과목 명: 시스템프로그래밍

담당 교수 명: 박 운 상

<<Project 3>>

**서강대학교 컴퓨터학과**

**[학번] 20161565**

**[이름] 권기윤**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도

2.1 proj2 상세 흐름도

2.1 proj3 상세 흐름도

3. 모듈 정의

3.1 input\_handing.c 내부 추가함수

3.1.1 void fix\_isvalid\_input(char\* input);

3.2 loader\_command.c 함수

3.2.1 int cmd\_progaddr(char input[100][100], int tok\_num);

3.2.2 int cmd\_loader(char input[100][100],int tok\_num);

3.2.3 void init\_estab();

3.2.4 void add\_estab(int file\_num,char\* symbol,int addr);

3.2.5 int return\_estabAddr(char\* name);

3.2.6 int loader\_pass1(FILE\* fp[], int file\_number);

3.2.7 int loader\_pass2(FILE\* fp[], int file\_number);

3.2.8 void print\_loadMap(int file\_num);

3.3 assembly\_command.c 내부함수

3.3.1 void init\_reg();

3.3.2 int cmd\_run(char input[100][100],int tok\_num);

3.3.3 void run\_instruction(int opcode);

3.3.4 void store\_memory(int addr, int value, int byte);

3.3.5 int indirect\_addr(int addr, int byte);

3.3.6 void sync\_reg();

3.3.7 int find\_AddrorConst(int PC,int format);

3.3.8 void find\_nixbpe(int nixbpe[], int PC);

3.3.9 void print\_Reg();

3.3.10 int cmd\_bp(char input[100][100],int tok\_num);

3.3.11 void print\_bp();

3.3.11void insert\_bp(int addr);

4. 전역 변수 정의 4

4.1 추가 구현 구조체 정의

4.1.1 \_estab

4.1.2 \_register

4.2 추가 구현 변수 정의

4.2.1 int program\_address;

4.2.2 int bp[100], bp\_idx;

4.2.3 int estab\_Max[4];

4.2.4 int program\_Length[4];

4.2.5 int EXEADDR;

4.2.6 int loader\_flag;

4.2.7 ESTAB estab[4][100];

5. 코드 설명

5.1 20161565.h

5.2 2016156.c

5.3 input\_handling.c

5.4 shell\_command.c

5.5 memory\_command.c

5.6 opcode\_command.c

5.7 assembly\_command.c

5.8 loader\_command.c

5.9 run\_command.c

# 프로그램 개요

이 프로그램은 앞으로 구현하게 될 SIC/XE 머신을 위한 전 단계로 어셈블러, 링크, 로더를 실행할 쉘과 컴파일을 통해 만들어진 object 코드가 저장되고 실행될 메모리공간을 구현하였습니다. 또한 ADD, COMP와 같은 mnemonic 을 opcode 값으로 변화하여 hashTable에 저장하고 일부 명령어 (help, dir, quit, history, dump, edit, fill, reset, opcode, opcodelist)를 구현하는 프로그램입니다.

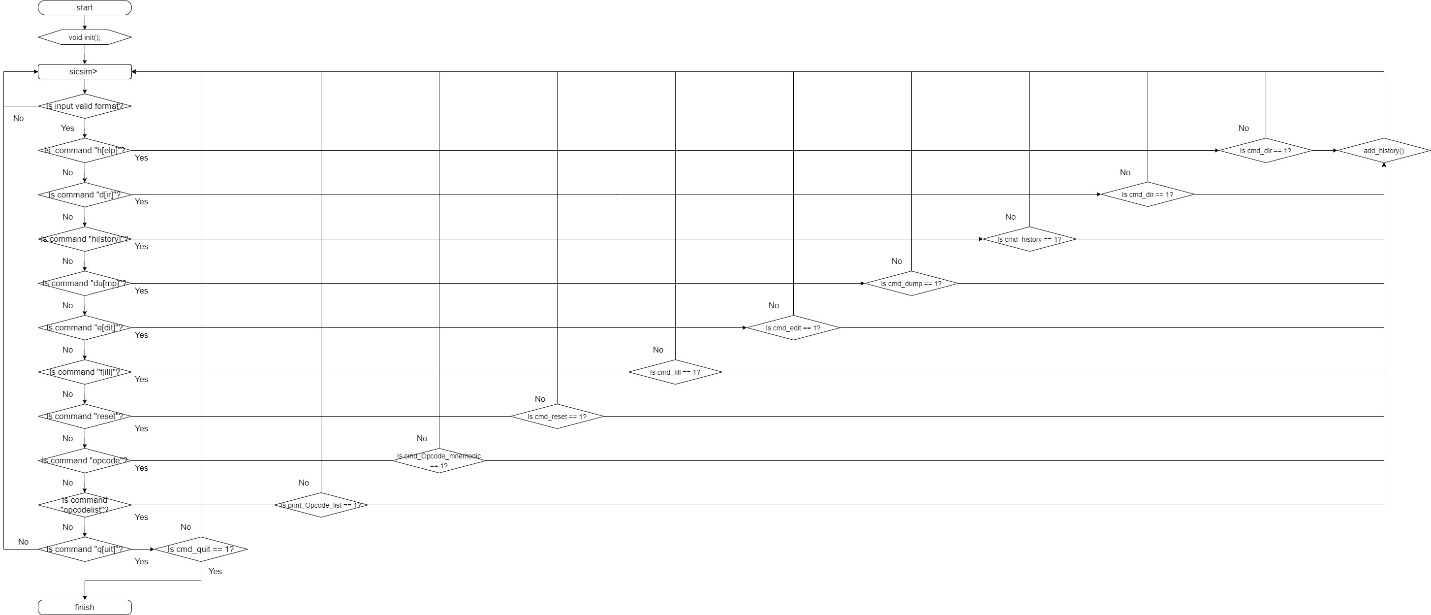
프로젝트 #1에서 구현한 셀(shell)에 assemble 기능을 추가하는 프로그램. SIC/XE의 assembly program source 파일을 입력 받아서 object파일을 생성하고, 어셈블리 과정 중 생 성된 symbol table과 결과물인 object 파일을 볼 수 있는 기능을 제공해야 함. 교재의 2.2까 지 설명된 SIC/XE 어셈블러의 기능을 구현함을 원칙으로 한다.

프로젝트 1, 2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가하는 프로그램이다. 프로젝트 2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모리에 올리는 일을 수행한다.

# 프로그램 설명

## 프로그램 흐름도

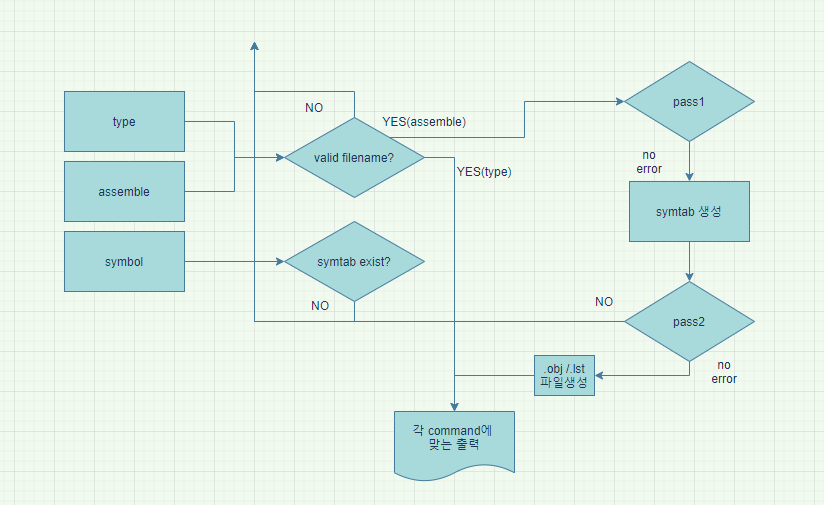
그림 1> 프로그램 흐름도



프로그램이 시작되면 void init(); 함수로 전역 구조체 동적할당, opcode hash tree 를 생성한다. 그 후 한 줄을 input\_str 에 입력받는다. 만약 올바른 형식으로 input\_str이 들어왔다면, input\_str 에서 tokenize한 명령어를 판별한다. 올바른 형식의 명령어와 명령어에 해당하는 올바른 형식이 들어온 것이 확인되면 add\_history 함수를 호출하여 input\_Str을 기록한다. 기록 후에는 다시 “sicsim> ”으로 돌아와 입력을 받으며, “q[uit]” 명령어가 입력되면 프로그램을 종료한다.

