**과목명: 시스템프로그래밍**

**분반**

**<<Project #3>>**

**서강대학교 [학부명]**

**20161580[학번]**

**김현규[이름]**

목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. 프로그램 흐름도
3. **모듈 정의**
   1. 정의한 모듈에 대해 간략적인 설명
4. **전역 변수 정의**
5. **코드 설명**
6. 프로그램 개요

|  |
| --- |
| -프로젝트 1, 2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가하는 프로그램이다.  -프로젝트 2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모 리에 올리는 일을 수행한다.  -주소 지정 명령어, Linking Loader, 프로그램 실행 명령어, debug 명령어 등을 구현해야 하는데 이를 위해 필요한 자료구조 및 알고리즘을 구상하여 전체 프로그램을 설계한다. |

1. 프로그램 설명

|  |
| --- |
| ● 구현해야 할 사항들  ① 주소 지정 명령어 ( progaddr )  - loader 또는 run 명령어를 수행할 때 시작하는 주소를 지정한다.  - sicsim이 시작되면 default로 progaddr는 0x00 주소로 지정한다.  ② Linking Loader ( loader )  - filename1, filename2, … 에 해당하는 object 파일을 읽어서 linking 작업을 수행 후, 가상 메모리(1M)에 그 결과를 기록한다.  - 교재에 나오는 Pass1과 Pass2를 수행한다.  - Pass1      - Pass2    - 파일 개수는 3개까지만 고려한다.  - Loader 실행이 후, load map을 화면에 출력합니다  ③ 프로그램 실행 ( run )  - loader 명령어의 수행으로 메모리에 load된 프로그램을 실행한다.  - progaddr 명령어로 지정한 주소부터 실행된다.  - 실행 결과로써 register 상태를 화면에 출력한다. 출력되는 register는 A, X, L, PC, B, S, T 이다.  - 모든 레지스터는 프로그램이 load되면 0으로 초기화 된다. 단, PC는 프로그램 의 시작 주소, L은 프로그램의 길이로 초기화 한다.  - Breakpoint까지 실행되고 Breakpoint가 없으면 프로그램 끝까지 실행한다.  - 수행하는 프로그램은 제공되는 copy.obj를 사용하며 다른 경우는 고려하지 않는다.  ④ debug 명령어 ( bp )  - sicsim에 breakpoint를 지정한다.  - breakpoint는 현재 프로그램의 Loc 기준으로 한다.  - Breakpoint는 bp 명령어를 통해서 프로그램 길이만큼 지정할 수 있다.  - run을 수행하면 breakpoint까지 프로그램이 실행되고 프로그램이 정지한다.  - 다음 run의 실행은 정지된 breakpoint부터 시작, 그 다음 breakpoint까지 진행된다.  - Breakpoint가 없으면 프로그램 끝까지 실행한다. |

2-1. 프로그램 흐름도

|  |
| --- |
|  |

1. 모듈 정의
   1. 모듈 이름: void run(void)
      1. 기능

-load되어 있는 memory를 실행한다.

-PC가 Breakingpoint, 혹은 ENDADDR에 도달하기 전까지 프로그램을 실행한다.

-data[k](k = opcode\_format)에 실행할 object code을 읽어온다.

-data[k]를 해석하여 object code의

1. opcode(char opcode\_str에 저장) 2. n, I, x, b, p, e 3. 2.의 정보로 relativeadr

을 알아낸다.

-object code를 실행한다. 즉 레지스터의 값을 변화시켜준다.

* 1. 모듈 이름: void addbp(char string[])
     1. 기능

char Breakingpoint[Bpcount(Breaking point의 개수)][]에 해당 breaking point의 정보(char string[])를 추가한다.

* 1. 모듈 이름: void printbp(void)
     1. 기능

char Breakingpoint[Bpcount][]의 정보를 출력한다.

* 1. void clearbp(void)
     1. 기능

-char Breakingpoint[Bpcount][]의 정보를 초기화한다.

-Bpcount을 0으로 초기화한다.

* 1. void progaddr(char address[])
     1. 기능

-PROGADDR의 정보를 초기화한다.

