**과목명: 시스템프로그래밍**

**분반: 1반(홀수)**

**<<Project #3>>**

**서강대학교 컴퓨터공학과**

**학번 : 20161641**

**이름 : 정성엽**

**목 차**

1. 프로그램 개요

2. 프로그램 설명

2.1 프로그램 흐름도

2.2 project3 상세 흐름도

3. 모듈 정의

3.1 loader.c 내부함수

3.1.1 int setProgaddr(int argc, char argv[100][100]);

3.1.2 void initEstab();

3.1.3 void storeEstab(int fn, char name[],int addr);

3.1.4 int returnEstab(char name[], int fileNum);

3.1.5 int Pass1(FILE \*fp[], int fileNumber);

3.1.6 int Pass2(FILE \*fp[], int fileNumber);

3.1.7 int loader(int argc, char argv[100][100]);

3.2 run.c 내부함수

3.2.1 int breakPoint(int argc, char argv[100][100]);

3.2.2 void printBp();

3.2.3 void store(int address,int value,int byte);

3.2.4 int returnValue(int addr,int byte);

3.2.5 void runOneInstruction();

3.2.6 void printRegisters();

3.2.7 int runProgram(int argc, char argv[100][100]);

4. 전역 변수 정의

4.1 구조체 정의

4.1.1 estabNode

4.2 에러관련 메크로 정의

4.2.1 #define DUPLICATED\_EXTERNAL\_SYMBOL (19)

4.2.2 #define UNDEFINED\_EXTERNAL\_SYMBOL (20)

4.2.3 #define PROGADDR\_OUT\_OF\_RANGE (22)

4.2.4 #define BP\_VALUE\_OUT\_OF\_RANGE (21)

4.3 loader.c, run.c 전역변수

4.3.1 int progaddr, execaddr, totalLen;

4.3.2 estabNode estab[5][100];

4.3.3 int esMax[5];

4.3.4 int progLen[5];

4.3.5 int reg[20];

4.3.6 int NewLoad = 0;

4.3.7 int bp[1000];

4.3.8 int bpMax = 0;