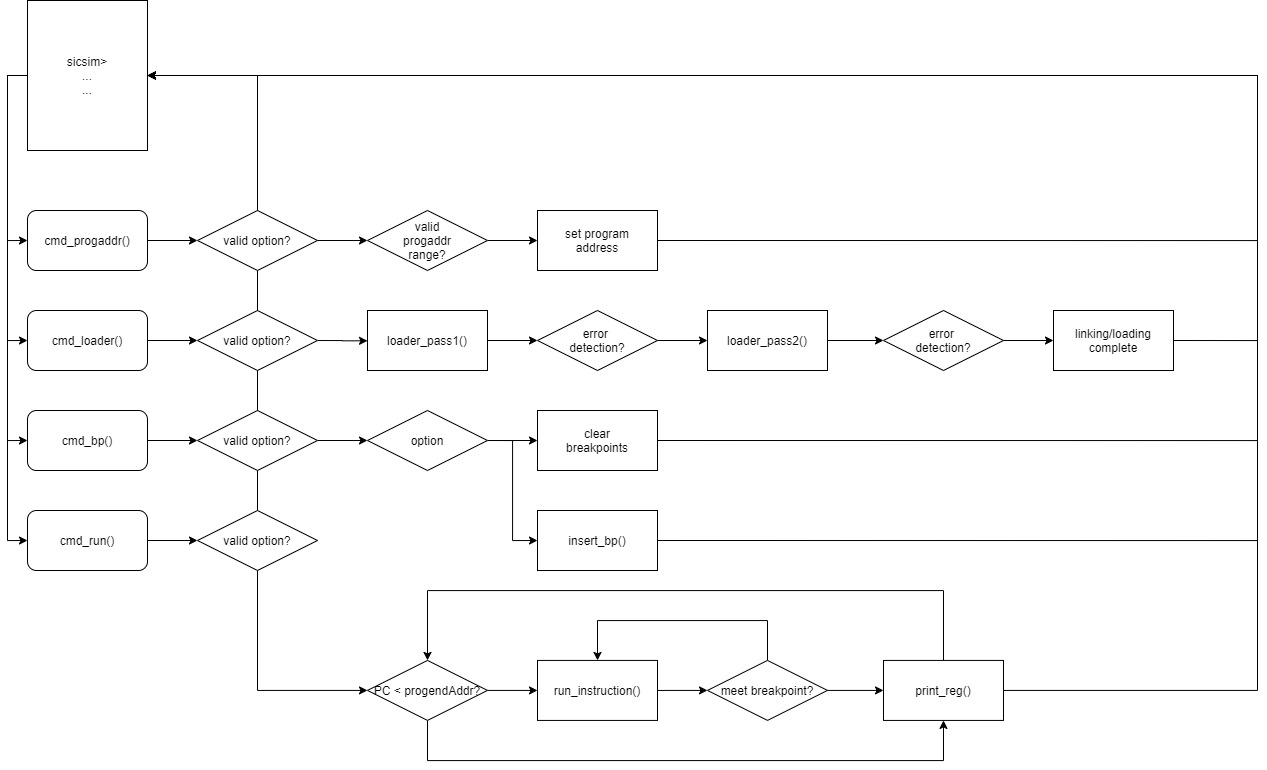
## Proj2 흐름도

그림 2> 프로그램 흐름도



## Proj3 흐름도

그림 3> 프로그램 흐름도



# 모듈 정의

## Input\_handling.c 추가함수

### void fix\_isvalid\_input(char\* input);

proj3에서 loader 명령어의 option에 맞추어 valid한 input format의 규칙을 다시 정해주었다.

다른 명령어들은 “command + ‘ ’ + ‘arg’ + ‘,’ + ‘arg’ + …”의 형태이지만

명령어 loader는 arg사이의 ‘,’가 없고 공백으로 arg를 구분해야하기 때문에 해당 경우, 즉, 명령어로 loader가 들어왔을 때의 valid 한 format 검사를 진행한다.

## Loader\_command.c 내부함수

### int cmd\_progaddr(char input[100][100], int tok\_num)h[elp]

명령어 progaddr 에 대한 처리를 담당한다. 적절한 option, 즉 address가 들어왔다면, load 시작 주소를 입력한 address로 초기화한다.

### void init\_estab()

struct array ESTAB estab[4][10] 의 내부 원소들을 모두 초기화한다. 또한 pLength, 프로그램 길이를 초기화한다.

### void add\_estab(int file\_num,char\* symbol,int addr)

file\_num 에 해당하는 estab[file\_num][estab\_Max[file\_num]] 에 parameter로 받은symbol 과 address를 저장한다. Symbol을 estab의 name에 넣을 때 공백이나 줄바꿈에 대해서 ‘\0’으로 초기화해준다.

### int return\_estabAddr(char\* name)

parameter로 받은 name과 estab의 name을 비교하여 같다면 해당 estab array 의 주소를 반환하고, 찾지 못했다면 -1을 반환한다.

### int cmd\_loader(char input[100][100],int tok\_num)

명령어 loader에 대한 처리를 담당한다. 처음에 input에 해당하는 string에 대해 해당 input이 현재 dircetory에 존재하는 filename.obj 파일인지 확인하고 맞다면 linker/loader에 대한 pass1, pass2를 실행한다. 실행과정에서 오류가 발생했다면 에러메시지와 함께 종료하고, 2pass에 대해 오류가 없다면 print\_loadMap() 함수를 호출하여 콘솔에서 linker/loader의 결과를 출력한다.

### int loader\_pass1(FILE\* fp[], int file\_number)

linker loader 의 2pass중 pass1에 해당하는 동작을 수행한다. Loader 명령어에서 입력받았던 object file들에 대해서 external symbol과 이에 해당하는 address들을 estab에 추가하는 동작을 수행한다. 만약 estab 원소 추가 중 해당 symbol이 중복이라면 에러메시지와 함께 종료한다. 중복없이 성공적으로 종료되었다면 각 file에 해당하는 pLength를 program\_Length array에 저장한다.

### int loader\_pass2(FILE\* fp[], int file\_number)

linker loader 의 2pass중 pass2에 해당하는 동작을 수행한다. loader명령어에서 입력받은 objectfile들에 대해서 각 file을 한 줄 씩 읽어가며 앞의 record 유형에 따라 해당하는 동작을 수행한다. 만약 ‘M’ record에서 ‘R’ record 에 존재하지 않는 refer number가 나온다면 에러메시지와 함께 종료한다. 성공적으로 종료되었다면 memory 배열에 각 file 과 estab에 해당하는 주소에 맞는 값이 들어있을 것이다.

### void print\_loadMap(int file\_num)

linker loader 2pass가 성공적으로 수행되었다면, cmd\_loader() 함수에서 load map의 정보를 출력하기위해 이 함수를 호출한다. 각 object file 에 대한 control section, symbol name, address, length를 출력한다.

## run\_command.c 내부함수

### void init\_reg()

struct store\_reg의 모든 원소를 초기화한다. PC 는 프로그램 시작 주소로, L은 program length 로 초기화한다.

### int cmd\_run(char input[100][100],int tok\_num);

run 명령어에 대한 처리를 담당한다. Loader가 성공적으로 수행되었다면 모든 register를 초기화하고, Program counter가 program\_endAddr 까지 수행될 동안 instruction을 수행하고, 수행된 결과에 대한 Program counter 값이 설정한 breakpoint와 같다면 함수를 종료하고, breakpoint에 해당하는 register 의 statement를 출력한다.

### void run\_instruction(int opcode);

run명령어에서 하나의 instruction을 동작시키기 위해 이 함수를 호출한다. 들어온 opcode 에 따라 format 을 1,2,3,4로 나눈다. Copy.obj는 format 1에 해당하는 opcode가 없으므로 이 경우 프로그램을 종료하고, 나머지의 format에 해당하는 opcode를 확인하고, 각 opcode에 대한 동작을 실행한다.

### void store\_memory(int addr, int value, int byte);

sic/xe 명령어중 store 관련 명령어를(ex. STA,STL) 처리할 때, 입력받은 value를 memory[addr] 부터 memory[addr+byte-1]에 8bit 씩 나누어 저장한다.

### int indirect\_addr(int addr, int byte);

해당 PC에 해당하는 instruction이 indirect addressing 일 경우 parameter로 들어온 addr에 해당하는 memory의 값을 반환한다.

