**컴퓨터 구조 Project 3, Simulating Pipelined Execution**

201911039 김태연

1. 수행 과제에 대한 간략한 설명

2. Pipeline state Register

각 레지스터는 struct 타입으로 생성하였으며, 구조체 멤버 변수로 int 변수들을 갖는다.

**1. IF\_ID**

- INSTR: fetch해 온 PC 주소에서 읽어 온 instruction이다.

- NPC: fetch해온 PC를 저장한다. 즉, IF stage에 와 있는 PC를 저장한다.

**2. ID\_EX**

- NPC: ID stage에서 실행된 instruction의 PC를 저장한다.

- Rs: Instruction의 25-21번째 bit 값을 저장한다.

- Rd: Instruction의 20-16번째 bit 값을 저장한다.

- Immd: Instruction의 하위 16bit를 저장한다.

- Shamt: Instruction의 15-11번째 bit값을 저장한다.

- JMP\_TARGET: J, JAL, JR인 경우, ID stage에서 계산된 Jump target을 저장한다. Jump instruction이 아니면 이 값은 0이다.

- ReadData1: Rs에서 읽어온 값을 저장한다.

- ReadData2: Rt에서 읽어온 값을 저장한다.

- ALU\_SIG: OP코드를 보고 지정된 ALU연산에 대핸 시그널을 저장한다. Instruction의 종류에 따라 서로 다른 ALU signal을 가진다. 편의상 op code, R-type의 경우 funct로 설정하였다.

- WB\_SIG: Write Back이 일어나는지를 나타내는 시그널. 일어나면 1, 일어나지 않으면 0으로 설정된다.

- MEM\_SIG: Memory에 접근하는 작업이 일어나는지를 나타내는 시그널. 0이면 일어나지 않고, 1이면 4byte word단위의 접근이 일어나며, 2면 1byte단위의 접근이 일어난다. 이 신호는 lb와 sb를 위해 사용된다.

- JMP\_SIG: Jump가 일어나는지를 나타내는 시그널.

**3. EX\_MEM**

- WriteRegs: Write Back이 일어나는 경우 destination register의 번호

- MEM\_SIG: 메모리 컨트롤 시그널. memwrite(01), memread(10)

- WB\_SIG: write back 컨트롤 시그널. 0이면 writeback일어나지 않고, 1이면 일어난다….? **차라리 writeRegs를 확인을 해서 이게 존재하는 경우 writeback을 해 주면 되지 않나? 이렇게 하면 될 듯.**

- ALU\_OUT: EX stage ALU의 연산 결과를 저장한다.

- BR\_TARGET: EX stage에서 계산한 Branch 주소를 담는다.

만약에 branch와 jump가 한 사이클 내에 들어와 있으면 어떡해……..??? branch다음에 바로 점프가 있으면? branch가 일어난다면 flush가 일어나기 때문에 문제 없고, branch가 일어나지 않으면 jump instr이 실행되겠지.

-

**4. MEM\_WB**

- WriteRegs: Write Back이 일어나는 경우 destination register의 번호

- ALU\_OUT: EX stage ALU의 연산 결과

- MEM\_OUT: MEM stage에서 읽어온 값

2. 컴파일 및 실행 방법과 환경

**Ubuntu 20.04** 환경에서 **g++ 버전 9.3.0**을 이용하여 다음 명령어로 컴파일 하였다.



위의 컴파일 명령으로 실행 파일인 runfile을 생성하여 다음과 같은 명령어로 실행하였다.



위의 예시에서는 필수 옵션만 포함했으나, 과제에서 제시된 바(아래)와 같이 여러 옵션을 주어 테스트했다.

**./runfile <-atp 또는 -antp> [-m addr1:addr2] [-d] [-p] [-n num\_instruction] <input file>**