5. 코드 설명

5.1 실행 조건

5.2 loader.c 코드

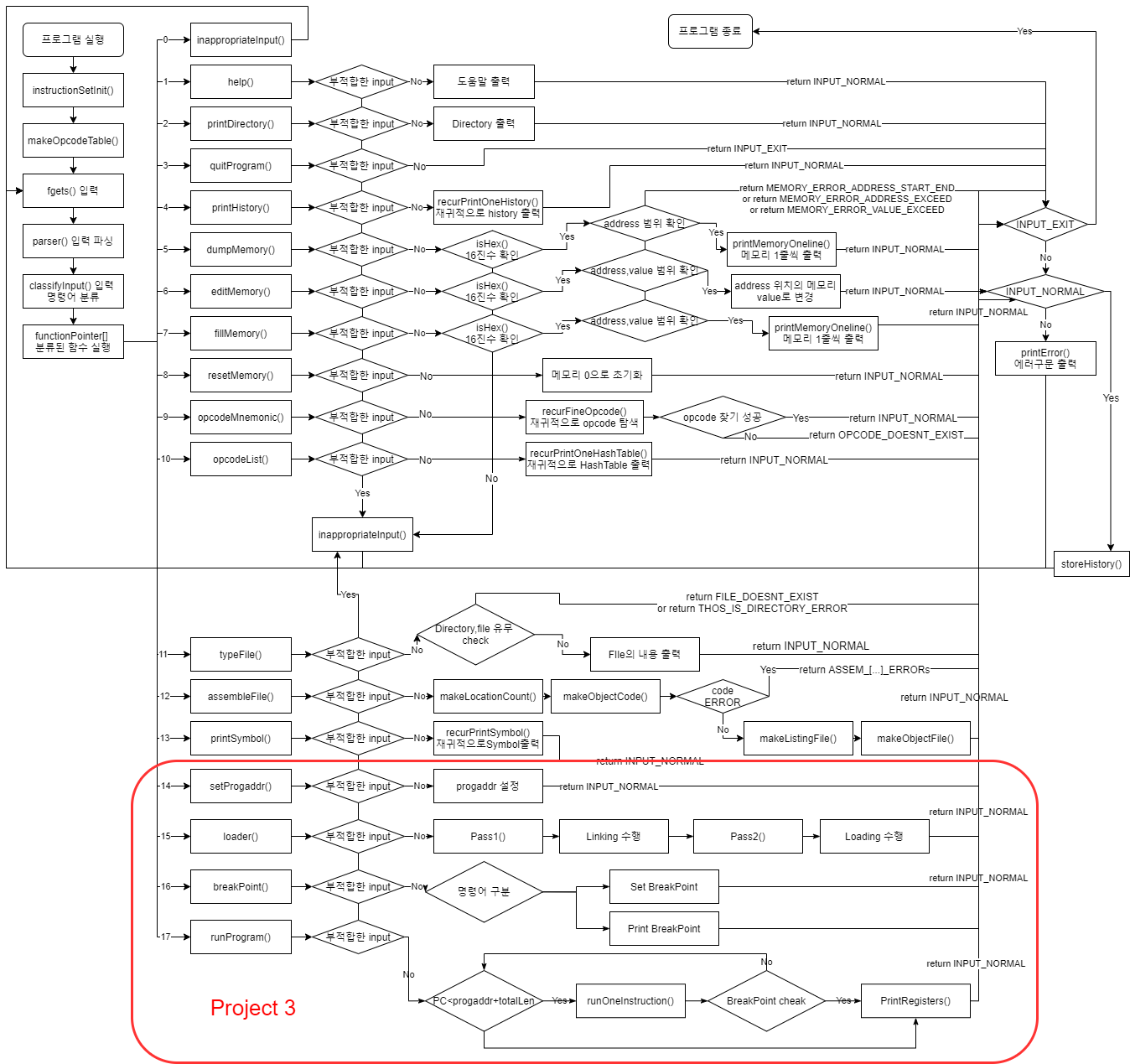
5.3 run.c 코드

1. 프로그램 개요

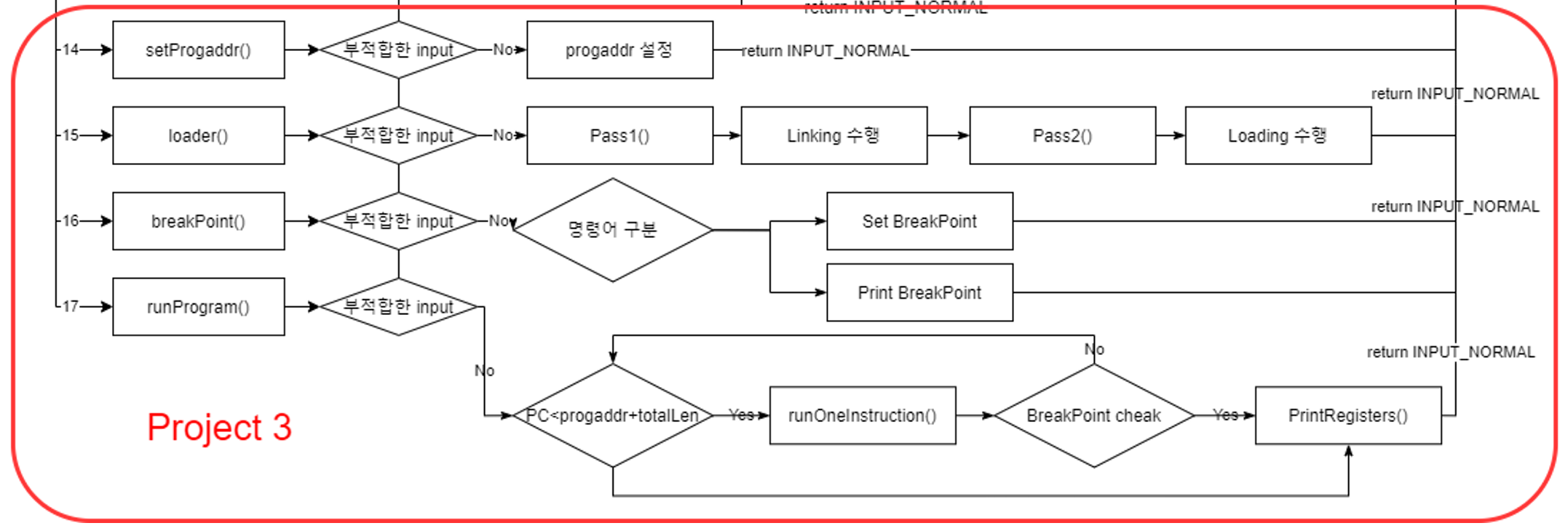
이 프로그램은 프로젝트1/2에서 구현한 셀(shell)에 Linking, Loading기능을 구현한 프로그램입니다. SIC/XE의 object source 파일을 입력 받아서 메모리에 loading하고, 이를 run 명령어를 사용해 직접 실행시켜 볼 수 있도록 합니다. 또한, Breakpoint와 같은 debug 도구를 제공합니다.