### void sync\_reg();

struct store\_reg 의 integer 원소들과 array간의 sync를 맞추어준다.

### int find\_AddrorConst(int PC,int format)

해당 PC의 값에 해당하는 address 를 반환한다. Format 3의 경우 address는 12bit이고, format 4는 20bit 이므로 해당 PC의 format에 맞는 address를 계산하여 저장해주었다.

### void find\_nixbpe(int nixbpe[], int PC);

format 3,4 에 대하여 nixbpe[6] 배열에 차례대로 n, i, x, b, p ,e bit들에 대한 정보를 저장한다.

Memory[PC] % 4 ~ memory[PC+1]/16 까지의 값을 이진수로 바꾸어 nixbpe배열에 저장한다.

### void print\_Reg();

register A, L, B, T, X, PC, S 의 현재 값을 출력형식에 맞게 출력한다.

### int cmd\_bp(char input[100][100],int tok\_num);

bp 명령어에 대한 처리를 담당한다. Bp 다음 option이 “clear”라면 저장되어 있던 모든 breakpoints를 모두 지우고, 다음 option이 [address] 형태이고 memory의 범위를 벗어나지 않았다면 해당 address를 breakpoint에 추가한다.

### void insert\_bp(int addr);

parameter로 들어온 addr를 array bp[]에 추가한다. bp[bp\_idx]에 addr 저장 후에 bp\_idx를 1 증가시키고, 해당 bp array를 오름차순으로 sort하기 위해 각 insert\_bp 호출시마다 각 원소들과 추가된 원소를 비교해 swap해주면서 프로그램 전체적으로 보면 bubble sort와 같이 구현되게 하였다.

### void print\_bp();

입력이 명령어 bp 단독으로 들어왔다면, 사용자가 전에 입력했던 모든 breakpoints를 출력한다.

# 전역 변수 정의

## 추가 구현 구조체 정의

### Struct \_estab

char name[100];

* External symbol을 저장한다.

int address;

* External symbol에 해당하는 address를 저장한다.

### Struct \_registser

int A; // A register

int X; // X register

int L; // L register

int B; // B register

int S; // S register

int T; // T register

int F; // F register

int F2; // F2 register

int PC; // PC register

int SW; // SW register

int reg[20];

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

## 추가 구현 변수 정의

### Int program\_address;

명령어 Progaddr의 [address]를 저장한다.

### Int int bp[100], bp\_idx;

명령어 bp [address]에 대한 address를 저장하는 array bp 와 array bp의 개수를 저장하는 bp\_idx

### Int estab\_Max[4];

ESTAB estab의 각 column에 원소가 들어있는 개수를 저장한다. 즉, 각 file의 external symbol의 개수를 저장한다.

### Int program\_Length[4];

각 objectfile 에 대한 CSLTH를 저장한다.

### Int EXEADDR;

Program execute address를 저장한다.

### Int loader\_flag;

Loader 명령어가 성공적으로 수행되었다면, 1로 초기화한다. Loader\_flag가 1이라면 run 명령어를 실행시킬때, register들이 모두 초기화된다.

### ESTAB estab[4][100];

Linker loader 의 Pass1에서 external symbol에 대한 정보를 저장한다.

# 코드

## 20161565.h

#include "20161565.h"

void init\_reg(){// register를 초기화한다. PC는 프로그램 시작주소, L 은 프로그램 길이로 초기화한다.

store\_reg.A = 0x0;

store\_reg.X = 0x0;

store\_reg.L = 0x0;

store\_reg.B = 0x0;

store\_reg.T = 0x0;

store\_reg.F = 0x0;

store\_reg.F2 = 0x0;

store\_reg.PC = 0x0;

store\_reg.SW = 0x0;

for(int i=0;i<20;i++)

store\_reg.reg[i] =0;

store\_reg.PC = program\_address;

store\_reg.reg[8] = program\_address;

store\_reg.L = pLength;

store\_reg.reg[2] = pLength;

}

int cmd\_run(char input[100][100],int tok\_num){// 명령어 run에 대한 처리를 담당한다.

if(tok\_num != 1){

return -1;

}

if(loader\_flag == 1){

init\_reg();

loader\_flag = 0;

}

int prog\_endAddr = program\_address + pLength;

while(store\_reg.reg[8] < prog\_endAddr){

run\_instruction(memory[store\_reg.reg[8]]);

for(int i = 0; i < bp\_idx;i++){

if(store\_reg.reg[8] == bp[i]){

print\_Reg();

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

return 1;

}

}

}

print\_Reg();

printf("\t\tEnd Program\n");

return 1;

}

void run\_instruction(int opcode){// 한 instruction에 대한 연산을 수행한다.

int format = 0;

int check\_format = opcode/0x10;

switch(check\_format){

case 0xC:

case 0xF:

format = 1;

break;

case 0xA:

case 0xB:

format = 2;

break;

default:

format = 3;

if(memory[store\_reg.reg[8] + 1] & 0b10000)//if e bit is set

format = 4;

}

if(format == 1 || opcode == 0){//format 1 없음. 해당 memory가 비어있다면

store\_reg.reg[8]++;

return;

}

else if(format == 2){

int r1,r2;

r1 = memory[store\_reg.reg[8] + 1] / 0x10;

r2 = memory[store\_reg.reg[8] + 1] % 0x10;

switch(opcode){

case 0xB4: //CLEAR

store\_reg.reg[r1] = 0;

break;

case 0xA0: //COMPR

if(store\_reg.reg[r1] == store\_reg.reg[r2]){

store\_reg.reg[9] = '='; //reg[9] :SW

}

else if(store\_reg.reg[r1] > store\_reg.reg[r2]){

store\_reg.reg[9] = '>'; //reg[9] :SW

}

else{ //store\_reg.reg[r1] < store\_reg.reg[r2]

store\_reg.reg[9] = '<'; //reg[9] :SW

}

break;

case 0xB8://TIXR

store\_reg.reg[1]++; // X++;

if(store\_reg.reg[1] == store\_reg.reg[r1]){

store\_reg.reg[9] = '='; //reg[9] :SW

}

else if(store\_reg.reg[1] > store\_reg.reg[r1]){

store\_reg.reg[9] = '>'; //reg[9] :SW

}

else{ //store\_reg.reg[1] < store\_reg.reg[r2]

store\_reg.reg[9] = '<'; //reg[9] :SW

}

break;

}

store\_reg.reg[8] +=2; //PC += 2;

}

else{ //format 3 or 4

int nixbpe[6]= {0,0,0,0,0,0};

int addr = find\_AddrorConst(store\_reg.reg[8], format);

find\_nixbpe(nixbpe, store\_reg.reg[8]);

store\_reg.reg[8] += format;

if(format == 3 && (addr > (1<<11)) && (nixbpe[4] == 1)){//PC relative & signed bit 1

addr -= (1 << 12); //2's complement

}

if(nixbpe[2] == 1){ // x bit set

addr += store\_reg.reg[1];

}

if(nixbpe[3] == 1){ // b bit set

addr += store\_reg.reg[3];

}

if(nixbpe[4] == 1){ // p bit set

addr += store\_reg.reg[8];

}

if(nixbpe[0] == 1 && nixbpe[1] == 0){ //indirect aadressing

addr = indirect\_addr(addr, 3);

}

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

case 0x14: //STL

store\_memory(addr, store\_reg.reg[2], 3);// memory <- L

break;

case 0x68://LDB

if(nixbpe[0] == 0 && nixbpe[1] == 1){

store\_reg.reg[3] = addr; // B <- addr

}

else{

store\_reg.reg[3] = indirect\_addr(addr,3);

}

break;

case 0x48://+JSUB

store\_reg.reg[2] = store\_reg.reg[8];

store\_reg.reg[8] = addr;

break;

case 0x00://LDA

if(nixbpe[0] == 0 && nixbpe[1] == 1){

store\_reg.reg[0] = addr; // B <- addr

}

else{

store\_reg.reg[0] = indirect\_addr(addr,3);

}

break;

case 0x28://COMP

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr,3);

}

if(store\_reg.reg[0] == addr){

store\_reg.reg[9] = '=';

}

else if(store\_reg.reg[0] > addr){

store\_reg.reg[9] = '>';

}

else{ //store\_reg.reg[0] < addr

store\_reg.reg[9] = '<';

}

break;

case 0x30://JEQ

if(store\_reg.reg[9] == '='){

store\_reg.reg[8] = addr;

}

break;

case 0x3C://J

store\_reg.reg[8] = addr;

break;

case 0x0C://STA

store\_memory(addr,store\_reg.reg[0],3);

break;

case 0x74://LDT

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr, 3);

}

store\_reg.reg[5] = addr;

break;

case 0xE0://TD

//다음 instruction으로 넘어가되 CC 는 '<'로 변경되었다고 가정한다.

store\_reg.reg[9] = '<';

break;

case 0xD8://RD

//input device로부터 아무것도 받지 못했다고 가정.

