Project 2 : Building a Simple MIPS Emulator Report

추성재 (201911185)

1. 개발 환경

   이번 프로젝트를 개발할 때 사용한 언어는 C++이다. Ubuntu 22.04에서 주로 개발했다. 사용한 컴파일러는 g++ 11.3.0 버전이며, make를 이용해 빌드했다. 또한 최종 결과를 Ubuntu 20.04, g++ 9.4.0 version을 이용해 빌드해보았고 성공적으로 빌드 되었다.

2. 컴파일 방법

   압축파일 속 "finalVersion" 폴더 속에 Makefile이 있다. 다음과 같은 명령어를 통해 컴파일하면 된다.

   ```
   $make clean; make
   ```

   Make가 작동하지 않는다면, "include" 폴더 속 헤더파일과 "src" 폴더 속 소스코드들을 한 디렉토리에 넣은 후 다음과 같은 명령어를 통해 컴파일하면 된다.

   ```
   $g++ -o runfile 00main.cpp 01datamanage.cpp 02doInstr.cpp 03memRegManage.cpp
   ```

   컴파일을 완료하면 "runfile"이라는 이름의 실행파일이 나온다.

3. 실행 방법

   실행파일 뒤에 과제 안내에 명시된 인자를 다음과 같이 입력 후 실행하면 된다. 인자들의 순서나, 목적 파일을 제외한 인자들의 유무는 상관없다.

   ```
   ./runfile –[m addr1:addr2] -d [-n num_instruction] sample.o
   ```

4. 코드의 flow

   이번 프로젝트의 코드는 다음과 같이 네 개의 소스코드로 이루어져 있다.

   - 00main.cpp

     메모리와 레지스터의 선언 및 초기화, 목적 파일 불러온 후 메모리에 로드, Emulator 옵션을 받아서 실행 방식 설정, instruction 수행 등과 같은 전반적인 흐름을 담당한다.

     메모리는 Big Endian 형식으로 값들이 저장되어 있다.

   - 01datamanage.cpp

     Project 1에서 넘어온 소스코드로 담당하던 기능이 Project 1에서와 유사하다. 일부 함수들은 Project 1에서 기능이 추가되었고, StrIsNum, BitExtend는 새로 추가되었다.

함수명	기능
DecimalToHex	unsigned int를 16진수 형태의 문자열로 바꾸는 함수다. Project 1과는 달리, isSigned 인자가 추가되어 기본값인 false일때는 unsigned int로 처리하고, true일때는 int형으로 명시적 casting 후 처리한다.
SplitLine	Project 1에서와 동일하게 string str가 들어오면 char seperator를 기준으로 분리한 후 이를 담은 vector를 반환한다.
StrHaveChar	string str에 char c가 있는지 여부를 확인하고 있으면 true, 없으면 false를 반환한다.
NumToBit	String 형식의 수를 받으면 이 중 least significant bit를 size의 크기에 맞춰서 string 형식으로 반환한다.
StrIsNum	string str이 숫자인지 판별한다. isHex의 기본값 false일때는 숫자를 10진수를 기준으로 판별하고, isHex가 true일때는 16진수를 기준으로 판별한다.
BitExtended	string 형식의 bit를 받으면 이를 isSignEx가 true일때는 sign extend, false일때는 zero extend한 후 그 결과를 string 형식의 bit로 반환한다.

- 02doInstr.cpp

  Instruction을 실제로 수행하는 함수로 Instruction의 형식에 따라서 구분 되어있다.

함수명	기능
doInstr_R	R형식의 Instruction을 funct 값을 이용해 구분한 후 실행한다.
doInstr_I	I형식의 Instruction을 op 값을 이용해 구분한 후 실행한다.
doInstr_J	J형식의 Instruction을 op 값을 이용해 구분한 후 실행한다.

- 03memRegManage.cpp

  Instruction을 decode하는 함수, 메모리에 값을 읽고 쓰는 함수, 레지스터와 메모리 속 값을 출력하는 함수로 이루어져 있다.

함수명	기능
RegMemCout	모든 레지스터 속 값을 출력한다. 이때, isMemout이 true일 경우 입력 받은 메모리 주소에 해당하는 메모리 속 값 또한 같이 출력한다.
instrDecode	PC 값이 가리키는 메모리에서 Instruction을 불러온 후 이를 분리해 unsigned int 형태로 변환한다. 그 후 Instruction의 형식에 따라 각각의 doInstr 함수들을 실행한다.
MemLoad	메모리 주소를 받은 후 그 주소에서 값을 읽어온다. isWord가

	true일 경우 Word 단위로 불러오고, false일 경우 byte 단위로 불러온다.
MemWrite	메모리 주소 addr과 값 value를 받은 후 메모리에 저장한다. isWord가 true일 경우 Word 단위로 데이터를 저장하고, false일 경우 bit 단위로 데이터를 저장한다. 이때, 메모리에 값을 저장할 때는 big endian을 고려해서 바이트 단위로 저장된다.

코드의 flow는 다음과 같다.

1. 코드가 시작되면 레지스터를 표현한 Reg와 Mem, PC를 초기화한다.
2. 프로그램이 시작될 때 받은 argument에 따라 isMemout과 같이 이후 프로그램 실행에 영향을 주는 변수들을 설정한다. 이때 argument의 수가 너무 많거나, 유효하지 않은 argument일 경우에는 에러 메시지를 출력하고 프로그램을 종료한다. 이때, 레지스터의 값과, 메모리가 출력되도록 설정되었다면 메모리 값도 에러 메시지와 같이 출력한다.
3. 목적 파일을 연 후, 목적 파일이 빈 파일인지, 유효한 파일인지 검사한다. 이후 목적 파일의 text section 속 내용과 data section 속 내용을 Mem에 넣는다.
4. 이후 PC 값에 따라 Instruction을 실행한다. 이때, -d 옵션이 있으면 Instruction이 끝날 때마다 레지스터 값과, 메모리 값을 출력하도록 되었으면 메모리 값도 같이 출력한다. PC값이 프로그램의 끝을 가리키면 실행을 중단한다.
5. 에뮬레이터가 끝날 때 레지스터 값과, 메모리 값을 출력하도록 되었으면 메모리 값도 같이 출력하며 프로그램을 종료한다.