2. 프로그램 설명

2.1 프로그램 흐름도



2.2 project3 상세 흐름도



3. 모듈 정의

3.1 loader.c 내부함수

3.1.1 int setProgaddr(int argc, char argv[100][100]);

▶사용 변수

(1) int\* argv;

: 파싱된 명령어 인자들의 개수를 저장한다.

(2) char argc[100][100];

: 파싱된 명령어 인자들을 저장한다.

(3) int progaddr;

: 프로그램을 load할 위치를 저장한다.

▶함수의 동작 및 기능

: progaddr 명령어에 의해 입력받은 주소로 progaddr변수를 변경한다. 이를 통해 이후 obj 파일을 loading할 위치를 가리킨다.

3.1.2 void initEstab();

▶사용 변수

(1) estabNode estab[5][100];

: symbol 과 address를 저장할 배열

▶함수의 동작 및 기능

: obj 파일에서 읽은 symbol을 저장하기 위해 estab배열을 초기화 한다.

3.1.3 void storeEstab(int fn, char name[],int addr);

▶사용 변수

(1) int fn;

: 여러 개의 obj 파일 중 저장할 테이블의 번호를 알려준다.

(2) char name[];

: 저장할 symbol의 이름.

(3) int addr

: 저장할 symbol의 address.

▶함수의 동작 및 기능

: 입력으로 받은 symbol과 address를 estab에 저장한다.

3.1.4 int returnEstab(char name[], int fileNum);

▶사용 변수

(1) char name[];

: 탐색할 symbol의 이름을 저장한다.

(2) int fileNum

: 여러 개의 obj 파일 중 저장된 테이블의 번호를 알려준다.

▶함수의 동작 및 기능

: 입력으로 받은 name에 해당하는 symbol의 address를 estab에 return한다. 만약 없다면 -1을 return 한다.

3.1.5 int Pass1(FILE \*fp[], int fileNumber);

▶사용 변수

(1) FILE \*fp[];

: obj파일의 filepointer 가 저장된 배열

(2) int fileNumber;

: obj 파일의 개수

(3) int csaddr;

: control section의 address

(4) int cslth;

: 현재 obj 코드의 길이

(5) int addr;

: 각 object code의 address를 저장

(6) int fpIdx;

: 현재 fp[]배열에 저장된 obj파일의 filepointer 위치

(7) char str[300];

: fgets로 읽은 obj파일의 내용을 한 줄 저장하는 배열

(8) char addrStr[10];

: address가 integer로 변환되기 전 문자열을 저장하는 배열

(9) char name[10];

: symbol의 이름을 저장하는 배열

▶함수의 동작 및 기능

: fp[fpIdx]에 해당하는 obj 파일을 한 줄 씩 읽고, 만약 H레코드라면 프로그램의 길이, 시작 주소를 저장하고, D레코드라면 정의된 symbol을 확인하고 symbol name과 address를 estab에 저장한다. 자세한 동작구조는 아래의 pseudo code에 자세히 보여진다.

start

CSADDR = PROGADDR;

while (fpIdx<fileNumber) :

read next input (HEADER)

set CSLTH : control section length (in HEADER)

search ESTAB for control section length

if found :

return ERROR (DUPLICATED\_EXTERNAL\_SYMBOL)

else

store control section name into ESTAB with CSADDR

while type != 'E' :

read next input

if type == 'D' :

for each symbol in the record :

search ESTAB for symbol name

if found :

return ERROR (DUPLICATED\_EXTERNAL\_SYMBOL)

else :

store symbol into ESTAB with value (CSADDR + indicated address)

CSADDR += CSLTH;

fpIdx++;

end

3.1.6 int Pass2(FILE \*fp[], int fileNumber);

(1) FILE \*fp[];

: obj파일의 filepointer 가 저장된 배열

(2) int fileNumber;

: obj 파일의 개수

(3) int csaddr;

: control section의 address

(4) int cslth;

: 현재 obj 코드의 길이

(5) int fpIdx;

: 현재 fp[]배열에 저장된 obj파일의 filepointer 위치

(6) char str[300];

: fgets로 읽은 obj파일의 내용을 한 줄 저장하는 배열

(7) char temp[10];

: address나 value등 여러 경우에 다용도로 문자열을 저장하는 배열

(8) char refDic[100][10];

: M레코드로 modification을 수행할 symbol들을 저장하는 배열

▶함수의 동작 및 기능

: fp[fpIdx]에 해당하는 obj 파일을 한 줄 씩 읽고, 만약 R레코드라면 이후 modification할 symbol들을 저장하고, T레코드라면 위치와 길이를 확인하여 object code를 memory에 저장한다. M레코드라면 refDic[idx]에 이름에 해당하는 symbol을 estab에서 찾는다. 만약 없다면 error를 return하고, 있다면 해당 값을 address위치의 메모리에 +,- 연산을 취해 저장한다. 자세한 동작구조는 아래의 pseudo code에 자세히 보여진다.