// 다음 명령어인 COMPR A,S의 결과 CC 가 '=' 이라고 가정한다.

store\_reg.reg[0] = 0; // A = 0

store\_reg.reg[9] = '=';

break;

case 0x54://STCH

store\_memory(addr, store\_reg.reg[0] % 0X10, 1);

break;

case 0x38://JLT

if(store\_reg.reg[9] == '<'){

store\_reg.reg[8] = addr;

}

break;

case 0x10://STX

store\_memory(addr, store\_reg.reg[1], 3);

break;

case 0x4C://RSUB

store\_reg.reg[8] = store\_reg.reg[2];//PC = l

break;

case 0x50://LDCH

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr,3);

}

store\_reg.reg[0] =(store\_reg.reg[0] & 0xFFFFFF00) + (addr/0x10000);

break;

case 0xDC://WD

//다음 instruction으로 넘어간다.

break;

}

}

sync\_reg();//store\_reg의 register array와 각 register간의 동기화

}

void store\_memory(int addr, int value, int byte){ // value를 memory의 해당 address에 byte만큼 저장한다.

char tmp[10], word[3];

if(value < 0){

value -= 0xFF000000;

}

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

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

strncpy(word,tmp + i \* 2 ,2);

word[2] = '\0';

memory[addr+i] = strtol(word, NULL, 16);

}

};

int indirect\_addr(int addr, int byte){//n=0,i=1 일때 indircet address를 반환한다.

int indir\_addr = memory[addr];

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

indir\_addr \*= 0x100;

indir\_addr += memory[addr+i];

}

return indir\_addr;

};

void sync\_reg(){//store\_reg의 register array와 각 register간의 동기화

store\_reg.A = store\_reg.reg[0];

store\_reg.X = store\_reg.reg[1];

store\_reg.L = store\_reg.reg[2];

store\_reg.B = store\_reg.reg[3];

store\_reg.S = store\_reg.reg[4];

store\_reg.T = store\_reg.reg[5];

store\_reg.F = store\_reg.reg[6];

store\_reg.F2 = store\_reg.reg[7];

store\_reg.PC = store\_reg.reg[8];

store\_reg.SW = store\_reg.reg[9];

}

int find\_AddrorConst(int PC,int format){ // format에 따른 address or constant를 반환한다.

int addr = memory[PC + 1] % 0x10;

addr \*= 0x100;

addr += memory[PC + 2];

if(format == 4){//format == 4

addr \*= 0x100;

addr += memory[PC+3];

}

return addr;

}

void find\_nixbpe(int nixbpe[], int PC){// array nixbpe[6]에 순서대로 n,i,x,b,p,e에 해당하는 bit condition을 저장한다.

int ni = memory[PC] % 0b100;

int xbpe = memory[PC + 1] / 0b10000;

if(ni & 0b10){

nixbpe[0] = 1;

}

if(ni & 0b01){

nixbpe[1] = 1;

}

if(xbpe & (0b1000)){//x bit set

nixbpe[2] = 1;

}

if(xbpe & (0b0100)){//b bit set

nixbpe[3] = 1;

}

if(xbpe & (0b0010)){//p bit set

nixbpe[4] = 1;

}

if(xbpe & (0b0001)){//e bit set

nixbpe[5] = 1;

}

}

void print\_Reg(){//register들을 출력한다.

printf("\tA : %06X X : %06X\n",store\_reg.A, store\_reg.X);

printf("\tL : %06X PC : %06X\n",store\_reg.L, store\_reg.PC);

printf("\tB : %06X S : %06X\n",store\_reg.B, store\_reg.S);

printf("\tT : %06X\n",store\_reg.T);

}

int cmd\_bp(char input[100][100],int tok\_num){// 명령어 bp에 대한 연산을 수행한다.

if(tok\_num > 2){

return -1;

}

else if(tok\_num == 1){

print\_bp();

}

else{

int i;

if(strcmp(input[1],"clear") == 0){// bp clear

//clear bp

for(i = 0; bp[i] != -1; i++){

bp[i] = -1;

}

bp\_idx = 0;

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

}

else{ // bp [address]

int bp\_address = strtoHex(input[1]);

if(bp\_address < 0 || bp\_address > 0xFFFFF){

printf("Invalid break point address.\n");

return 0;

}

insert\_bp(bp\_address);

printf("\t\t[ok] create breakpoint %X\n",bp\_address);

}

}

return 1;

}

void insert\_bp(int addr){ //명령어 bp [address] : address를 bp\_address 배열에 저장 후 오름차순으로 sort하는 함수

if(bp\_idx >= 100){

printf("Number of breakpoint no more than 100.\n");

return;

}

bp[bp\_idx++] = addr;

if (bp\_idx == 1) return ;

for(int i = bp\_idx-1 ; i > 0;i--){

if(bp[i] < bp[i-1]){

int tmp = bp[i];

bp[i] = bp[i-1];

bp[i-1] = tmp;

}

}

};

void print\_bp(){ //명령어 bp : 모든 breakpoint를 출력하는 함수

printf("\t\tbreakpoint\n");

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

for(int i = 0; i < bp\_idx; i++){

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

}

};

## 20161565.c

#include "20161565.h"

int main(){

char input\_str[100];// SIC/XE 환경에서 들어온 input string을 저장

char real\_input[100]; // input\_str 에서 받은 input에서 앞, 뒤 공백 제거한 string

char tmp\_history[100]; // input history

char input\_tok[100][100]; // parsing 된 real\_input 의 command, register를 저장

char command[100]; // parsing 된input\_tok[0]을 저장

char\* ptr;

int par\_num; //parsing 된 문자열의 개수

int input\_switch; // 명령어 관련 함수에서 반환되는 값을 저장하여 맞게 작동되었는지 확인한다.

//input\_switch = 1 : 함수가 올바르게 작동

//input\_switch = 0 : 함수가 올바르게 작동하지 못하여 함수 내에서 error message 를 출력

//input\_switch = -1: 함수가 올바르게 작동하지 못하여 error message 출력 필요

init();// opcode hash table을 생성하고 전역변수의 동적할당을 실행한다.

while(1){

printf("sicsim> ");

fgets(input\_str,100,stdin);

if((int) strlen(input\_str) > 100){

printf("INPUT COMMAND LENGTH MUST BE LESS THAN 100.\n");

continue;

}

input\_str[strlen(input\_str) -1] = '\0'; // fgets는 '\n' 까지 입력받음으로 마지막 인덱스에 해당하는 값을 '\0'로 변경

strcpy(real\_input,trim\_str(input\_str)); // 들어온 input string을 ltrim, rtrim 하여 real\_input으로 복사

strcpy(tmp\_history, real\_input);

//

fflush(stdin); // 남아있는 buffer를 비워준다.

//

ptr = strstr(real\_input,"loader");

if(ptr == real\_input){

if(fix\_isvalid\_input(real\_input) == -1){

printf("\"%s\" is a invalid input. Check 'h[elp]'.\n",real\_input);

}

}

else if(isvalid\_input(real\_input)== -1){// input stirng 이 invalid 한 형태일경우

printf("\"%s\" is a invalid input. Check 'h[elp]'.\n",real\_input);

continue;

}

parse\_input(input\_str,input\_tok,&par\_num); // real\_input을 tokenize한다.

if(real\_input[0] == '\0'){// 입력받은 input string이 없거나 공백일경우

continue;

}

strcpy(command,input\_tok[0]); //첫번째 tokenize string 을 command 로 복사

if(strcmp(command,"h") == 0 || strcmp(command,"help") == 0){// h[elp]

input\_switch = cmd\_help(par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command,"d") == 0 || strcmp(command,"dir") == 0){// d[ir]

input\_switch = cmd\_dir(par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command,"q") == 0 || strcmp(command, "quit") == 0){// q[uit]

input\_switch = cmd\_quit(par\_num);

if (input\_switch == 1) break;

