EE312

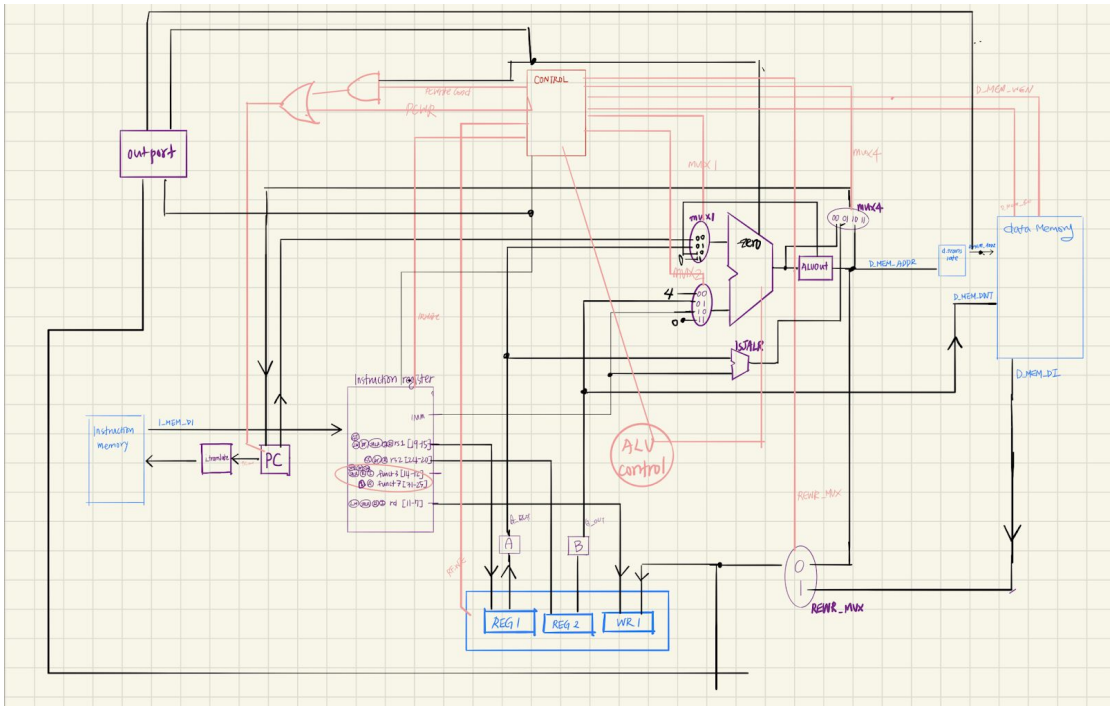
LAB4 REPORT

20160542 이현지 20160707 형준하

1. Introduction

- 이번 과제는 multi-cycle CPU를 구현해보는 lab이었다. single cycle과 다르게 각각의 stage에서 해주어야하는 일, 최소한의 resource를 사용하는 방법 그리고 synchronous한 memory / register 등 이전과는 달라진 조건들에 대해 어떻게 바뀌어야 하는지 고민해보는 시간을 가질 수 있었다.
- 수업에서 배운 바와 같이 IF, ID stage를 동일하게 그리고 최소한의 stage and resource를 사용한 효과적인 design을 만들기 위해 노력하였다. FSM을 이용할 예정이었기에 RT부터 작성하고 공통된 instruction에 대해서 같은 stage로 묶은 다음 각 stage에서 필요한 control을 알맞게 excel에 작성하였다. 이후 그림으로 회로를 design 함으로써 단계를 진행해나갔다.

2. Design



- module은 대체적으로 lab3에서와 비슷한 역할을 한다.
- control이라는 module을 통해 분홍색 control path를 그려주었고, 검은선을 따라 data path를 그리고 보라색을 통해 만든 module들을 표현하였다. ISJALR의 경우는 사실상 ALU와 같은 역할을 하지만 특정하게 JALR instruction에 대해서만 작동이 필요하여 간략하게 다른 모듈로 표현하였다. 다른 alu가 필요한 연산은 위에 큰 alu에서 진행이 된다.
- ALU control에 opcode 랑 stage를 넣어주어 필요한 연산을 할 수 있도록 alu에 넣어주었다. isJALR의 경우는 비록 alu지만 하는 연산이 정해져 있어서 굳이 alu control을 연결시키지 않았다.