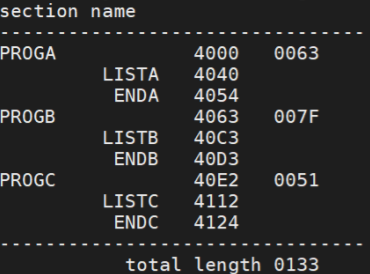
* 1. void loader(int parametercount, char file[][20])
     1. 기능

-paramtercount는 load해야 하는 file의 개수를 나타낸다.

PASS1

-\*.obj 파일을 읽는다.

-파일의 정보를 읽는다. 읽어야 하는 정보는 다음과 같다.



-PROGA: EXTAB\_CSN[0](Control Section Name)

-4000: EXTAB\_CSN\_ADR[0](Control Section Name’s Address)

-0063: EXTAB\_CSN\_LEN[0](Control Section Name’s Length)

-LISTA: EXTAB\_SYM[0][0](External Symbol’s name) -ENDA: EXTAB\_SYM[0][1]

-4040: EXTAB\_ADR[0][0](Extarnal Symbol’s Address) -4054: EXTAB\_ADR[0][1]

-PROGB: EXTAB\_CSN[1]

-4063: EXTAB\_CSN\_ADR[1]

-007F: EXTAB\_CSN\_LEN[1]

… (이하 동일)

-읽은 정보는 출력한다.

PASS2

-\*.obj 파일을 읽는다.

-Text Record: 해당 내용을 memory에 load한다.

-Modification Record: memory를 Record의 정보에 맞게 수정한다.

int Operand: 기존 memory에 있는 값

int Operator: 기존 memory에 추가해줘야 하는 값

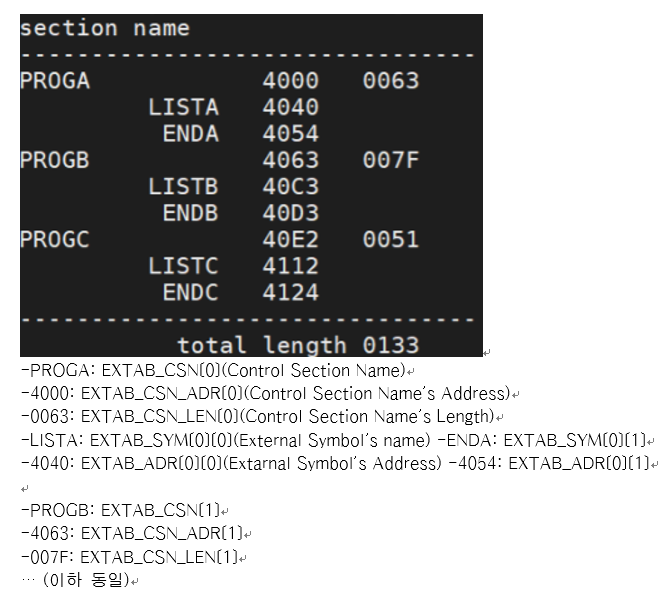
int Replace: 수정되어야 하는 값. ‘+’인 경우 Replace = Operand + Operator, ‘-‘인 경우 Replace = Operand – Operator

4 전역 변수 정의

4.1 이름: char EXTAB\_CSN[3][10], char EXTAB\_CSN\_ADR[3][10], char EXTAB\_CSN\_LEN[3][10], char EXTAB\_SYM[3][10][10], char EXTAB\_ADR[3][10][10]

4.1.1 기능:

EXTERNAL TABLE을 구성하는 값.



4.2 char Breakingpoint[10][10]

4.2.1 기능: Breakingpoint의 위치를 저장하는 자료구조

4.3 int Bpcount = 0

4.3.1 기능: Breakingpoint의 개수를 저장하는 값

4.4 int PROGADDR = 0

4.4.1 기능: Program이 load될 주소를 저장하는 값

4.5 int totallength = 0

4.5.1 기능: load된 Program의 전체 길이를 저장하는 값

4.6 int A = 0, L = 0, B = 0, T = 0, X = 0, PC = 0, S = 0

4.6.1 기능: run하는 Program의 Register의 정보를 저장하는 값

4.7 int runEXECADDR

4.7.1 기능: run할 시작 주소를 저장.