start

CSADDR = PROGADDR;

while (fpIdx<fileNumber) :

read next input (HEADER)

set CSLTH : control section length (in HEADER)

while type != 'E' :

read next input

if type == 'R' :

make r record dictionmaty;

...

if type == 'T' :

(if object code is in character form, convert into internal representation)

move object code from record to location

(CSADDR + specified address)

else if type == 'M' :

search ESTAB for modifying symbol name

if found :

add or subtract symbol vlaue at location

(CSADDR + specified address)

else :

return ERROR (UNDEFINED\_EXTERNAL\_SYMBOL)

if E[address] is specified :

set EXECADDR to (CSADDR + specified address)

CSADDR += CSLTH;

fpIdx++;

end

3.1.7 int loader(int argc, char argv[100][100]);

▶사용 변수

(1) int\* argv;

: 파싱된 명령어 인자들의 개수를 저장한다.

(2) char argc[100][100];

: 파싱된 명령어 인자들을 저장한다.

(3) int fileNumber;

: obj 파일의 개수

(4) int ret;

: Pass1(), Pass2() 함수가 반환하는 값을 저장한다.

▶함수의 동작 및 기능

: fopen 함수를 사용해 obj파일들을 열고, 해당하는 file pointer를 fp배열에 저장한다. 이후, Pass1()함수, Pass2()함수들을 호출하여 linking, loading과정을 수행한다. 수행이 성공적으로 마치면, load map을 출력하고 새롭게 load가 되었음을 run함수에 알려주기 위해 NewLoad flag를 1로 저장한다.

3.2 run.c 내부함수

3.2.1 int breakPoint(int argc, char argv[100][100]);

▶사용 변수

(1) int\* argv;

: 파싱된 명령어 인자들의 개수를 저장한다.

(2) char argc[100][100];

: 파싱된 명령어 인자들을 저장한다.

▶함수의 동작 및 기능

: 입력된 address값으로 break point를 설정한다. bp 만 입력되는 경우 printBp()함수를 호출하여 저장된 bp들을 모두 출력한다. bp clear가 입력되는 경우 bp테이블을 초기화 한다.

3.2.2 void printBp();

▶사용 변수

▶함수의 동작 및 기능

: 저장된 break point들을 모두 출력한다.

3.2.3 void store(int address,int value,int byte);

▶사용 변수

(1) int address;

: 저장할 memory의 위치

(2) int value;

: 메모리에 저장할 값

(3) int byte;

: 메모리에 저장할 byte길이

▶함수의 동작 및 기능

: 메모리에 address위치에 byte길이만큼 value를 저장한다.

3.2.4 int returnValue(int addr,int byte);

▶사용 변수

(1) int address;

: 저장할 memory의 위치

(2) int byte;

: 메모리에서 읽을 byte길이

▶함수의 동작 및 기능

: 메모리의 address위치에서 byte길이만큼 저장된 값을 읽는다.

3.2.5 void runOneInstruction();

▶사용 변수

(1) int opcode;

: object code의 opcode

(2) int format;

: object code의 format

▶함수의 동작 및 기능

: 현재 PC의 기준으로 memory의 object code를 1개 읽고, 해당하는 format을 구분한다. 이후 해당 명령어를 수행한다. 명령어 수행 후, 명령어 길이만큼의 PC를 증가시킨다.

3.2.6 void printRegisters();

▶사용 변수

▶함수의 동작 및 기능

: 각 register의 값을 출력형식에 맞추어 출력한다.

3.2.7 int runProgram(int argc, char argv[100][100]);

▶사용 변수

(1) int\* argv;

: 파싱된 명령어 인자들의 개수를 저장한다.

(2) char argc[100][100];

: 파싱된 명령어 인자들을 저장한다.

▶함수의 동작 및 기능

: 현재 시작위치에 맞게 PC가 progaddr+totalLen보다 작은 동안 memory에 저장된 명령어를 읽어 수행한다. 수행 중 breakPoint를 만나게 되면 레지스터의 상태를 출력하고 종료한다. 이때 다시 run명령어를 통해 함수가 호출된다면 종료 지점부터 다시 수행한다. 프로그램의 끝까지 수행한후 레지스터 상태를 출력하고 함수를 종료한다.

4. 전역 변수 정의

4.1 구조체 정의

4.1.1 estabNode

▶사용 변수

(1) char name[10];

: symbol의 이름을 저장한다.

