컴퓨터구조 프로젝트2 보고서

DGIST 기초학부 201611144 임지훈

1. 작성한 과제에 대한 간략한 설명

메모리는 전체 가상 메모리를 만드는 것은 공간 낭비가 심하기에 Data Section 메모리와 Text Section 메모리를 크기를 이용해 만들었다. 메모리는 각 바이트에 접근할 수 있기때문에 전부 32 Byte임에도 Char로 만들었다. 이와 달리 명령어 버퍼나 PC 등은 int형으로 만들었다. Register File은 32개의 정수 배열로 만들었다.

간단하게는 실행 명령어을 읽고 로딩해서 MIPS단계에 따라 실행한다. 좀 더 자세하게는 먼저 파일과 옵션들을 받아 모드를 설정한다. 그리고 레지스터를 초기화하고 메모리에 데이터와 텍스트를 로딩한다. 다음 프로세서 시뮬레이션 파트에서 명령어, 제어비트 그리고 회로 내의 값을 의미하는 변수들을 선언하고 n옵션 num_instruction만큼 명령어를 실행한다. 먼저 Instruction Fetch 파트에서 Program Counter를 증가시키고 Instruction Decode에서는 명령어를 분해하고 제어 신호를 생성한다. Instruction Execution에서는 ALU에서 연산이 일어나는 것을 시뮬레이션 했으며 PCSrc (Program Counter가 어떤 값을 가질지 결정하는 제어 신호)를 결정한다. Memory 파트에서는 메모리를 읽거나 쓴다. WriteBack에서는 레지스터를 읽거나 쓴다. 마지막으로 D옵션이 있다면 레지스터 상황을 출력한다. 이렇게 필요한 만큼 명령어를 실행하고 나서는 D옵션이 없다면 레지스터 상황을 출력하고 메모리 옵션에 따라 해당 메모리 영역을 출력한다.

2. 과제의 컴파일 방법 및 컴파일 환경/ 과제의 실행 방법 및 실행 환경

C언어를 사용하여 코딩했고 Ubuntu 18.04 터미널에서 프로젝트 파일에 들어간 뒤 gcc 7.5.0 버전을 이용하여 main.c이 있는 창에서 다음과 같이 컴파일 하였다.

```
gcc -o runfile main.c
```

3. 과제의 실행 방법 및 실행 환경

만약 sample.s를 어셈블러로 처리한 sample.o를 실행하기위해서는 터미널에서 다음과 같이 실행하였다.

```
./runfile sample.o
```

그리고 다시 sample2.o를 실행할 때에는 다음과 같이 했다.

```
./runfile sample2.o
```

다양한 옵션을 테스트하기위해 ./runfile -n 0 sample.o 나 ./runfile -m 0x400000:0x400010 -d -n 3 sample.o 등을 사용해 특정 메모리를 참조할 수 있고(-m 원하는 메모리 영역) 명령어 수행 후 레지스터 결과를 출력할 수 있으며 원하는 갯수만큼 명령어를 실행시킬 수 있다.