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "hi") == 0 || strcmp(command, "history") == 0){//hi[story]

input\_switch = cmd\_history(real\_input, par\_num);

if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "du") == 0 || strcmp(command, "dump") == 0 ){//du[mp] [start, end]

input\_switch = cmd\_dump(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command,"e") == 0 || strcmp(command, "edit")==0){// e[dit] start, end

input\_switch = cmd\_edit(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("e[dit]\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "f") == 0 || strcmp(command, "fill") == 0){// f[ill] start, end, value

input\_switch = cmd\_fill(input\_tok, par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "reset") == 0){// reset

input\_switch = cmd\_reset(par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "opcode") == 0){// opcode mnemonic

input\_switch = cmd\_Opcode\_mnemonic(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "opcodelist") == 0){// opcodelist

input\_switch = print\_Opcode\_list(par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "type")== 0){ // cmd\_type

input\_switch = cmd\_typeFilename(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if (strcmp(command, "assemble") == 0){// cmd\_assemble

input\_switch = cmd\_assembleFilename(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "symbol") == 0){// cmd\_symbol

input\_switch = cmd\_symbol(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "progaddr") == 0){// cmd\_symbol

input\_switch = cmd\_progaddr(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "loader") == 0){// cmd\_symbol

input\_switch = cmd\_loader(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "bp") == 0){// cmd\_symbol

input\_switch = cmd\_bp(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else if(strcmp(command, "run") == 0){// cmd\_symbol

input\_switch = cmd\_run(input\_tok,par\_num);

if(input\_switch == 1){

add\_history(tmp\_history);

}

else if(input\_switch == -1){

printf("\"%s\" is invalid option. Check h[elp].\n",real\_input);

}

}

else{ // invalid command

printf("\"%s\" is invalid input. Check h[elp].\n",tmp\_history);

}

}

return 0;

}

void init(){

int i;

history\_head = NULL; //history\_head 초기화

symtab = NULL;

tmp\_symtab = NULL;

opTable = (Opcode\_table \*\*)malloc(20 \* sizeof(Opcode\_table \*)); // opTable[20] 동적할당

for(i = 0;i < 20 ; i++) opTable[i] = NULL; //opTable 초기화

Open\_Opcode(); // opcode hash table 생성

memory = (unsigned char \*)calloc(MAX\_MEMORY,sizeof(unsigned char));// 가상 메모리 동적할당

// init assembly\_command.c

st\_flag = 0;

Starting\_address = 0;

pLength = 0;

programName[0] = 0;

BASE\_line\_idx = 0;

BASE\_flag = 0;

BASE\_name[0] = 0;

End\_flag = 0 ;

lst\_idx = 0;

Modify\_idx = 0;

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

modify[i] = 0;

}

for(int i = 0 ; i< 2000;i++){

lstArr[i].Loc\_counter = 0;

lstArr[i].label\_flag = 0;

lstArr[i].Objcode = 0;

lstArr[i].printObjnum = 0;

lstArr[i].line[0] = 0;

lstArr[i].str[0] = 0;

}

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

bp[i]= -1;

}

bp\_idx = 0;

init\_estab();

loader\_flag = -1;

}

## Input\_handling.h

#include "20161565.h"

void parse\_input(char\* input,char parsed\_tok[100][100],int\* tok\_num){// 들어온input을 tokenzie 한다.

char\* token;

\*tok\_num =0;

token = strtok(input,"\t ,"); // '\t', ' ', ','을 기준으로 input string을 tokenize

while(token != NULL){

strcpy(parsed\_tok[(\*tok\_num)],token);

(\*tok\_num)++;

token = strtok(NULL,"\t ,");

}

};

int fix\_isvalid\_input(char\* input){// 들어온 input 중 command가 "loader"인 경우

int i,flag =0;

char tmp\_str[100];

int char\_func;

strcpy(tmp\_str, input);

for(i = 0; tmp\_str[i] != '\0'; i++){

char\_func = isvalid\_char(tmp\_str[i]);

if(flag == 0){

switch(char\_func){// command...

case 0:

flag = 1;

break;

case 1:

break;

default:

return -1;

}

}

else if(flag == 1){// command + ' '...

switch(char\_func){

case 0:

break;

case 1:

flag = 2;

break;

default:

return -1;

}

}

else if(flag == 2){// command + ' ' + arg + ...

switch(char\_func){

case 0:

flag = 1;

break;

case 1:

break;

default:

return -1;

}

}

}

if(flag == 2)

return 1;

return -1;

}

int isvalid\_input(char\* input){// 들어온 input string이 valid 한 형태인지 확인한다.

int i;

int flag = 0;

//flag 0 : command 단독

//flag 1 : command + ' 'or '\t'..

//flag 2 : command + ' ' + arg..

//flag 3 : command + ' ' + arg + ','..

//flag 4 : command + ' ' + arg + ' '..

char tmp\_str[100];

int char\_func;

strcpy(tmp\_str, input);

for(i = 0 ; tmp\_str[i] != '\0';i++){

char\_func = isvalid\_char(tmp\_str[i]);

if(flag == 0){//command...

switch(char\_func){

case 0:

flag = 1;

break;

case 1:

break;

default:

return -1;

}

}

else if (flag == 1){//command"\_\_"...

switch(char\_func){

case 0:

break;

case 1:

flag = 2;

break;

default:

return -1;

}

}

else if(flag == 2){//command\_\_"arg"...

switch(char\_func){

case 2:

flag = 3;

break;

case 1:

break;

case 0:

flag = 4;

break;

}

}

else if(flag == 3){//command\_\_arg","...

switch(char\_func){

case 2:

return -1;

case 1:

flag = 2;

break;

}

}

else if(flag == 4){//command\_\_arg"\_\_"

switch(char\_func){

case 1:

return -1;

case 2:

flag = 3;

break;

}

}

}

if(flag == 3) return -1; // input string이 , 로 끝나는 경우

else return 1; // valid한 형태인 경우

};

int isvalid\_char(char c ){// 해당 character 가 valid 한지 확인

int i;

for(i = 0; i < 26; i++){

if(('a' + i) == c || ('A' + i) == c) return 1;

}// character is 'a' ~ 'z' or 'A' ~'Z''

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

if(('0' + i) == c) return 1;

}// character is '0' ~ '9'

if(c == ' ' || c == '\t') return 0; //character is space or tab

if (c == ',') return 2; // character is comma

return 1;

}

char\* trim\_str(char\* input){// input string 의 앞, 뒤 공백 제거

int i=0;

int len;

int ptr = 0;

char trim[100];

if(input == NULL){

return input;

}

len = strlen(input);

strcpy(trim,input);

for(i = 0 ; i< len ;i++){// ltrim

if(trim[i] == ' ' || trim[i] == '\t' || trim[i] == '\n'){

ptr++;

}

else break;

}

for(i = len-2; i >= 0;i--){ //rtrim

if(input[i] == ' ' || input[i] == '\t' || input[i] == '\n'){

trim[i] = '\0';

}

else break;

}

return input + ptr; //trim 된 input string 반환

};

## Shell\_command.h

#include "20161565.h"

int cmd\_help(int tok\_num){// h[elp] 명령어를 처리한다.

if(tok\_num != 1){//invalid arg

return -1;

}

//valid command

printf("\t\th[elp]\n");

printf("\t\td[ir]\n");

printf("\t\tq[uit]\n");

printf("\t\thi[story]\n");

printf("\t\tdu[mp] [start, end]\n");

printf("\t\te[dit] address, value\n");

printf("\t\tf[ill] start, end, value\n");

printf("\t\treset\n");

printf("\t\topcode mnemonic\n");

printf("\t\topcodelist\n");

printf("\t\tassemble filename\n");

printf("\t\ttype filename\n");

printf("\t\tsymbol\n");

printf("\t\tprogaddr [address]\n");

printf("\t\tloader [object filename1] [object filename2] [...]\n");

printf("\t\tbp [address]\n");

printf("\t\tbp clear\n");

printf("\t\tbp\n");

printf("\t\trun\n");

return 1;

};

int cmd\_dir(int tok\_num){// d[ir] 명령어를 처리한다.

if(tok\_num != 1){// invalid arg

return -1;

}

int i = 1;

DIR\* cur\_dir = NULL;

struct dirent\* dir\_name;

struct stat file\_info;

cur\_dir = opendir("."); // 현재 directory open

if(NULL != cur\_dir){

while((dir\_name = readdir(cur\_dir))!= NULL){

printf("%25s",dir\_name->d\_name);

stat(dir\_name->d\_name, &file\_info);

switch(file\_info.st\_mode & S\_IFMT){

case S\_IFREG://일반 file 인 경우

printf("\*");

break;

case S\_IFDIR://directory 인 경우

printf("/");

break;