(2) int addr;

: symbol의 주소를 저장한다.

▶역할

: modification을 위해 symbol의 이름과 주소를 저장하는 배열의 노드

4.2 에러관련 메크로 정의

4.2.1 #define DUPLICATED\_EXTERNAL\_SYMBOL (19)

▶역할

: symbol이 중복되는 경우를 error check 한다.

4.2.2 #define UNDEFINED\_EXTERNAL\_SYMBOL (20)

▶역할

: symbol이 정의되지 않은 경우를 error check 한다.

4.2.3 #define PROGADDR\_OUT\_OF\_RANGE (22)

▶역할

: progaddr 명령어의 address가 범위를 벗어나는 경우 error check한다.

4.2.4 #define BP\_VALUE\_OUT\_OF\_RANGE (21)

▶역할

: break point의 지정 address가 범위를 벗어나는 경우 error check한다.

4.3 loader.c, run.c 전역변수

4.3.1 int progaddr, execaddr, totalLen;

▶역할

: progaddr은 progaddr 명령어로 지정된 프로그램을 load할 위치를 저장한다. execaddr은 load 후 프로그램의 실행위치를 저장한다. totalLen은 load된 프로그램의 전체 길이를 저장한다.

4.3.2 estabNode estab[5][100];

▶역할

: modification을 위해 symbol의 이름과 주소를 저장하는 배열

4.3.3 int esMax[5];

▶역할

: estab 배열에 저장된 symbol의 개수를 저장한다.

4.3.4 int progLen[5];

▶역할

: 각각의 load된 프로그램의 길이를 저장한다.

4.3.5 int reg[20];

▶역할

: (0:A, 1:X, 2:L, 3:B, 4:S, 5:T, 6:F, 7:F2, 8:PC, 9:SW)으로 각각의 레지스터 값을 저장한다.

4.3.6 int NewLoad = 0;

▶역할

: 프로그램이 새롭게 load된 경우만 레지스터 값을 초기화 하기 위해 load된 여부를 flag로 저장한다.

4.3.7 int bp[1000];

▶역할

: break point를 저장하는 배열

4.3.8 int bpMax = 0;

▶역할

: break point의 개수를 저장

5. 코드 설명

5.1 실행 조건

: 해당 코드는 명세서의 조건에 따라 load할 obj 코드에 error가 없다고 상정하고 제작되었다. 또한 run 명령어는 progaddr 0의 상태로 copy.obj 코드만을 실행한다고 상정하였기 때문에 obj코드가 오류가 있거나 실행조건이 다르다면 오류가 발생할 수 있다.

5.2 loader.c 코드

#include "20161641.h"

extern unsigned char memory[65536][16];

extern int NewLoad; // 프로그램이 새롭게 Load되었는지 알려주는 Flag

int progaddr;

int execaddr;

estabNode estab[5][100];

int progLen[5];

int totalLen;

int esMax[5]; // estab에 저장된 symbol의 개수

int setProgaddr(int argc, char argv[100][100]){ // loader 또는 run 명령어를 수행할 때 시작하는 주소를 지정한다

if(argc != 2)

return INPUT\_ERROR;

progaddr = strtol(argv[1],NULL,16);

if(progaddr<0 || progaddr>0xFFFFF){

progaddr = 0;

return PROGADDR\_OUT\_OF\_RANGE;

}

return INPUT\_NORMAL;

}

void initEstab(){ // estab 초기화

for(int i = 0;i<5;i++){

for(int j = 0;j<100;j++){

estab[i][j].addr = 0;

estab[i][j].name[0] = '\0';

}

esMax[i] = 0;

}

}

void storeEstab(int fn, char name[],int addr){ // estab 저장

for(int i = 6;i>0;i--)

if(name[i] == ' ' || name[i] == '\n')

name[i] = '\0';

estab[fn][esMax[fn]].addr = addr;

strcpy(estab[fn][esMax[fn]].name,name);

esMax[fn]++;

}

int returnEstab(char name[], int fileNum){ // estab symbol에 해당하는 address 반환

for(int i = 0;i<fileNum;i++)

for(int j = 0;j<esMax[i];j++)

if(strcmp(name,estab[i][j].name) == 0)

return estab[i][j].addr;

return -1;

}

int Pass1(FILE \*fp[], int fileNumber){ // Pass1 수행, ESTAB 생성

int csaddr = progaddr;

int cslth,addr;

int fpIdx = 0;

char str[300];

char addrStr[10];

char name[10];

while(fpIdx<fileNumber){

fgets(str,300,fp[fpIdx]);

strncpy(name, str+1,6);

name[6] = '\0';

cslth = strtol(str+13,NULL,16);

if(returnEstab(name,fileNumber) != -1){ // symbol duplicated

printf("%s :: ",name);

return DUPLICATED\_EXTERNAL\_SYMBOL;

