컴퓨터 구조 Project 3 보고서

201711174 조준규

0. 컴파일/실행 방법, 환경

코드는 Windows 10 운영체제에서 Visual Studio 2019 IDE를 사용하여 C++로 작성하였다. 테스트는 WSL(Windows Subsystem for Linux)의 Ubuntu 18.04 환경에서 진행하였다. 컴파일러는 g++ 7.5.0 버전을 사용하였고 옵션으로 -std=c++14를 추가하여 컴파일하였다.

컴파일을 위한 command는 makefile을 만들어 사용하였다. 파일의 내용은 다음과 같다.

```
runfile : main.o console.o
    g++ -o runfile main.o console.o -std=c++14

main.o : main.cpp
    g++ -c main.cpp -std=c++14

console.o : console.h console.cpp
    g++ -c console.h console.cpp -std=c++14

clean :
    rm main.o
    rm console.o
    rm runfile
```

위의 파일은 소스 코드와 함께 첨부하였다. 첨부한 makefile을 소스 코드와 같은 폴더에 넣고 터미널에 make를 입력하여 컴파일할 수 있다. 소스 코드가 업데이트된 경우에는 make clean을 입력하여 이전에 컴파일 된 파일들을 삭제한 후 다시 컴파일 할 수 있다.

$ make clean

$ make

$ ./runfile -m 0x400000:0x400010 -d -n 20 sample.o

1. console.h console.cpp

Console이라는 class를 생성하여 콘솔창에 출력하는 부분을 함수로 따로 구현하였다.

또한, pipeline의 각 stage마다 함수를 생성하여 구현하였다.

unsigned int ifStage(unsigned int pc, vector<string> textList, unsigned int ifidRegList[])

unsigned int idStage(unsigned int ifidRegList[], unsigned int controlUnit[], unsigned int regList[], unsigned int idexRegList[])

unsigned int exStage(unsigned int idexRegList[], unsigned int exmemRegList[], unsigned int regList[])

unsigned int memStage(unsigned int exmemRegList[], vector<string> textList, vector<string> dataList, unsigned int memwbRegList[], unsigned int idexRegList[], unsigned int regList[])

unsigned int wbStage(unsigned int controlUnit[], unsigned int memwbRegList[], unsigned int regList[])

stage register와 control unit은 다음과 같이 정하였다.

unsigned int controlUnit[9];

// RegDst, ALUOp, ALUSrc, Branch, MemRead, MemWrite, RegWrite, MemtoReg, Jump

unsigned int ifidRegList[5];

// NPC, Instr, pc, rs, rt

unsigned int idexRegList[19];

// NPC, op, rsd, rtd, IMM, rt, rd, target, RegDst, ALUOp, ALUSrc, Branch, MemRead, MemWrite, RegWrite, MemtoReg, pc, rs, rt

unsigned int exmemRegList[10];

// op, ALU_OUT, rtd, rd, Branch, MemRead, MemWrite, RegWrite, MemtoReg, pc

unsigned int memwbRegList[7];

// op, MEM_OUT, ALU_OUT, rd, RegWrite, MemtoReg, pc

void printPCReg(unsigned int pc, unsigned int regList[])

<매개변수>

- unsigned int pc는 program counter를 저장하는 매개변수이다.
- unsigned int regList[]는 각 register의 값들을 저장하고 있는 배열을 저장하는 매개변수이다.

<함수설명>

현재 pc와 register의 값들을 출력하기 위해 두 변수를 불러온다. 각 값들은 콘솔에 16진수로 표현된다.

void printMemory(string addr1, string addr2, vector<string> textList, vector<string> dataList)

<매개변수>

- string addr1은 -m 인자로 인한 주소 범위의 시작 주소를 저장하는 매개변수이다.
- string addr2은 -m 인자로 인한 주소 범위의 끝 주소를 저장하는 매개변수이다.
- vector<string> textList는 instruction의 값들의 집합을 저장하는 매개변수이다.
- vector<string> dataList는 data의 값들의 집합을 저장하는 매개변수이다.