4.8 int actingflag

4.8.1 기능: 이미 이전에 해당 프로그램이 run한 적이 있으면 flag = 1

4.9 int CC = 0

* + 1. 기능: run하는 중 일부 Opcode에 한하여 이용되는 CC의 정보를 저장하는 값

1. 코드 설명

-설명은 주석으로 상세히 기입하였다.

void run(void){

//ENDADDR: Load한 Program의 끝 주소

int ENDADDR = PROGADDR + totallength;

//breaking point

int bp;

//actingflag: 이전에 이미 run을 실행했을 경우

//runEXECADDR: actingflag가 1일 경우, 이번 프로그램이 실행되어야할 주소

if(actingflag) PC = runEXECADDR;

else{ PC = PROGADDR; L = ENDADDR; }

int check = 0;

int break\_flag = 0;

while(PC != ENDADDR){

int y = PC / line\_size; int x = PC % line\_size;

int left[5]; int right[5]; int data[5];

//data[k]: PC 기준 k번째 memory가 가지고 있는 data

//left[k]: data[k]에서 왼쪽에 있는 data

//right[k]: data[k]에서 오른쪽에 있는 data

//ex)data[k] = 17일 경우, left[k] = 1 right[k] = 7

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

data[k] = mem[y][x];

left[k] = data[k] / 16;

right[k] = data[k] % 16;

x++;

if(x >= line\_size) { y++; x = 0;}

}

//opcode

//opcode\_inf: opcode의 정보를 저장

int opcode\_inf = data[0]; char opcode\_str[5];

while(opcode\_inf % 4 != 0) opcode\_inf--;

sprintf(opcode\_str, "%02X", opcode\_inf);

//n, i, x(xx로 저장함), b, p, e

int n,i,xx,b,p,e;

n = (right[0] / 2) % 2;

i = right[0] % 2;

//주소 지정 방식

int simpleadr = 0; int immediateadr = 0; int indirectadr = 0;

if(n == 1 && i == 1) simpleadr = 1;

else if(n == 1 && i == 0) indirectadr = 1;

else if(n == 0 && i == 1) immediateadr = 1;

else { }

int pcrelative = 0; int baserelative = 0; int basexrelative = 0; int norelative = 0;

//e,p,b,x

e = left[1] % 2;

p = (left[1] / 2) % 2;

b = (left[1] / 2 / 2) % 2;

xx = left[1] / 2 / 2 / 2;

//format에 따라 PC count

//CLEAR, COMPR, TIXR(format = 2)

if(!strcmp(opcode\_str, "B4") || !strcmp(opcode\_str, "A0") || !strcmp(opcode\_str, "B8")) {PC += 2;}

else{

//format = 4

if(e == 1){

PC += 4;

if(x >= line\_size) data[3] = mem[y+1][0];

else data[3] = mem[y][x];

left[3] = data[3] / 16; right[3] = data[3] % 16;

}

//format = 3

else {PC += 3;}

}

//상대 주소 지정 방법 선택

if(p == 1) pcrelative = 1;

else if(b == 1){

if(xx == 1) basexrelative = 1;

else baserelative = 1;

}

else norelative = 1;

//상대 주소 계산

int relativeadr;

if(e == 0){relativeadr = right[1]\*16\*16 + data[2];}

else{relativeadr = right[1]\*16\*16\*16\*16 + data[2]\*16\*16 + data[3];}

if(right[1] >= 8){

char adr\_b[30];

itoa(relativeadr, adr\_b, 2);

if(e) adr\_b[20] = '\0'; else adr\_b[12] = '\0';

int i=0;

for(i=0; i<strlen(adr\_b); ++i){

if(adr\_b[i] == '0') adr\_b[i] = '1';

else adr\_b[i] = '0';

}

int minus\_adr = (int)strtol(adr\_b, NULL, 2);

minus\_adr++;

minus\_adr \*= -1;

relativeadr = minus\_adr;