}

storeEstab(fpIdx,name,csaddr);

while(fgets(str,300,fp[fpIdx]) != NULL){

if(str[0] != 'D')

continue;

for(int i = 1;i<strlen(str)-1;i+=12){ // fgets 에서 '\n' 까지 저장함

strncpy(name,str+i,6);

strncpy(addrStr,str+6+i,6);

name[6] = addrStr[6] = '\0';

addr=strtol(addrStr,NULL,16);

if(returnEstab(name,fileNumber) != -1){ // symbol duplicted

printf("%s :: ",name);

return DUPLICATED\_EXTERNAL\_SYMBOL;

}

storeEstab(fpIdx,name,csaddr+addr);

}

}

csaddr += cslth;

progLen[fpIdx] = cslth;

fpIdx++;

}

return INPUT\_NORMAL;

}

int Pass2(FILE \*fp[], int fileNumber){ // Pass2 수행, Linking Loading 수행

int csaddr = progaddr;

int cslth;

int fpidx = 0;

char str[300];

char temp[10];

char refDic[100][10] = {0,};

while(fpidx<fileNumber){

fgets(str,300,fp[fpidx]);

cslth = strtol(str+13,NULL,16);

while(fgets(str,300,fp[fpidx]) != NULL){

if(str[0] == 'R'){ // R record

int refIdx;

strcpy(refDic[1],estab[fpidx][0].name); // 01은 프로그램 자체를 가리킴

for(int i = 1;i<strlen(str)-1;i+=8){

strncpy(temp,str+i,2);

temp[2] = '\0';

refIdx = atoi(temp);

strncpy(temp,str+2+i,6);

temp[6] = '\0';

for(int j = 0;j<6;j++){

if(temp[j] == ' ' || temp[j] == '\n')

temp[j] = '\0';

}

strcpy(refDic[refIdx],temp);

}

}

else if(str[0] == 'T'){ // T record

char addrStr[7];

char lenStr[3];

strncpy(addrStr,str+1,6);

strncpy(lenStr,str+7,2);

addrStr[6] = lenStr[2] = '\0';

int Taddr = strtol(addrStr,NULL,16) + csaddr;

int Tlen = strtol(lenStr,NULL,16);

for(int i = 0;i<Tlen;i++){

strncpy(temp,str+9+2\*i,2);

temp[2] = '\0';

memory[Taddr/16][Taddr%16] = strtol(temp,NULL,16);

Taddr++;

}

}

else if(str[0] == 'M'){ // M record

// modify 하는 길이가 05, 06인지 구분할 필요 X

char addrStr[7];

strncpy(addrStr,str+1,6);

addrStr[6] = '\0';

int Maddr = strtol(addrStr,NULL,16) + csaddr;

for(int i = 0;i<3;i++)

sprintf(temp+i\*2,"%02X",memory[(Maddr+i)/16][(Maddr+i)%16]);

temp[6] = '\0';

int objValue;

objValue = strtol(temp,NULL,16);

if(!('0' <= temp[0] && temp[0] < '8')) // 음수인 경우 2의 보수

objValue = -(0xFFFFFF - objValue +1);

strncpy(temp,str+10,2);

temp[2] = '\0';

int refIdx = atoi(temp);

int refAddr = returnEstab(refDic[refIdx],fileNumber);

if(refAddr == -1){ // 정의되지 않은 reference 값인 경우

printf("%s :: ",refDic[refIdx]);

return UNDEFINED\_EXTERNAL\_SYMBOL;

}

if(str[9] == '+')

objValue += refAddr;

else if(str[9] == '-')

objValue -= refAddr;

if(objValue < 0) // 음수인 경우 2의 보수

objValue -= 0xFF000000;

sprintf(temp,"%06X",objValue);

char shortStr[3];

for(int i = 0;i<3;i++){

strncpy(shortStr,temp+i\*2,2);

shortStr[2] = '\0';

memory[(Maddr+i)/16][(Maddr+i)%16] = strtol(shortStr,NULL,16);

}

}

}

if(str[0] == 'E'){ // E record

if(str[1] != '\n'){ // execaddr이 있는 경우

strncpy(temp,str+1,6);

temp[6] = '\0';

execaddr = csaddr+strtol(temp,NULL,16);

}

}

csaddr += cslth;

fpidx++;

}

return INPUT\_NORMAL;