}

if(i % 3 == 0){

printf("\n");

i %= 3;

}

i++;

}

}

else{// 현재 directory에 어떤 file, directroy 도 존재하지 않는 경우

printf("There is no files or directories in this directory.\n");

}

closedir(cur\_dir);// directory close

printf("\n");

return 1;

};

int cmd\_quit(int tok\_num){

//free linked list

// history, opcode hash Table

if(tok\_num != 1){// invalid arg

return -1;

}

HISTORY\* pRemove;

while(history\_head != NULL){

pRemove = history\_head;

//

//printf("%d\n",pRemove->num);

//

history\_head = history\_head->link;

free(pRemove);

}

//linked list : history free

//opcode\_hash\_table

int i;

Opcode\_table\* opFree, \*contents;

for(i = 0; i<20; i++){

contents = opTable[i];

while(contents != NULL){

opFree = contents;

contents = contents->link;

free(opFree);

}

free(contents);

}

//linked list : opcode hash table free

free(memory);

// 1MB memory free

return 1;

};

int cmd\_history(char\* input\_str, int tok\_num){// hi[story] 명령어를 처리한다.

if(tok\_num != 1){//invalid arg

return -1;

}

add\_history(input\_str); //입력된 hi[story] 또한 history linked list 에 저장

HISTORY \*pWalk;

pWalk = history\_head;

while(pWalk != NULL){

printf("\t%-4d %s\n",pWalk->num,pWalk->command);

pWalk = pWalk->link;

}

return 1;

};

int cmd\_typeFilename(char input[100][100], int tok\_num){

if(tok\_num != 2){

return -1;

}

char filename[100];

FILE\* fp = NULL;

strcpy(filename, input[1]);

fp = fopen(filename, "r");

if(fp == NULL){

printf("Cannot find filename \"%s\" in this directory.\n",filename);

return 0;

}

char print\_ch;

while(1){

print\_ch = fgetc(fp);

if(print\_ch == EOF){

break;

}

printf("%c",print\_ch);

}

fclose(fp);

return 1;

}

void add\_history(char \*input\_command){// valid 한 input string 을 history\_head 에 push

HISTORY\* pNew = (HISTORY \*)malloc(sizeof(HISTORY));

HISTORY\* pMove;

strcpy(pNew->command, input\_command);

pNew->link = NULL;

pNew->num = 1;

if(!history\_head){

history\_head = pNew;

return;

}

pMove = history\_head;

while(pMove->link != NULL){

pMove = pMove->link;

}

pMove->link = pNew;

pNew->num += pMove->num;

};

## Opcode\_command.h

#include "20161565.h"

void Open\_Opcode(){// "opcode.txt"를 연다.

FILE\* fp = NULL;

fp = fopen("opcode.txt","r"); // open file "opcode.txt"

if(!fp){ // can't read '"opcode.txt"

printf("Cannot read \"opcode.txt\" file.\n");

printf("File not exist or currpted.\n");

return;

}

int opcode,idx;

char mnemonic[10];

char format[10];

while(fscanf(fp,"%X %s %s\n",&opcode, mnemonic, format)!=EOF){// "opcode.txt"에서 한줄씩 scan

Opcode\_table\* pNew = (Opcode\_table \*)malloc(sizeof(Opcode\_table));

// create new node

pNew->link = NULL;

pNew->value = opcode;

strcpy(pNew->key, mnemonic);

strcpy(pNew->opcode\_format, format);

idx = create\_hashfunction(mnemonic);// 입력받은 mnemonic에 해당하는 hash function 생성

gen\_Opcode\_table(pNew,idx); // opcode hash table 구성

}

fclose(fp); // close "opcode.txt"

};

int create\_hashfunction(char\* mnemonic){// 입력받은 mnemonic에 해당하는 hasg function 생성

// add all character in mnemonic string

// return mod this by 20

int hash = 0,i;

for(i = 0; i< strlen(mnemonic);i++){

hash += (int) mnemonic[i];

}

return hash % 20;

}

void gen\_Opcode\_table(Opcode\_table \*pNew,int idx){// opcode hash table 을 구성한다.

//hash function에 해당하는 index에 opcode, mneminic 저장

if(opTable[idx] == NULL){

opTable[idx] = pNew;

}

else{

Opcode\_table\* pWalk;

pWalk = opTable[idx];

while(pWalk->link != NULL){

pWalk = pWalk->link;

}

pWalk->link = pNew;

}

};

int cmd\_Opcode\_mnemonic(char input[100][100],int tok\_num){// opcode mnemonic 명령어를 처리한다.

// if input start with "opcode" and next input is mnemonic

// find mnemonic in opTable and if mnemonic exist, print matchde opcode.

char mnemonic[100];

int flag;

if(tok\_num != 2){//invalid args

return -1;

}

strcpy(mnemonic, input[1]);

flag = find\_Opcode\_mnemonic(mnemonic);//입력받은 mnemonic과 일치하는 opcode return

if( flag == -1){

printf("\"%s\"is not exist in Opcode table.\n",input[1]);

return 0;

}

else{

return 1;

}

};

int find\_Opcode\_mnemonic(char\* mnemonic){//opcode hash table에서 입력받은 mnemonic 과 일치하는 opcode return

// find Opcode\_mnemonic

// if find input mnemonic in Opcode\_hash\_table, return 0.

// if not print caution message and return 1.

int i;

Opcode\_table\* pFind;

for(i = 0; i< 20 ; i++){

pFind = opTable[i];

while(pFind != NULL){

if(strcmp(pFind->key, mnemonic) == 0){

printf("opcode is %X\n",pFind->value);

return 1;

}

pFind = pFind->link;

}

}

// if input mnemonic can not found in opTable.

return -1;

}

int print\_Opcode\_list(int tok\_num){// opcodelist 명령어를 처리한다

//print opTable

if(opTable == NULL){//opcode hash table이 없는 경우

printf("Opcode list not exist. Check your file.\n");

return 0;

}

if(tok\_num != 1){//invalid args

return -1;

}

int i;

Opcode\_table \*pMove;

for(i = 0; i<20;i++){

pMove = opTable[i];

printf("%2d : ",i);

if(pMove == NULL){

printf("\n");

continue;

}

while(pMove->link != NULL){

printf("[%s, %X] -> ",pMove->key, pMove->value);

pMove = pMove->link;

}

printf("[%s %X]\n",pMove->key, pMove->value);

}

return 1;

};

## Memoey\_command.h

#include "20161565.h"

#define MIN\_VALUE 32

#define MAX\_VALUE 126

int cmd\_dump(char input[100][100], int tok\_num){// du[mp] [start, end] 명령어를 처리한다.

// no arg

// start only

// start, end

int start, end;

int ptr = MEMORY\_IDX;

if(tok\_num > 3){

return -1;

}

switch(tok\_num){

case 1: //dump

if(MEMORY\_IDX >0xFFFFF) MEMORY\_IDX = 0;

start = MEMORY\_IDX;

end = min(MEMORY\_IDX + 0x9F, 0xFFFFF);

MEMORY\_IDX = end + 1;

break;

case 2: //dump start

start = strtoHex(input[1]);

end = min(start + 0x9F,0xFFFFF);

MEMORY\_IDX = end + 1;

break;

case 3://dump start, end

start = strtoHex(input[1]);

end = strtoHex(input[2]);

MEMORY\_IDX = end + 1;

break;

}

if(start == -1 || end == -1){ // input start or end are out of range

printf("check input. start or end address is not hexamecimal number.\n");

MEMORY\_IDX = ptr;

return 0;

}

if(valid\_startendRange(start, end) == -1){//check both start and end address are valid format

MEMORY\_IDX = ptr;

return 0;

}

int i,j;

int dump\_start = (start / 16) \* 16; //print dump start line index

int dump\_end = (end / 16) \* 16; //print dump end line index

int print\_line; // number of print line

int line\_idx;

print\_line = (dump\_end - dump\_start) / 16 + 1;

for(i = 0; i < print\_line; i++){

line\_idx = dump\_start + 16 \* i;

printf("%05X ", line\_idx);

for(j = 0; j <16 ; j++){

if(line\_idx + j < start || line\_idx + j> end){

printf(" ");

}

else{

printf("%02X ", memory[line\_idx + j] );

}

}

printf("; ");

for(j = 0; j < 16; j++){

if(line\_idx + j < start || line\_idx + j > end){

printf(".");

}

else if ( memory[line\_idx + j] < MIN\_VALUE || memory[line\_idx + j] > MAX\_VALUE){

printf(".");

}

else{

printf("%c",memory[line\_idx + j]);

}

}

printf("\n");

}

return 1;

};

int strtoHex(char\* parameter){ // 해당 parameter를 16진수로 변환하여 return

int hexToint = 0;

if(valid\_hexa(parameter) == -1){//invalid character in parameter

return -1;

}

hexToint = strtol(parameter, NULL, 16); // strTohex

return hexToint;

}

int valid\_hexa(char\* c){// 입력받은 c string이 16진수의 형태를 띄고 있는지 확인한다.