}

//주소 계산

int adr;

if(pcrelative) adr = relativeadr + PC;

else if(baserelative) adr = relativeadr + B;

else if(basexrelative) adr = relativeadr + B + X;

else adr = relativeadr;

//OPCODE 실행

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Load\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//1. LDA

if(!strcmp(opcode\_str, "00")){

if(simpleadr){

int y = adr / line\_size; int x = adr % line\_size;

int data[5];

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

data[k] = mem[y][x];

x++;

if( x >= line\_size){ y++; x = 0;}

}

int finaldata = data[0] \* 16 \* 16 \* 16 \* 16 + data[1] \* 16 \* 16 + data[2];

A = finaldata;

}

else if(immediateadr){

A = adr;

}

}

//2. LDB

if(!strcmp(opcode\_str, "68")){

if(simpleadr){

int y = adr / line\_size; int x = adr % line\_size;

int data[5];

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

data[k] = mem[y][x];

x++;

if( x >= line\_size){ y++; x = 0;}

}

int finaldata = data[0] \* 16 \* 16 \* 16 \* 16 + data[1] \* 16 \* 16 + data[2];

B = finaldata;

}

else if(immediateadr){

B = adr;

}

}

//3. LDT

if(!strcmp(opcode\_str, "74")){

if(simpleadr){

int y = adr / line\_size; int x = adr % line\_size;

int data[5];

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

data[k] = mem[y][x];

x++;

if( x >= line\_size){ y++; x = 0;}

}

int finaldata = data[0] \* 16 \* 16 \* 16 \* 16 + data[1] \* 16 \* 16 + data[2];

T = finaldata;

}

else if(immediateadr){

T = adr;

}

}

//4. LDCH

if(!strcmp(opcode\_str, "50")){

A = mem[adr / line\_size][adr % line\_size];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Load\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Store\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//1. STA

if(!strcmp(opcode\_str, "0C")){

int data1, data2, data3;

char datastr1[3]; char datastr2[3]; char datastr3[3];

char datastr[10];

sprintf(datastr, "%06X", A); datastr[6] = '\0';

datastr1[0] = datastr[0]; datastr1[1] = datastr[1]; datastr1[2] = '\0';

data1 = (int)strtol(datastr1, NULL, 16);

datastr2[0] = datastr[2]; datastr2[1] = datastr[3]; datastr2[2] = '\0';

data2 = (int)strtol(datastr2, NULL, 16);

datastr3[0] = datastr[4]; datastr3[1] = datastr[5]; datastr1[2] = '\0';

data3 = (int)strtol(datastr3, NULL, 16);

edit(adr, data1);

edit(adr + 1, data2);

edit(adr + 2, data3);

}

//2. STX

if(!strcmp(opcode\_str, "10")){

int data1, data2, data3;

char datastr1[3]; char datastr2[3]; char datastr3[3];

char datastr[10];

sprintf(datastr, "%06X", X); datastr[6] = '\0';

datastr1[0] = datastr[0]; datastr1[1] = datastr[1]; datastr1[2] = '\0';

data1 = (int)strtol(datastr1, NULL, 16);

datastr2[0] = datastr[2]; datastr2[1] = datastr[3]; datastr2[2] = '\0';

data2 = (int)strtol(datastr2, NULL, 16);

datastr3[0] = datastr[4]; datastr3[1] = datastr[5]; datastr1[2] = '\0';

data3 = (int)strtol(datastr3, NULL, 16);

edit(adr, data1);

edit(adr + 1, data2);

edit(adr + 2, data3);

}

//3. STL

if(!strcmp(opcode\_str, "14")){

int data1, data2, data3;

char datastr1[5]; char datastr2[5]; char datastr3[5];

char datastr[10];

sprintf(datastr, "%06X", L); datastr[6] = '\0';

datastr1[0] = datastr[0]; datastr1[1] = datastr[1]; datastr1[2] = '\0';

data1 = (int)strtol(datastr1, NULL, 16);

datastr2[0] = datastr[2]; datastr2[1] = datastr[3]; datastr2[2] = '\0';

data2 = (int)strtol(datastr2, NULL, 16);

datastr3[0] = datastr[4]; datastr3[1] = datastr[5]; datastr1[2] = '\0';

data3 = (int)strtol(datastr3, NULL, 16);