}

int loader(int argc, char argv[100][100]){ // .obj 파일을 읽어서 linking/loading작업을 수행후, memory에 저장한다

if(argc<2)

return INPUT\_ERROR;

initEstab();

FILE\* fp[5];

for(int i = 1;i<argc;i++){ // open the .obj files

fp[i-1] = fopen(argv[i],"r");

if(fp[i-1] == NULL)

return FILE\_DOESNT\_EXIST;

}

int fileNumber = argc-1;

int ret = Pass1(fp,fileNumber);

if(ret != INPUT\_NORMAL)

return ret;

for(int i = 1;i<argc;i++){ // open the .obj files

fclose(fp[i-1]);

fp[i-1] = fopen(argv[i],"r");

if(fp[i-1] == NULL)

return FILE\_DOESNT\_EXIST;

}

ret = Pass2(fp,fileNumber);

if(ret != INPUT\_NORMAL)

return ret;

totalLen = 0;

printf("control symbol address length\n"); // 출력

printf("section name\n");

printf("--------------------------------\n");

for(int i = 0;i<fileNumber;i++){

for(int j = 0;j<esMax[i];j++){

if(j == 0)

printf("%-7s%14.04X %04X\n",estab[i][j].name,estab[i][j].addr,progLen[i]);

else

printf("%14s %04X\n",estab[i][j].name,estab[i][j].addr);

}

totalLen += progLen[i];

}

printf("--------------------------------\n");

printf(" total length %04X\n",totalLen);

NewLoad = 1;

return INPUT\_NORMAL;

}

5.3 run.c 코드

#include "20161641.h"

extern int execaddr;

extern int progaddr;

extern int totalLen;

extern unsigned char memory[65536][16];

int reg[20]; // Registers

typedef enum {A, X, L, B, S, T, F, FF, PC, SW}regNum;

// 0:A, 1:X, 2:L, 3:B, 4:S, 5:T, 6:F, 7:F2, 8:PC, 9:SW

int NewLoad = 0; // 프로그램이 새롭게 Load되었는지 알려주는 Flag

int bp[1000];

int bpMax = 0;

int breakPoint(int argc, char argv[100][100]){ // break point를 설정한다

if(argc > 2)

return INPUT\_ERROR;

if(argc == 1)

printBp();

else if(strcmp(argv[1],"clear") == 0){ // bp 초기화

bpMax = 0;

printf("\t[");

printf("%c[1;32m",27); // ok 초록색으로 출력

printf("ok");

printf("%c[0m",27);

printf("] clear all breakpoints\n");

}

else{

if(isHex(argv[1]) == 0)

return INPUT\_ERROR;

int tmp = strtol(argv[1],NULL,16);

if(tmp<0 || tmp >totalLen)

return BP\_VALUE\_OUT\_OF\_RANGE;

bp[bpMax++] = tmp;

printf("\t[");

printf("%c[1;32m",27); // ok 초록색으로 출력

printf("ok");

printf("%c[0m",27);

printf("] creat breakpoint %s\n",argv[1]);

}

return INPUT\_NORMAL;

}

void printBp(){ // 저장된 BreakPoint를 출력한다

printf(" breakpoint\n");

printf(" ----------\n");

for(int i = 0;i<bpMax;i++)

printf(" %X\n",bp[i]);

}

void store(int address,int value,int byte){ // 메모리에 byte 길이의 값을 저장한다

char temp[10], shortStr[3];

if(value<0)

value -= 0xFF000000;

sprintf(temp,"%06X",value);

for(int i = 0;i<byte;i++){

strncpy(shortStr,temp+i\*2,2);

shortStr[2] = '\0';

memory[(address+i)/16][(address+i)%16] = strtol(shortStr,NULL,16);

}

}

int returnValue(int addr,int byte){ // 메모리에 저장된 byte 만큼의 값을 반환한다

int temp = memory[addr/16][addr%16];

for(int i = 1;i<byte;i++){

temp \*= 16\*16;

temp += memory[(addr+i)/16][(addr+i)%16];

}

return temp;

}

void runOneInstruction(){ // PC기준 1개의 명령어 실행

// copy.obj 파일의 명령어만 구현되어있음 (ERROR 처리도 미구현)

int opcode = memory[reg[PC]/16][reg[PC]%16];

int format = 0;

switch(opcode/0x10){ //format별로 나누어주기

case 0x9:

case 0xA:

case 0xB:

format = 2;

break;

case 0xC:

case 0xF:

format = 1;

break;

default:

format = 3;

if(memory[(reg[PC]+1)/16][(reg[PC]+1)%16]&(1<<4))

format = 4;

}

if(format == 1 || opcode == 0){ //format1 미구현, 빈 memory인 경우 예외처리

reg[PC]++;

return;

