HW1_201911050_Damrok DO

1. 실행 환경

Windows terminal의 ubuntu 환경에서 진행하였다. 다음과 같은 명령어로 실행하였고,

결과로 object file들이 생성되었다.

```
root@DESKTOP-8U739LJ:/mnt/c/Users/test/downloads/hw1# g++ -o runfile hw1_201911050.cpp
root@DESKTOP-8U739LJ:/mnt/c/Users/test/downloads/hw1# ./runfile sample.s
root@DESKTOP-8U739LJ:/mnt/c/Users/test/downloads/hw1# ./runfile sample2.s
root@DESKTOP-8U739LJ:/mnt/c/Users/test/downloads/hw1# ls
hw1_201911050.cpp  runfile  sample2.o  sample2.s  sample.o  sample.s
root@DESKTOP-8U739LJ:/mnt/c/Users/test/downloads/hw1#
```

2. 코드 설명

```
struct addr_Tag {
        unsigned long long int address;
        char* tag = {};
};
struct addr_Data {
        unsigned long long int address;
        unsigned long long int data;
};
struct Memory {
        addr_Tag Set[512];
        addr_Data Set2[2048] = {};
```

(line 10) Tag와 데이터들을 저장해 놓기 위해 Tag에 따른 주소와 주소에 따른 데이터들을 구조체로 만들어 Memory라는 구조체 안에 넣어 놓았다.

```
// "LA"
char LAchecker[128] = {};
strcpy(LAchecker, instructions[readcounter]);
int LAcount = 0;
bool lacheck = false;
char* latok = strtok(LAchecker, " ");
while (latok) {
        if (!strcmp(latok, "la") and LAcount<2) {
                lacheck = true;
                strtok(NULL, " ");
                char* nametec = strtok(NULL, " ");
                char* name = strtok(nametec, "\n");
                unsigned long long address = Datamem.get_addr(name);
                //printf("addr is: 0x%llX\n", address);
                if ((address)%0x1000 != 0 and address){
                        textcounter += 4;
                        break;
                }
        }
        latok = strtok(NULL, " ");
        LAcount += 1;
}
if(lacheck){
        textcounter+=4;
        continue;
}
```

Line 96부터 Memory에 Label이 붙은 주소 혹은 Data part의 값들을 집어 넣는다.

(line 176) LA의 경우 미리 처리가 필요하다. 하지만, LA보다 나중에 나오는 Label의 주소 대해서는 정확한 정보를 얻기 어렵다고 판단해, 이전의 Label의 경우 주소에 따라 명령 형식이 달라지도록(저장해 놓는 counter를 추가로 4 증가시켜 두 줄과 같은 효과)를 볼 수 있게 해 두었다.

```
//"instructions"라는 배열에 한 줄 한줄 입력
assmbly = fopen(filename, "r");
char line[64] = { 0 };
char instructions[512][64] = { 0 };
int inscount = 0;
while (fgets(line, 64, assmbly)) {
        if (strcmp(line,"\n")) {
                strcpy(instructions[inscount], line);
                inscount += 1;
        }
}
```

(line 80). file에서 한 줄씩 받아오고, 비어있는 줄은 받아오지 않는다.

Line 222부터 line 246까지는 순수한 instruction만 있는 배열을 만든다. Label이나 빈 줄을 지운 새 배열을 만들고, 그 배열을 Binary Instruction으로 하나하나 바꾼다.

Line 256부터 시작이고, 빈 줄이 나오면 멈추도록 되어 있다.

LA의 경우에는 여기에서도 다른 처리가 적용되었다.

```
if (!strcmp(OP, "la")) {
        int rt = atoi(strtok(NULL, " ") + 1);
        char* temp = strtok(NULL,"\n");
        unsigned long long int addr = Textmem.get_addr(temp);
        if(!addr){
                addr = Datamem.get_addr(temp);
        }

        int op = 0xf;
        unsigned int imm = addr >> 16;
        result = (op << 26) + (rt << 16) + imm;
        textsec[textsize] = result;
        PC+=4;
        textsize +=1;
        if (addr % 0x10000 != 0) {

                int op = 0xd;
                int rs = rt;
                unsigned int imm = addr % 0x10000;
        //      std::cout<<imm<<std::endl;
        //      printf("0x%lX\n", result);

                textsec[textsize] = (op << 26) + (rs << 21) + (rt << 16) + imm;
                textsize += 1;
                PC+=4;
        }
        continue;
}
```

기본적으로 lui instruction으로 변환하고, 주소값의 하위 16비트를 확인한 뒤 0이 아니면 ori instruction도 list에 넣어 준다.

```
char wfilename[128]={};
strcpy(wfilename,argv[1]);

char *ptr = strchr(wfilename,'.')+1;
*ptr='o';
FILE* file;
file = fopen(wfilename, "wb");
fprintf(file, "0x%X\n", textsize*4);
fprintf(file, "0x%llX\n", (dataCounter - 0x100000000));
for (int i = 0; i < textsize; i++) {
        fprintf(file, "0x%llX\n", textsec[i]);
}
//data section
unsigned long long int dataaddr = 0x100000000;
for (int i = 0; i < Datamem.Datacount; i++) {
        fprintf(file, "0x%llX\n",Datamem.get_data(dataaddr));
        dataaddr += 4;
}
```

line 446부터, file 이름을 바꾸어 한 줄씩 저장해 준다.