2024 Spring / Computer Architecture / Lab-2
주제: Multi-cycle Processor 및 파이프라인 구현

1. 목표

* 파이프라이닝(pipelining)은 IF, ID, EX, MEM, WB의 5개의 단계로 이루어짐.
  - 지난 과제#01을 통해 ID(instruction decoding)에 대한 구현을 수행함.
* 나머지 단계(EX, MEM, WB) 동작을 수행할 수 있는 코드의 구현을 목표.
* pipelining에 대한 이해, simulation에 대한 이해 제고

2. 과제

* 공통사항:

- 주어진 뼈대 코드는 주어진 명령어를 수행할 수 있는 multi-cycle processor를 모델링한 것이다. 주어진 과제를 수행할 수 있도록 뼈대코드를 완성하시오.
- 뼈대 코드의 완성을 위해 수정해야 할 부분은 //TODO로 표기되어 있으며, 모든 수정해야 할 코드는 clock_cycle() 함수 내에 있음.
- 프로그램의 동작 중간단계를 기본적으로 표시하고 있으나, 추가적인 정보 획득을 위한 printf(...) 함수를 자유롭게 사용할 수 있음.
- 구현하고자 하는 시스템은 32-bit 시스템을 가정하고, 명령어에서 주소를 표현하기 위해 사용되는 offset 또는 imm값은 byte단위임.
  따라서, 메모리를 구현한 memory 배열의 각 원소는 4-byte(i.e., 32-bit) 단위의 주소로 접근됨을 가정함.

* Part#1 50% - Program A를 수행할 수 있도록 주어진 뼈대 코드를 완성하시오.

- 주어진 프로그램은 “hw02_programA.txt” 파일 내 작성된 명령어를 한 줄씩 읽어가며 수행됨.
- 모든 명령어에서 사용되고 있는 register들은 서로 의존적이지 않으며 hazard가 존재하지 않음.

* Part#2 50% - Evaluation: 구현한 프로그램에 대한 분석 및 이해

- 프로그램의 동작을 분석하여 보고서로 작성
- 구현한 항목 및 전체 프로그램 코드의 동작에 대해 구체적으로 설명

: 프로그램의 최종 수행 결과에 대한 실행 결과화면 캡처 포함

* Part#Optional 10% - Program B를 수행할 수 있도록 data forwarding을 구현

- Program B는 “hw02_programB.txt” 파일 내 작성된 명령어를 읽고 수행함
  : Program B를 수행하기 위해, fetch() 함수 내 fopen 파일 이름 수정
- Program B는 사용하는 레지스터 별 의존성이 있으며 data hazard가 존재함. 따

라서 의존성에 의한 성능하락을 방지하기 위해 **data forwarding을 구현.**
:HINT - 파이프라인 스테이지 간 정보를 저장하는 register 구조체인 "*PipelineRegister*" 구조체 변수들 사이의 정보 전달을 통해 forwarding이 지원될 때와 동일한 동작을 구현하고 시뮬레이션 가능함.

- 구현에 성공한 경우, Optional 구현을 수행했다는 사실과, 관련된 내용을 보고서에 기술함.

<< Optional 수행에 의한 평가 설명 >>
: 해당 점수는 전체 과제 사이에서 자유롭게 추가 점수로써 사용될 수 있음.
단, 이는 과제의 최대 점수를 초과하지 못함.
(예) 각 100점 만점인 3개의 과제(최대점수 300점) 중, 1차 90점, 2차 100+10(optional)점, 3차 100점인 학생은 총합 300점의 점수를 부여받음.
(예) 1차 100점, 2차 110점, 3차 100점인 학생은 총점 300점을 부여받음.