<함수설명>

먼저, textList와 dataList에는 사전에 저장해 놓은 값들이 들어있다. 함수에서는 addr1과 addr2 사이의 모든 주소들에 대해서 textList와 dataList에 할당되어 있는지 확인한다. 할당되어 있는 것이 확인된다면 그 주소에 들어있는 값을 콘솔에 16진수로 출력한다. 만약 찾지 못했다면 할당되어 있지 않다고 가정하고 0x0을 출력하게 된다.

2. main.cpp

Main 함수에서는 크게 명령줄의 인자를 받는 과정, 파일을 읽어 메모리에 instruction과 data를 저장하는 과정, 저장된 데이터들을 이용하여 실제로 구현하는 과정으로 나뉘어져 있다.

2.0 main 함수 인자 확인

Main 함수의 int argc와 char* argv[] 인자를 통해 command의 인자를 받게 된다. -m 인자와 -d 인자, -p 인자의 경우 flag 변수를 생성하여 인자가 입력되었는지를 따로 저장한다. 후에 콘솔에 출력할 때 출력 여부를 결정짓는 변수가 된다. -n 인자의 경우에는 num_instruction 변수에

실행할 instruction 수에 대한 값을 저장하였다. 값이 들어오지 않은 경우에는 -1로 초기화해 놓았다. 이는 후에 instruction을 num_instruction 만큼만 실행할 것인지, return address를 만날 때까지 실행할 것인지 결정하는 변수가 된다.

2.1 instruction과 data 저장

파일은 inputFile이라는 ifstream을 통해 한 줄마다 읽게 된다. 처음 두 줄의 경우 text section과 data section의 size를 나타내므로 따로 변수에 저장하여 파일 내에서 몇 번째 줄 까지가 instruction이고 data인지 확인하는 데에 사용한다.

이후에는 instruction과 data의 값들을 STL vector에 저장한다. 각 원소는 16진수 형태로 된 2자리 string으로 되어있다. 따라서 예로 첫 data의 정보는 dataList[0], dataList[1], dataList[2], dataList[3]에 16진수 2자리씩 앞에서부터 저장되는 것이다. 8자리가 안되는 데이터들의 경우 앞의 나머지 자리가 0이라는 뜻이므로 부족한 만큼 "0"을 앞에 이어 붙여주어 하나의 데이터가 4칸을 차지하도록 하였다.

이 방법을 선택한 이유는 각 vector의 인덱스가 메모리의 주소에서 시작 주소로부터의 offset에 해당하기 때문이다. 예로 두 번째 instruction의 값은 textList[4], textList[5], textList[6], textList[7]에 16진수 2자리씩 저장될 것인데, 이는 각 인덱스에 0x400000만큼 더한 0x400004, 0x400005, 0x400006, 0x400007에 16진수 2자리씩 저장되어 있다는 것과 같은 의미가 된다.

2.2 instruction 구현

구현을 시작하기 전에 program counter와 register의 값들을 각각 0x400000과 0으로 초기화하였다. Register의 경우에는 편의성을 위해 32칸짜리 배열을 생성하여 인덱스로 접근할 수 있도록 하였다.

이후에는 program counter가 return address가 될 때까지 계속 루프가 돌아간다. 한 루프 속에는 현재 program counter에 해당하는 instruction의 값을 불러오고 instruction의 opcode에 따라 동작이 달라지도록 하는 과정이 들어있다. 이 속에는 이와 더불어 -n 인자가 입력된 경우 그 개수만큼 루프가 돌고 break하는 부분과 -d 인자가 입력된 경우 매 instruction마다 콘솔에 각 정보를 출력하는 부분도 들어있다.

루프가 종료된 후에는 최종 program counter와 register 값들이 출력된다. -m 인자에 따라 각 주소에 해당하는 메모리를 콘솔에 출력하는 부분도 존재한다.

함수 호출을 통해 구현하였다.