//printf("수정된 data의 주소: %X, +1, +2 값: %X %X %X\n", adr, data1, data2, data3);

edit(adr, data1);

edit(adr + 1, data2);

edit(adr + 2, data3);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Store\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Comp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//1. COMP

//2. COMPR

if(!strcmp(opcode\_str, "A0")){

CC = 0;

}

//3. TD

if(!strcmp(opcode\_str, "E0")){

CC = -1;

}

//4. RD

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Comp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHANGE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//1. TIXR

if(!strcmp(opcode\_str, "B8")){

X++;

//T: left[1] == 5

if(X < T) CC = -1;

if(X == T) CC = 0;

}

//2. CLEAR

if(!strcmp(opcode\_str, "B4")){

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHANGE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JUMP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//1. JSUB

if(!strcmp(opcode\_str, "48")){

L = PC;

PC = adr;

}

//2. JEQ

if(!strcmp(opcode\_str, "30")){

if(CC == 0) PC = adr;

}

//3. J

if(!strcmp(opcode\_str, "3C")){

if(indirectadr){

PC = mem[adr / line\_size][adr % line\_size] \* 16 \* 16 \* 16 \* 16 + mem[(adr + 1) / line\_size][(adr + 1) % line\_size] \* 16 \* 16 + mem[(adr + 2) / line\_size][(adr + 2) % line\_size];

}

else PC = adr;

}

//4. JLT

if(!strcmp(opcode\_str, "38")){

if(CC < 0) PC = adr;

}

//5. RSUB

if(!strcmp(opcode\_str, "4C")){

PC = L;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JUMP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//breaking point 만나면 break

for(int b = 0; b < Bpcount; ++b){

bp = (int)strtol(Breakingpoint[b], NULL, 16);

if(bp == PC) {runEXECADDR = PC; actingflag = 1; break\_flag = 1; break;}

}

if(break\_flag) break;

}

printf("A: %06X X: %06X\nL: %06X PC: %06X\nB: %06X S: %06X\nT: %06X\n", A, X, L, PC, B, S, T);

//ENDADDR에 PC가 아직 도달하지 않았을 경우

if(PC < ENDADDR) printf(" Stop at checkpoint[%X]\n", bp);

//ENDADDR에 PC가 도달했을 경우

else printf(" End Program\n");

}

void addbp(char string[]){

//breaking point를 추가하는 함수

char datastr[10]; int data;

data = (int)strtol(string, NULL, 16);

sprintf(datastr, "%04X", data);

datastr[4] = '\0';

strcpy(Breakingpoint[Bpcount++], datastr);

printf(" [ok] create breakpoint %X\n", data);

}

void printbp(void){

//지금까지 저장한 breaking point를 출력하는 함수

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

for(int c = 0; c < Bpcount; ++c){

int data = (int)strtol(Breakingpoint[c], NULL, 16);

printf(" %X\n", data);

}

}

void clearbp(void){

//지금까지 저장한 breaking point를 삭제하는 함수

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

for(int c = 0; c < Bpcount; ++c){Breakingpoint[c][0] = '\0';}

Bpcount = 0;

}

void progaddr(char address[]){

//progaddr을 지정하는 함수

PROGADDR = (int)strtol(address, NULL, 16);

}

void loader(int parametercount, char file[][20])

{

totallength = 0;

int CSN\_sym\_count[3];

int EXTAB\_SYM\_FLAG = 0;

//PASS1

for(int f=0; f<parametercount; ++f){

int line = 1;

FILE \*ip;

int i = 0;

while((file[f][i] >= 'a' && file[f][i] <= 'z') || file[f][i] == '.'){

i++;

}

file[f][i] = '\0';

if(!(ip = fopen(file[f], "r"))){

fputs("file open error\n", stderr);

}

//해당 이름을 가진 파일이 디렉토리에 존재할 경우

else{

char buffer[100];

while(!feof(ip)){

if(fgets(buffer, sizeof(buffer), ip) == NULL) {break;}

else {

//Header Record

if(buffer[0] == 'H'){

int blank\_index = 0;

for(int i=1; i<=6; ++i){

int j = i - 1;

if(buffer[i] == ' ') {blank\_index = j; break;}

else EXTAB\_CSN[f][j] = buffer[i];

