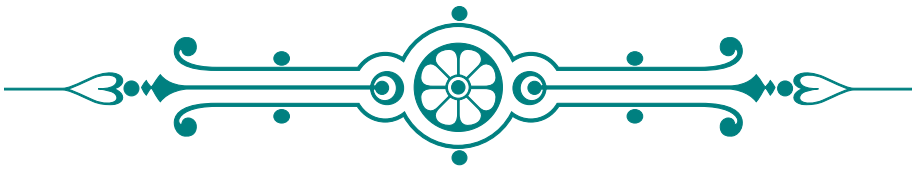

시스템 S/W 실습6
선택 과제

학번 : 2016110056

학과 : 불교학부

이름 : 박승원

날짜 : 2016년 11월 2일

2. SIC srcfile을 읽어서 각 줄을 LABEL, OPCODE, OPERAND 로 분리하여 Intfile에 출력하면서 SYMTAB[]을 생성한다.(실습 5의 PASS1 내용), 그리고 Intfile을 입력하고, SYMTAB[]의 내용을 사용하여 OPCODE가 기계명령이면 LOCCTR, CODE, ADDRESS를 ObjTmpfile에 출력하는 어셈블러 PASS2에 해당하는 프로그램을 구현하고 실습하시오. LOCCTR은 Location, Counter이고, CODE는 OPTAB[]에서 읽은 기계코드이고, ADDRESS는 OPERAND로서 SYMTAB[]에서 읽은 값이다.

단, C program compile 명령은 다음과 같고, 입력파일 srcfile은 각자 준비한다.

$g++ -o pass2 pass2.c

$./pass2 srcfile Intfile ObjTmpFile

Listing 1: 실행 파일의 소스, pass2.cpp

```
#include<iostream>
#include<iomanip>
#include<fstream>
#include<vector>
#include<map>
using namespace std;

vector<array<string, 3>> instructions;

std::map<std::string, unsigned char> op_table = {
    {"lda", 0x00}, {"ldx", 0x04}, {"sta", 0x0c}, {"stx", 0x10},
    {"add", 0x18}, {"sub", 0x1c}, {"mul", 0x20}, {"div", 0x24},
    {"comp", 0x28}, {"jeq", 0x30}, {"jgt", 0x34}, {"jlt", 0x38},
    {"jsub", 0x48}, {"rsub", 0x4c},
    {"rd", 0xd8}, {"wd", 0xdc}, {"td", 0xe0},
    {"ldch", 0x50}, {"stch", 0x54}, {"addx", 0x19}
};

bool is_opcode(string s)
{
    return op_table.find(s) != op_table.end();
}

int main(int argc, char** v)
{
    ifstream f(v[1]);
    ofstream of(v[2]);
```

```
ofstream lf("tmp");

// 라벨과 코드 오퍼란드 구분하는 부분
char c;
string com[3];
int n = 0;
bool sp_flag = false;
while(f >> noskipws >> c) {
    if(c != '\n') {
        if(c == '\t' || c == ' ') {
            if(!sp_flag) n++;
            sp_flag = true;
        } else {
            com[n] += c;
            sp_flag = false;
        }
    } else {
        instructions.push_back({com[0], com[1], com[2]});
        n = 0;
        for(int i=0; i<3; i++) com[i].clear();
        sp_flag = false;
    }
}
for(auto& a : instructions) {
    of << a[0] << ' ' << a[1] << ' ' << a[2] << endl;
}

//symtab 생성 부분
map<string, short> sym_table;

int addr = 0; //addr == LOCCTR
bool data_begin = false;
for(auto& a : instructions) {
    if(a[0] != "") {
        sym_table[a[0]] = addr;
    }
    if(a[1] == "start") {
        addr = stoi(a[2], nullptr, 16);
        sym_table["start"] = addr;
    } else if(a[1] == "end") sym_table["end"] = addr;
```

```
        else if(is_opcode(a[1])) {// opcode일 경우
            lf << hex << addr << ' ' << hex << +op_table[a[1]] << ' ' << hex << a[2] << endl;
            addr += 3;
        } else {
            if(! data_begin) {
                sym_table["data_begin"] = addr;
                data_begin = true;
            }
            if(a[1] == "word") addr += 3;
            else if(a[1] == "byte") addr++;
            else if(a[1] == "resb") addr += stoi(a[2]);
            else if(a[1] == "resw") addr += 3 * stoi(a[2]);
        }
    }// return LOCCTR- starting address = program size to load to memory
    lf.close();
    f.close();
    of.close();

    ifstream ft("tmp");
    ofstream ff(v[3]);
    string s[3];
    ff << "LCTR CD ADDR\n";
    while(ft >> s[0] >> s[1] >> s[2]) {
        ff << s[0] << ' ' << setw(2) << setfill('0') << s[1] << ' ';
        ff << hex << sym_table[s[2]] << endl;
    }
}
```

Listing 2: 임의의 srcfile

```
    start 1000
first lda seven
    sta alpha
    lda two
    add incr
    sta beta
    lda gamma
    sub two
    sta delta
    ldch charx
    stch cha
```

```
seven  word 7
two word 2
alpha  resw 1
beta  resw 1
gamma word 10
 delta  resw 1
 incr  word 3
cha  resb  1
charx  byte  78
      end  first
```

Listing 3: 소오스와 거의 차이가 없는 Intfile

```
  start  1000
 first   lda  seven
  sta  alpha
  lda  two
  add  incr
  sta  beta
  lda  gamma
  sub  two
  sta  delta
  ldch  charx
  stch  cha
seven  word 7
two word 2
alpha  resw 1
beta  resw 1
gamma word 10
 delta  resw 1
 incr  word 3
cha  resb  1
charx  byte  78
  end  first
```

실제로는 vector에 분리하여 넣었지만, 파일 상으로는 소오스와 거의 차이가 없다.

Listing 4: 결과로 나온 ObjTmpFile

```
LCTR CD ADDR
1000  00  101e
1003  0c  1024
1006  00  1021
```

```
1009  18  1030
100c  0c  1027
100f  00  102a
1012  1c  1021
1015  0c  102d
1018  50  1034
101b  54  1033
```

결과가 주소와 opcode, operand의 주소를 제대로 찾고 있음을 알 수 있다.

소감

기존에 냈던 소스들을 약간 수정하여 합쳤습니다.