Computer Organization Lab 2

2019-16515 황현태

1. Forwarding.v

Ex stage(from ex/mem pipe register)에서 forward하는 경우와 mem stage(mem/wb)에서 forward하는 경우로 나누어 forward 신호를 전달한다. Rs1, Rs2 각각 mux를 넣고 1) EX stage에서 ALU에서 사용되는 operand rs1(or rs2)와 다음 stage에서 전달되고 있는 rd가 일치하는지 2) rs1(or rs2)이 0이 아닌지 3) RF에 write할 것이었는지를 확인하고 모두 만족하면 그 값을 alu의 operand rs1 혹은 rs2로 전달한다. 이때 rs2는 forwarding mux의 결과물이 다시 immediate와의 mux를 거쳐 최종적으로 ALU에 전달된다.

2. Hazard.v

(1) stall : Load의 경우에만 발생하는 stall condition을 감지한다. Load 명령어 바로 다음 명령어에서 loaded value를 사용하는 경우에 1 cycle stall한다. 1) ID stage에서 operand rs1 or rs2가 EX stage에서의 rd와 일치하는지 2) EX stage에서 load가 돌아가고 있는지 (mem_read) 3) rs1(or rs2)이 0이 아닌지 확인하고 모두 만족하면 stall을 전달한다. Stall이 전달되면 if/id, id/ex register에서는 다른 값은 hold하고 control signal을 모두 0으로 보내 nop을 1 cycle 실행한다. 현재의 PC 값 역시 그대로 hold한다.

(2) flush : jump, branch 명령어에서 다른 PC로 뛸 때 앞서 if, id ex stage의 명령어를 모두 kill한다. EX에서 branch가 resolve되고, 이 값이 pipe를 통해 mem으로 전달된 mem_taken 값을 이용하여 flush 여부를 결정한다. Flush는 branch와 jump 명령어에 대해 켜져야하므로, control signal 중 jump와 taken 신호를 or 하여 flush로 사용한다. Flush 할 때, if/id, id/ex, ex/mem register는 모든 값을 전부 0으로 초기화한다.

Flush는 stall보다 우선적으로 실행되어야 한다. 따라서 flush, stall이 동시에 켜진 경우 PC는 hold하지 않고 next PC를 로드한다. 즉 !flush && stall 일때 PC를 hold한다.

3. Pipeline registers (IF/ID, ID/EX, EX/MEM, MEM/WB)

값을 synchronous하게 다음 stage로 전달한다. Flush일 때 모든 레지스터를 0으로 초기화(kill)하고, stall 시에는 value는 hold, control signal은 0으로 설정하여 nop을 실행한다. 이 역시 flush가 우선순위이므로, if문을 flush – stall 순으로 배치한다.

4. Simple_cpu.v

앞서 설명한 모듈들을 모두 이어주는 부분이다. Control signal에 따라 적절한 path를 이어줄 수 있도록, mux를 배치한다. 이번 lab2에서는 어떤 stage의 값을 forward 할 것인지 결정해주는 forward mux(rs1, rs2)가 추가되었다.