// check if st and end are right range

int i;

for(i = 0 ; i < strlen(c);i++){

if(('a' <= c[i]&& c[i] <= 'f') || ('A' <= c[i] && c[i] <= 'F') || ('0' <= c[i] && c[i] <= '9')) continue;

else {

return -1;

}

}

return 1;

};

int valid\_startendRange(int st\_addr, int en\_addr){// 입력받은 start end address 가 valid 한지 확인한다.

if( st\_addr > 0xFFFFF || en\_addr > 0xFFFFF){ // address is too big

printf("start or end address is less than 0xFFFFF.\n");

return -1;

}

if(st\_addr > en\_addr){// start address is bigger than end address

printf("end address must be greater than start address.\n");

return -1;

}

return 1;

};

int cmd\_edit(char input[100][100], int tok\_num){// e[dit] address, value 명령어를 처리한다.

int address;

int value;

if(tok\_num != 3) {//invalid args

return -1;

}

address = strtoHex(input[1]);

value = strtoHex(input[2]);

if(address == -1 || value == -1){// address or value is not hexamecimal number

printf("Check input. Address or value is not hexadecimal number.\n");

return 0;

}

if(value > 0xFF){// value is too big

printf("Check input. Value must be less than 0xFF.\n");

}

if (address > 0xFFFFF){// address is too big

printf("Check input. Address must be less than 0xFFFFF.\n");

}

if((value > 0xFF) || (address > 0xFFFFF)){

return 0;

}

unsigned char cpy\_val = value;

memory[address] = cpy\_val;

return 1;

};

int cmd\_fill(char input[100][100], int tok\_num){// f[ill] start, end, value 명령어를 처리한다.

int start, end , value;

int i;

if(tok\_num != 4){

return -1;

}

start = strtoHex(input[1]);

end = strtoHex(input[2]);

value = strtoHex(input[3]);

if(start == -1 || end == -1 || value == -1){

printf("Check input. Start, end address or value are not hexadecimal number.\n ");

return 0;

}

//start, end range check

if(valid\_startendRange(start,end) == -1){

return 0;

}

//value range check;

if(value > 0xFF){

printf("Value must be less than 0xFF.\n");

return 0;

}

unsigned char cpy\_val = value;

//fill memory with value from start address to end address

for(i = start; i < end + 0x01; i++){

memcpy(memory + i , &cpy\_val, 1);

}

return 1;

};

int cmd\_reset(int tok\_num){// reset 명령어를 처리한다.

if(tok\_num != 1){

return -1;

}

int i;

for(i = 0; i < MAX\_MEMORY; i++){

memory[i] = 0x00;

}

return 1;

};

## Assembly\_command.h

#include "20161565.h"

int pass1(FILE\* asm\_file){ // process for pass1

char asm\_tok[100][100]; //store tokenize file line

char one\_line[100], tmp\_line[100];//file line string

char\* token; //token string

int asmToken; // number of tokenize string

int LOCCTR = 0; //Location counter

int label\_flag = 0; // check if line have label

while(fgets(one\_line, 100, asm\_file) != NULL){// read one line in asm ifle

if(strlen(one\_line) != 0){//enter 입력제외

one\_line[strlen(one\_line)-1] = '\0';

strcpy(tmp\_line,one\_line);

}

strcpy(lstArr[lst\_idx].line, one\_line);

//parsing

asmToken = 0;

token = strtok(tmp\_line,", \t");

while(token != NULL){

strcpy(asm\_tok[asmToken], token);

asmToken++;

token = strtok(NULL,", \t");

}

if(asmToken == 0){//all blank

lstArr[lst\_idx].Loc\_counter = -1;

lstArr[lst\_idx].Objcode = -1;

lstArr[lst\_idx].label\_flag = 2;

lst\_idx++;

continue;

}

switch(one\_line[0]){

case '.': //주석

label\_flag = -1;

lstArr[lst\_idx].label\_flag = label\_flag;

break;

case ' ': case '\t': // no label

label\_flag = 0;

lstArr[lst\_idx].label\_flag = label\_flag;

break;

default://has label

label\_flag = 1;

lstArr[lst\_idx].label\_flag = label\_flag;

break;

}

if(label\_flag == -1){//주석

lstArr[lst\_idx].Loc\_counter = -1;

lstArr[lst\_idx].Objcode = -1;

lst\_idx++;

continue;

}

else if(label\_flag == 1){//if have label, store in symtab

int sym\_flag = add\_symTab(asm\_tok,asmToken,LOCCTR);

if(sym\_flag == -1){

printf("Line %3d : \"%s\" duplicate symbol.\n", 5 + lst\_idx \* 5, asm\_tok[0]);

}

}

//assembly directive

//"START"

if(st\_flag == 0){//first line

if((strcmp(asm\_tok[label\_flag],"START") == 0)){//OPCODE"START"

Starting\_address = atoi(asm\_tok[label\_flag + 1]);

lstArr[lst\_idx].Loc\_counter = Starting\_address;

lstArr[lst\_idx].Objcode = -1;

LOCCTR = Starting\_address;

strcpy(programName,asm\_tok[0]);

lst\_idx++;

st\_flag = 1;

continue;

}

// OPCODE "START" 미존재시 Starting address에 0 저장 후 LOCCTR 에 Starting address 저장

LOCCTR = Starting\_address = 0;

st\_flag = -1;

}

//"BASE"

if(strcmp(asm\_tok[label\_flag],"BASE") == 0){

lstArr[lst\_idx].Loc\_counter = -1;

lstArr[lst\_idx].Objcode = -1;

BASE\_flag = 1;

BASE\_line\_idx = lst\_idx;

strcpy(BASE\_name,asm\_tok[label\_flag+1]);

}

//"BYTE"

else if(strcmp(asm\_tok[label\_flag],"BYTE") == 0){

lstArr[lst\_idx].Loc\_counter = LOCCTR;

lstArr[lst\_idx].Objcode = 0;

if(asm\_tok[label\_flag+1][0] == 'X'){

LOCCTR += 1;

}

else if(asm\_tok[label\_flag+1][0] == 'C'){

LOCCTR += strlen(asm\_tok[label\_flag+1])-3; // C , ',' 제외

}

}

//"WORD"

else if(strcmp(asm\_tok[label\_flag],"WORD") == 0){

lstArr[lst\_idx].Loc\_counter = LOCCTR;

lstArr[lst\_idx].Objcode = 0;

LOCCTR +=3;

}

//"RESB"

else if(strcmp(asm\_tok[label\_flag],"RESB") == 0){

lstArr[lst\_idx].Loc\_counter = LOCCTR;

lstArr[lst\_idx].Objcode = -1;

LOCCTR += atoi(asm\_tok[label\_flag +1]);

}

//RESW

else if(strcmp(asm\_tok[label\_flag],"RESW") == 0){

lstArr[lst\_idx].Loc\_counter = LOCCTR;

lstArr[lst\_idx].Objcode = -1;

LOCCTR += 3 \* atoi(asm\_tok[label\_flag + 1]);

}

//END

else if(strcmp(asm\_tok[label\_flag],"END") == 0){

lstArr[lst\_idx].Loc\_counter = -1;

lstArr[lst\_idx].Objcode = -1;

End\_flag = 1;

pLength = LOCCTR - Starting\_address;

lst\_idx++;

break;

}

//opcode mnemonic

else{

char opcd[100];

int ext\_flag =0;

int format;

if(asm\_tok[label\_flag][0] == '+'){//extended mode

ext\_flag = 2;

strcpy(opcd, asm\_tok[label\_flag] + 1);

}

else strcpy(opcd,asm\_tok[label\_flag]);

