ALU_src = 1 (1,5)

4. Evaluation

모든 testcase를 통과하였으며 속도도 빨랐다.

(1) TB_RISCV_inst.v

```
# Test #    1 has been passed
# Test #    2 has been passed
# Test #    3 has been passed
# Test #    4 has been passed
# Test #    5 has been passed
# Test #    6 has been passed
# Test #    7 has been passed
# Test #    8 has been passed
# Test #    9 has been passed
# Test #   10 has been passed
# Test #   11 has been passed
# Test #   12 has been passed
# Test #   13 has been passed
# Test #   14 has been passed
# Test #   15 has been passed
# Test #   16 has been passed
# Test #   17 has been passed
# Test #   18 has been passed
# Test #   19 has been passed
# Test #   20 has been passed
# Test #   21 has been passed
# Test #   22 has been passed
# Test #   23 has been passed
# Finish:         24 cycle
# Success.
```

(2) TB_RISCV_forloop.v

```
# Finish:         73 cycle
# Success.
```

(3) TB_RISCV_sort.v

```
# Finish:       9408 cycle
# Success.
```

5. Discussion

simulation을 할 때 모든 testbench가 하나로 되어있어서 각각의 테스트를 할 때 마다 컴파일을 해야해서 불편함이 있었다.

test 결과 cycle의 개수가 나오는데 success가 나오는 것과 별개로 정해진 cycle의 수가 있다면 공지되면 더 좋을 것 같다.

6. Conclusion

모든 testcase를 통과했으며 실행도 빠르게 되는 것으로 보아 잘 설계된다는 것으로 추정된다. verilog로 구현하는 과정에서 single-cycle CPU의 control path, datapath에 대해 보다 구체적으로 이해할 수 있었다.