}

EXTAB\_CSN[f][blank\_index] = '\0';

for(int i=7; i<=12; ++i){

int j = i - 7;

EXTAB\_CSN\_ADR[f][j] = buffer[i];

}

for(int i=13; i<=18; ++i){

int j = i - 13;

EXTAB\_CSN\_LEN[f][j] = buffer[i];

}

}

//Define Record

else if(buffer[0] == 'D'){

EXTAB\_SYM\_FLAG = 1;

CSN\_sym\_count[f] = ((int)strlen(buffer) - 2) / 12;

int T = CSN\_sym\_count[f];

int c = 0;

int blank\_index = 0;

for(int t = 0; t < T; ++t){

for(int i=1 + 12\*t; i<=6 + 12\*t; ++i){

int j = i - (1 + 12\*t);

if(buffer[i] == ' ') {blank\_index = j; break;}

EXTAB\_SYM[f][c][j] = buffer[i];

}

EXTAB\_SYM[f][c][blank\_index] = '\0';

for(int i=7 + 12\*t; i<=12 + 12\*t; ++i){

int j = i - (7 + 12\*t);

EXTAB\_ADR[f][c][j] = buffer[i];

}

c++;

}

}

}

}

fclose(ip);

}

}

//가져온 정보를 이전에 저장한 PROGADDR에 알맞게 계산한다.

for(int f=0; f<parametercount; ++f){

if(f == 0) sprintf(EXTAB\_CSN\_ADR[0], "%X", (int)strtol(EXTAB\_CSN\_ADR[0], NULL, 16) + PROGADDR);

else sprintf(EXTAB\_CSN\_ADR[f], "%X", (int)strtol(EXTAB\_CSN\_ADR[f-1], NULL, 16) + (int)strtol(EXTAB\_CSN\_LEN[f-1], NULL, 16) + (int)strtol(EXTAB\_CSN[f], NULL, 16));

}

for(int f=0; f<parametercount; ++f){

for(int c=0; c<CSN\_sym\_count[f]; ++c){

sprintf(EXTAB\_ADR[f][c], "%04X", (int)strtol(EXTAB\_ADR[f][c], NULL, 16) + (int)strtol(EXTAB\_CSN\_ADR[f], NULL, 16));

}

}

//저장한 정보 출력

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

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

int EXTAB\_CSN\_ADR\_int = (int)strtol(EXTAB\_CSN\_ADR[0], NULL, 16);

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

{

if(parametercount > 1)

printf("%-20s%-10s%-10s\n", EXTAB\_CSN[i], EXTAB\_CSN\_ADR[i], EXTAB\_CSN\_LEN[i] + 2);

else

printf("%-20s %04X %-10s\n", EXTAB\_CSN[i], EXTAB\_CSN\_ADR\_int, EXTAB\_CSN\_LEN[i] + 2);

if(EXTAB\_SYM\_FLAG){

for(int j=0; j<2; ++j){

printf("%15s%9s\n", EXTAB\_SYM[i][j], EXTAB\_ADR[i][j]);

}

}

}

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

int len = (int)strtol(EXTAB\_CSN\_LEN[i], NULL, 16);

totallength += len;

}

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

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

int CSADDR = PROGADDR;

int EXECADDR = PROGADDR;

//PASS2

for(int f=0; f<parametercount; ++f){

int line = 1;

FILE \*ip;

if(!(ip = fopen(file[f], "r"))){

fputs("file open error\n", stderr);

}

//해당 이름을 가진 파일이 존재할 경우

else{

char buffer[100];

char ref[10][10];

while(!feof(ip)){

if(fgets(buffer, sizeof(buffer), ip) == NULL) {break;}

else {

//현재 다루고 있는 Control Section의 정보를 저장한다.

int CS\_code;

char ProgramNameStr[10];

//Header Record

if(buffer[0] == 'H'){

int blank\_index;

char ProgramNameStr[10];