3. Implementation

- 각각의 instruction type에 대한 RT이다.

[R-type]

IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID - 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 7
ALUout <= A OP B
WB – 11
RF[IR[11:7]] <= ALUout

[I-type]

IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID - 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 5
ALUout <= A OP sign-extend[IMM]
WB – 11
RF[IR[11:7]] <= ALUout

[LW]
IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID - 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 5
ALUout <= A + sign-extend(IMM)
MEM - 12
MEM[ALUout]
WB - 6
RF[IR[11:7]] <= MEM[ALUout]

[SW]
IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID - 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 5
ALUout <= A + sign-extend(IMM)
MEM - 4
MEM[ALUout] <= B

[JAL]
IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID – 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 3
PC <= ALUout
ALUout <= PC + 0
WB – 11
RF[IR[11:7]] <= ALUout

[JALR]
IF - 1
IR <= MEM[PC]
ALUout <= PC+4
ID - 2
PC <= ALUout
A <= RF[IR[19:15]]
B <= RF[IR[24:20]]
ALUout <= PC + (sign-extend(IMM)))
EX - 8
PC <= (A + sign-extend(IMM)) & 0xfffffffe
ALUout <= PC + 0
WB – 11
RF[IR[11:7]] <= ALUout

[BR]

IF - 1

IR <= MEM[PC]

ALUout <= PC+4

ID - 2

PC <= ALUout

A <= RF[IR[19:15]]

B <= RF[IR[24:20]]

ALUout <= PC + (sign-extend(IMM)))

EX - 9

PC <= ALUout if condition met

WB - 10

DUMMY (PC+4)

	MUX1	MUX2	MUX4	PCWR	ALUControl	RF_WE	PCWriteCond	IRWrite	D_MEM_WEN	REWR_MUX	D_MEM_BE	REPRESENTS
stage1	0	2b00	x	0	add	0	0	1	1	x	x	IF
stage2	0	2b10	2b10	1	add	0	0	0	1	x	x	ID
stage3	0	2b00	2b10	1	add	0	0	0	1	x	x	(JAL) EX
stage4	x	x	x	0	x	0	0	0	0	x	4'b1111	(ST) MEM
stage5	1	2b10	x	0	add/Op	0	0	0	1	x	x	(ST/LW/I) EX
stage6	x	x	x	0	x	1	0	0	1	1	x	(LW) WB
stage7	1	2b01	x	0	Op	0	0	0	1	x	x	(R) EX
stage8	0	2b00	2b01	1	add	0	0	0	1	x	x	(JALR) EX
stage9	1	2b01	2b10	0	OP	0	1	0	1	x	x	(BR) EX
stage10	0	2b00	x	0	x	0	0	0	1	x	x	(BR) WB
stage11	x	x	x	0	x	1	0	0	1	0	x	(JAL/JALR/I/R) WB
stage12	x	x	x	0	x	0	0	0	1	x	x	(LD) MEM

- IF, ID stage의 경우 수업시간에 배운 것과 같이 모든 instruction에 대해서 같은 instruction이 되도록 비록 필요하지 않은 일이더라도 하게 했다.

-Branch의 경우 WB stage 없이 3 stage로 끝낼 수 있지만 resource를 최대한 sharing하고 IF, ID stage를 공통으로 만들려다보니 4개의 stage를 만들어서 마지막 stage가 아닌 그 전 stage에서 PC에 값을 넣을 수 있도록 dummy instruction / stage를 넣어주었다.

4. Evaluation

- 기본적으로 주어진 testcase를 위주로 modelsim에 나온 wave가 의도한대로 잘 흘러가는지 확인하였다.
- inst의 경우 다 통과하였고 88 cycle이 나왔다.
- forloop의 경우 다 통과하였고 314 cycle이 나왔다.
- sort의 경우 다 통과하였고 42438 cycle이 나왔다.

5. Discussion

- cycle 수가 평가에 들어간다하여 JAL과 JALR에 알맞는 instruction을 찾고 implement를 잘한 것인가를 고민하는데 시간이 오래 걸렸다.

6. Conclusion

- 이번 과제를 하면서 수업시간에 배웠지만 놓쳤던 부분들을 하나씩 알아갈 수 있었다. 특히 stage를 한 두어번 정도 변경하였는데, 점차적으로 수업 시간에 배운 것과 비슷하게 stage를 나누게 된 것 같아 신기했다.