201911146 정소희 컴퓨터구조 과제 P3

실행 환경:

- C 사용
- wsl ubuntu bash 환경에서 실행하였으며, 모두 올바르게 작동함을 확인하였다.

컴파일 방법:

- $ gcc -o runfile pipeline.c
- gcc -o runfile pipeline.c
- wsl ubuntu bash 환경에서 gcc 9.4.0 버전을 이용하여 위의 명령어로 컴파일 하였다.

실행 방법:

- 과제에 쓰인 설명대로 다음과 같이 (예시) 로 입력한다
- $ ./runfile -antp -m 0x10000000:0x10000010 -d -p -n 100 sample2.o
- ./runfile -atp -m 범위1:범위2 -d -p -n 명령어개수 에뮬레이트할오브젝트파일이름

구현 방법:

사용한 Register:

- 각 Register는 Immediate Register와 State Register로 구분된다.
- wb-mem-exe-id-if를 실행하는 과정에는 Immediate Register에 값을 저장하고, 5 단계가 모두 끝난 뒤 Immediate Register의 값을 State Register에 한번에 업데이트 시켜 준다. (동시에 파이프라인이 실행되는 것을 구현하기 위하여 사용하였다.)
- Control Register들은 교과서에 적힌 것들을 사용하였다.

ID 단계

- PC = I_PC;

PC = Program Counter 와 Immediate Program Counter

IF 단계

- IF_ID_NPC = I_IF_ID_NPC;

```
    -               IF_ID_INSTR = I_IF_ID_INSTR;
```

NPC = ID 단계에서 실행되고 있는 명령어의 주소 + 4
INSTR = ID 단계에서 사용할 명령어, 메모리에서 읽어옴
EXE 단계

```
    -               ID_EX_NPC = I_ID_EX_NPC;
    -               ID_EX_RD1 = I_ID_EX_RD1;
    -               ID_EX_RD2 = I_ID_EX_RD2;
    -               ID_EX_RS = I_ID_EX_RS;
    -               ID_EX_RT = I_ID_EX_RT;
    -               ID_EX_IMM = I_ID_EX_IMM;
    -               ID_EX_RD = I_ID_EX_RD;
    -               ID_EX_SHAMT = I_ID_EX_SHAMT;
```

NPC = EXE 에서 실행되는 명령어의 주소 + 4
RD1 = 첫번째로 읽은 값
RD2 = 두번째로 읽은 값
RS = rs
Rt = rt
Rd = rd
Imm = ID 단계에서 SignExtended 된 Imm 값
SHAMT = r 타입 shift 변수

MEM 단계

```
    -               EX_MEM_NPC = I_EX_MEM_NPC;
    -               EX_MEM_BR_TARGET = I_EX_MEM_BR_TARGET;
    -               EX_MEM_ALUOUT = I_EX_MEM_ALUOUT;
    -               EX_MEM_RD2 = I_EX_MEM_RD2;
    -               EX_MEM_REGISTERRD = I_EX_MEM_REGISTERRD;
```

NPC = MEM 단계에서 사용되는 명령어의 주소 + 4
BR_TARGET = Branch Target
ALUOUT = ALU 결과값
RD2 = 두번째로 읽은 값
REGISTERRD = 레지스터에 나중에 쓸 주소

WB 단계

```
    -               MEM_WB_NPC = I_MEM_WB_NPC;
    -               MEM_WB_ALU_OUT = I_MEM_WB_ALU_OUT;
    -               MEM_WB_MEM_OUT = I_MEM_WB_MEM_OUT;
    -               MEM_WB_REGISTERRD = I_MEM_WB_REGISTERRD;
```

NPC = WB 단계에서 사용되는 명령어의 주소 + 4
ALUOUT = ALU 결과값
MEM_OUT = 메모리 읽은값
REGISTERRD = 레지스터에 쓸 주소

동작 구조 :