//Program 이름 저장

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

if(buffer[i+1] == ' ') {blank\_index = i; break;}

else ProgramNameStr[i] = buffer[i+1];

}

ProgramNameStr[blank\_index] = '\0';

strcpy(ref[1], ProgramNameStr);

//현재 다루고 있는 Control Section 정보 저장

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

if(!strcmp(ProgramNameStr, EXTAB\_CSN[i])) CS\_code = i;

}

//현재 다루고 있는 Control Section의 주소 저장

CSADDR = (int)strtol(EXTAB\_CSN\_ADR[CS\_code], NULL, 16);

}

//Reference Record

else if(buffer[0] == 'R'){

strcpy(ref[1], ProgramNameStr);

int i = 2;

char \*temp\_ptr = strtok(buffer, " ");

strcpy(ref[i++], temp\_ptr + 3);

ref[i - 1][strlen(ref[i-1])] = '\0';

//token 하나하나에 접근한다.

//Refer되는 Operand의 정보를 저장한다.

while (temp\_ptr != NULL) {

temp\_ptr = strtok(NULL, " ");

if(temp\_ptr == NULL) {ref[i-1][strlen(ref[i-1])-1] = '\0'; break;}

strcpy(ref[i++], temp\_ptr + 2);

ref[i - 1][strlen(ref[i-1])] = '\0';

}

}

//Text Record

else if(buffer[0] == 'T'){

char SpectifiedAddrStr[10]; int SpectifiedAddr;

for(int i=1; i<=6; ++i) SpectifiedAddrStr[i-1] = buffer[i];

SpectifiedAddrStr[6] = '\0';

SpectifiedAddr = (int)strtol(SpectifiedAddrStr, NULL, 16);

char LineSizeStr[10]; int LineSize;

for(int i=7; i<=8; ++i) LineSizeStr[i-7] = buffer[i];

LineSizeStr[2] = '\0';

LineSize = (int)strtol(LineSizeStr, NULL, 16);

//Text Record에 있는 Object code들을 저장할 위치를 저장한다.

EXECADDR = CSADDR + SpectifiedAddr;

for(int i=9; i <= 9 + (LineSize \* 2 - 1) - 1; i = i + 2){

//Text Record의 Object code들을 적절한 위치에 load해준다.

char unitdatastr[5]; int unitdata;

unitdatastr[0] = buffer[i]; unitdatastr[1] = buffer[i+1]; unitdatastr[2] = '\0';

unitdata = (int)strtol(unitdatastr, NULL, 16);

edit(EXECADDR, unitdata);

EXECADDR++;

}

}

else if(buffer[0] == 'M'){

//수정해야 할 주소 지정

EXECADDR = (int)strtol(EXTAB\_CSN\_ADR[CS\_code], NULL, 16);

char RelativeAddrStr[10]; int RelativeAddr;

for(int i=1; i<=6; ++i) RelativeAddrStr[i-1] = buffer[i];

RelativeAddrStr[6] = '\0';

RelativeAddr = (int)strtol(RelativeAddrStr, NULL, 16);

EXECADDR += RelativeAddr;

//Replace = Operand(기존 mem에 저장된 값) + Operator(ex)LISTB)

int Replace; char temp\_Replace\_Str[10]; char Replace\_Str[10];

//1. Operand

char OperandStr[10]; int Operand; int OperandSize;

if(buffer[8] == '5') OperandSize = 5;

else if(buffer[8] == '6') OperandSize = 6;

if(OperandSize == 5){

int j=0;

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

int data;

if(i == EXECADDR) data = mem[i / line\_size][i % line\_size] % 16;

else data = mem[i / line\_size][i % line\_size];

char data\_str[3];

sprintf(data\_str, "%02X", data);

if(i == EXECADDR){OperandStr[j] = data\_str[0]; j++;}

else {OperandStr[j] = data\_str[0]; OperandStr[j+1] = data\_str[1]; j = j + 2;}

}

OperandStr[5] = '\0';

Operand = (int)strtol(OperandStr, NULL, 16);

}

else if(OperandSize == 6){

int j=0;

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

int data = mem[i / line\_size][i % line\_size];

char data\_str[3];

sprintf(data\_str, "%02X", data);