}

else if(format == 2){

int reg1 = memory[(reg[PC]+1)/16][(reg[PC]+1)%16];

int reg2 = reg1%16;

reg1 /= 16;

switch(opcode){

case 0xB8: // TIXR

reg[X]++;

if(reg[X] == reg[reg1])

reg[SW] = '=';

else if (reg[X]<reg[reg1])

reg[SW] = '<';

else

reg[SW] = '>';

break;

case 0xA0: // COMPR

if(reg[reg1] == reg[reg2])

reg[SW] = '=';

else if (reg[reg1]<reg[reg2])

reg[SW] = '<';

else

reg[SW] = '>';

break;

case 0xB4: // CLEAR

reg[reg1] = 0;

break;

}

reg[PC]+=2;

}

else{ // format 3 and 4

int ni = opcode%4;

int addr = memory[(reg[PC]+1)/16][(reg[PC]+1)%16];

int xbpe = addr/16;

addr %= 16;

addr \*= 16\*16;

addr += memory[(reg[PC]+2)/16][(reg[PC]+2)%16];

if(format == 3 && (xbpe & (1<<1)) && addr>(1<<11)) // 2의 보수

addr -= (1<<12);

reg[PC] += 3;

if(format == 4){

addr \*=16\*16;

addr += memory[reg[PC]/16][reg[PC]%16];

reg[PC]++;

}

if(xbpe & (1<<3))

addr += reg[X];

if(xbpe & (1<<2))

addr += reg[B];

if(xbpe & (1<<1))

addr += reg[PC];

if(ni == 2)

addr = returnValue(addr,3);

switch((opcode/4)\*4){

case 0x14: // STL

store(addr,reg[L],3);

break;

case 0x68: // LDB

if(ni == 1)

reg[B] = addr;

else

reg[B] = returnValue(addr,3);

break;

case 0x48: // +JSUB

reg[L] = reg[PC];

reg[PC] = addr;

break;

case 0x00: // LDA

if(ni == 1)

reg[A] = addr;

else

reg[A] = returnValue(addr,3);

break;

case 0x28: // COMP

if(ni != 1)

addr = returnValue(addr,3);

if(reg[A] == addr)

reg[SW] = '=';

else if (reg[A]<addr)

reg[SW] = '<';

else

reg[SW] = '>';

break;

case 0x30: // JEQ

if(reg[SW] == '=')

reg[PC] = addr;

break;

case 0x3C: // J

reg[PC] = addr;

break;

case 0x0C: // STA

store(addr,reg[A],3);

break;

case 0x74: // LDT

if(ni != 1)

addr = returnValue(addr,3);

reg[T] = addr;

break;

case 0xE0: // TD

reg[SW] = '<';

break;

case 0xD8: // RD

reg[A] = 0;

break;

case 0x54: // STCH

store(addr,reg[A]%16,1);

break;

case 0x38: // JLT

if(reg[SW] == '<')

reg[PC] = addr;

break;

case 0x10: // STX

store(addr,reg[X],3);

break;

case 0x4C: // RSUB

reg[PC] = reg[L];

break;

case 0x50: // LDCH

if(ni != 1)

addr = returnValue(addr,3);

reg[A] = (reg[A]&0xFFFFFF00) + (addr/0x10000);

break;

case 0xDC: // WD

//

break;

}

}

}

void printRegisters(){ // Print Registers

printf(" A : %06X X : %06X\n", reg[A], reg[X]);

printf(" L : %06X PC : %06X\n", reg[L], reg[PC]);

printf(" B : %06X S : %06X\n", reg[B], reg[S]);

printf(" T : %06X\n", reg[T]);

}

int runProgram(int argc, char argv[100][100]){ // memory에 load된 프로그램을 실행한다

if(argc > 1)

return INPUT\_ERROR;

if(NewLoad){ // 새롭게 loading 한 경우만 초기화 수행

NewLoad = 0;

for(int i = 0;i<10;i++)

reg[i] = 0;

reg[PC] = execaddr; // 프로그램의 시작주소로 초기화

reg[L] = totalLen; // 프로그램의 길이로 초기화

}

while(reg[PC]<progaddr+totalLen){ // while문으로 runOneInstruction() 실행하면서

runOneInstruction();

for(int i = 0;i<bpMax;i++){

if(reg[PC] == bp[i]){ // bp를 만나면 함수 종료

printRegisters();

printf(" Stop at checkpoint[%X]\n",bp[i]);

return INPUT\_NORMAL;

}

}

}

printRegisters();

printf(" End Program\n");

return INPUT\_NORMAL;

}