OperandStr[j] = data\_str[0]; OperandStr[j+1] = data\_str[1];

j = j + 2;

}

OperandStr[6] = '\0';

Operand = (int)strtol(OperandStr, NULL, 16);

}

//2. Operator

int plus\_flag = 0, minus\_flag = 0;

if(buffer[9] == '+') plus\_flag = 1;

else minus\_flag = 1;

char REF\_pos\_str[5]; int REF\_pos;

REF\_pos\_str[0] = buffer[10]; REF\_pos\_str[1] = buffer[11]; REF\_pos\_str[2] = '\0';

REF\_pos = (int)strtol(REF\_pos\_str, NULL, 16);

int code1, code2;

int FIND = 0;

int REFposIs1 = 0;

//Operator의 정보를 찾아서 저장한다.

for(code1 = 0; code1 < 3; ++code1){

for(code2 = 0; code2 < 2; ++code2){

if(!strcmp(ref[REF\_pos], EXTAB\_SYM[code1][code2])) {FIND = 1; break;}

}

if(FIND) break;

}

if(!strcmp(ref[REF\_pos], ProgramNameStr)) {REFposIs1 = 1;}

char OperatorStr[10]; int Operator;

if(REFposIs1) strcpy(OperatorStr, EXTAB\_CSN\_ADR[CS\_code]);

else strcpy(OperatorStr, EXTAB\_ADR[code1][code2]);

Operator = (int)strtol(OperatorStr, NULL, 16);

//지금까지 과정으로 구한 Operand와 Operator을 더하거나 뺌으로써 대체해야할 값을 계산한다.

if(plus\_flag) Replace = Operand + Operator;

if(minus\_flag) Replace = Operand - Operator;

if(OperandSize == 5){

sprintf(temp\_Replace\_Str, "%010X", Replace);

strcpy(Replace\_Str, temp\_Replace\_Str + 5);

int Str\_data = (int)strtol(Replace\_Str, NULL, 16);

sprintf(Replace\_Str, "%05X", Str\_data);

Replace\_Str[5] = '\0';

}

if(OperandSize == 6){

sprintf(temp\_Replace\_Str, "%010X", Replace);

strcpy(Replace\_Str, temp\_Replace\_Str + 4);

int Str\_data = (int)strtol(Replace\_Str, NULL, 16);

sprintf(Replace\_Str, "%06X", Str\_data);

Replace\_Str[6] = '\0';

}

//새로운 data(Replace)을 load한다.

if(OperandSize == 5){

int j = 0;

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

char newdata\_Str[3];

if(i == EXECADDR){

newdata\_Str[0] = Replace\_Str[j]; newdata\_Str[1] = '\0';

j++;

}

else{

newdata\_Str[0] = Replace\_Str[j]; newdata\_Str[1] = Replace\_Str[j+1];

newdata\_Str[2] = '\0';

j = j + 2;

}

int newdata = 0;

if(i == EXECADDR){

int olddata = mem[i / line\_size][i % line\_size];

newdata = (int)strtol(newdata\_Str, NULL, 16) + olddata - olddata % line\_size;

edit(i, newdata);

}

else {newdata = (int)strtol(newdata\_Str, NULL, 16); edit(i, newdata);}

}

}

else if(OperandSize == 6){

int j = 0;

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

char newdata\_Str[3];

newdata\_Str[j] = Replace\_Str[j]; newdata\_Str[j+1] = Replace\_Str[j+1];

newdata\_Str[j+2] = '\0';

int newdata = (int)strtol(newdata\_Str, NULL, 16);

edit(i, newdata);

j = j + 2;

}

}

}

}

}

fclose(ip);

}

}

//EXTAB 초기화

for(int i=0; i<3; ++i) {EXTAB\_CSN[i][0] = '\0'; EXTAB\_CSN\_ADR[i][0] = '\0'; EXTAB\_CSN\_LEN[i][0] = '\0';}

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

for(int j=0; j<10; ++j){

EXTAB\_SYM[i][j][0] = '\0';

EXTAB\_ADR[i][j][0] = '\0';

}

}

}