Opcode\_table\* opFind = find\_OpcodeNode(opcd);

if(opFind == NULL){ //cannot find opcode mnemonic in opcode hash table

//error message

printf("line %3d : Invalid opcode mnemonic \"%s\".\n",5 \* lst\_idx + 5, asm\_tok[label\_flag]);

return 0;

}

lstArr[lst\_idx].Loc\_counter = LOCCTR;

lstArr[lst\_idx].Objcode = 0;

format = (int) opFind->opcode\_format[ext\_flag] - '0';

// opcode node 에 저장된 정보 -> lstNode

LOCCTR += format;

}

lst\_idx++;

}

return 1;

}

int pass2(){ // process for pass2

int asmToken = 0; //number of tokenize string

char asm\_tok[100][100]; //store tokenize string

char tmp\_line[100], tmp\_str[100]; //store temp string

char OPCODE[100]; // OPCODE string

char OPERAND[100]; // OPERAND string

char \*token; //tokenize string

int idx; // index

int n ,i, x, b, p, e, opHex1,opHex2,disp; // bit n,i,x,b.p,e and opHex1,2 for store opcode ,and disp address

int address\_loc, tmpValue; // address\_loc for symbol label

long long Cal\_ObjCode = 0; // store calculated objccode

long long nixbpe = 0; // store nixbpe bits

for(idx = 0; idx < lst\_idx; idx++){

if(lstArr[idx].Objcode == -1){ //dosen't need to calculate objcode

continue;

}

//init

n = i = x = b = p = e = opHex1 = opHex2 = disp = 0;

strcpy(tmp\_line,lstArr[idx].line);

// tokenize string

asmToken = 0;

token = strtok(tmp\_line,", \t");

while(token != NULL){

strcpy(asm\_tok[asmToken], token);

asmToken++;

token = strtok(NULL,", \t");

}

for(int k = 0; k < 100 ;k++){//init2

asm\_tok[asmToken][k] = 0;

}

int idx\_label = lstArr[idx].label\_flag;

if(strcmp(asm\_tok[idx\_label],"BYTE") == 0){//byte

if(asm\_tok[idx\_label + 1][0] == 'C'){// store char

strcpy(tmp\_str, asm\_tok[idx\_label+1] +2);

tmp\_str[strlen(tmp\_str)-1] = '\0';

for(int k = 0;k < strlen(tmp\_str);k++){

disp \*= 256;

disp += tmp\_str[k];

}

lstArr[idx].Objcode = disp;

}

else if(asm\_tok[idx\_label + 1][0] == 'X'){//store hex

strcpy(tmp\_str, asm\_tok[idx\_label+1] +2);

tmp\_str[strlen(tmp\_str)-1] = '\0';

lstArr[idx].Objcode = strtoHex(tmp\_str);

strcpy(lstArr[idx].str,tmp\_str);

lstArr[idx].printObjnum = -1;

if(lstArr[idx].Objcode == -1){

printf("Line %3d : Wrong Hexnumber.\n", idx\*5 + 5);

}

}

else{// wrong operand

printf("Line %3d : Wrong byte operand.\n", idx \* 5 + 5);

return 0;

}

}

else if(strcmp(asm\_tok[idx\_label],"WORD") == 0){//word

lstArr[idx].Objcode = (long long)atoi(asm\_tok[idx\_label+1]);

lstArr[idx].printObjnum = 6;

}

else if(strcmp(asm\_tok[idx\_label],"RESB") == 0){//resb

lstArr[idx].Objcode = -1;

}

else if(strcmp(asm\_tok[idx\_label],"RESW") == 0){// resw

lstArr[idx].Objcode = -1;

}

else{

int format\_idx = 0;

if(asm\_tok[idx\_label][0] == '+'){ // extend mode

strcpy(OPCODE,asm\_tok[idx\_label]+1);

e = 1;

}

else{

strcpy(OPCODE,asm\_tok[idx\_label]);

}

if(e == 1) format\_idx = 2;

Opcode\_table\* tmpOpnode = find\_OpcodeNode(OPCODE);

opHex1 = tmpOpnode->value / 16;

opHex2 = tmpOpnode->value % 16;

if(tmpOpnode->opcode\_format[format\_idx] == '1'){

//format 1

lstArr[idx].Objcode = tmpOpnode->value;

lstArr[idx].printObjnum = 2;

//lstArr[]

}

else if(tmpOpnode->opcode\_format[format\_idx] == '2'){

//format 2

tmpValue = find\_register(asm\_tok[idx\_label+1]);

if(tmpValue == -1){

printf("Line %3d : Cannot find register \"%s\".\n", idx\* 5 + 5, asm\_tok[idx\_label+1]);

return 0;

}

Cal\_ObjCode = (long long) (tmpOpnode->value << 8) + (tmpValue << 4);

tmpValue = find\_register(asm\_tok[idx\_label+2]);

if(tmpValue == -1){

printf("Line %3d : Cannot find register \"%s\".\n", idx\* 5 + 5, asm\_tok[idx\_label+1]);

return 0;

}

Cal\_ObjCode += (long long) tmpValue;

lstArr[idx].Objcode = Cal\_ObjCode;

lstArr[idx].printObjnum = 4;

}

else if(tmpOpnode->opcode\_format[format\_idx] == '3' ||tmpOpnode->opcode\_format[format\_idx] == '4' ){

//format 3 or 4

if(asmToken > idx\_label + 1){

if(asm\_tok[idx\_label+1][0] == '@'){// indirec addressing

n = 1;

i = 0;

strcpy(OPERAND,asm\_tok[idx\_label+1]+1);

}

else if(asm\_tok[idx\_label+1][0] == '#'){//immediate addressing

n = 0;

i = 1;

strcpy(OPERAND,asm\_tok[idx\_label+1]+1);

}

else{// simple addressing

n = 1;

i = 1;

strcpy(OPERAND,asm\_tok[idx\_label+1]);

}

if(asmToken == idx\_label + 3){//if number of operand is 2

x = 1;

}

address\_loc = find\_symbol(OPERAND);

if(address\_loc == -1){

//error message

disp = atoi(OPERAND);

if(disp == 0 && OPERAND[0] != '0'){// cannot find symbol

printf("Line %3d : Symbol \"%s\" dosen't exist.\n", idx\*5+5,OPERAND);

return 0;

}

//stored symbol LOCCTR out of range

if(e == 0 && disp >= 4096){//format 3

printf("Line %3d : Displacement out of range.\n", idx\*5+5);

return 0;

}

if(e == 1 && disp >= 1048576){//format 4

printf("Line %3d : Displacement out of range.\n", idx\*5+5);

return 0;

}

}

else if(e == 0){

int PC = address\_loc - lstArr[idx].Loc\_counter - 3;

if(-2048 <= PC && PC <= 2047){//PC relative

p = 1;

if(PC < 0){

PC += 4096;

disp = PC;

}

else{

disp = PC;

}

}

else {// base relative

b = 1;

if(BASE\_flag == 1){

int BASE\_LOC = find\_symbol(BASE\_name);

if(BASE\_LOC == -1){

printf("Line %3d : Wrong Base name.\n",idx\*5 +5);

return 0;

}

disp = address\_loc - BASE\_LOC;

}

else{// no "BASE" OPCODE

disp = -1;

}

if(0 > disp && disp > 4095){

printf("Line %3d : Address out of range.\n", idx\*5 + 5);

return 0;

}

}

}

}

else{

n = 1;

i = 1;

}

if(e){

b = 0;

p = 0;

}

//calculate objcode

Cal\_ObjCode = (long long) tmpOpnode->value >> 2;

nixbpe = (long long) (n << 5);

nixbpe += (long long) (i << 4);

nixbpe += (long long) (x << 3);

nixbpe += (long long) (b << 2);

nixbpe += (long long) (p << 1);

nixbpe += (long long) e;

lstArr[idx].Objcode = (Cal\_ObjCode << 6) + nixbpe;

if(e == 0){// format 3

lstArr[idx].Objcode \*= (1 << 12);

lstArr[idx].Objcode += disp;

lstArr[idx].printObjnum = 6;

}

else{ //if extend

if(address\_loc != -1){

disp = address\_loc;

modify[Modify\_idx++] = lstArr[idx].Loc\_counter + 1;

lstArr[idx].printObjnum = 8;

}

lstArr[idx].Objcode \*= (1<<20);

lstArr[idx].Objcode += disp;

lstArr[idx].printObjnum = 8;

}

}

}

}

return 1;

}

int cmd\_assembleFilename(char input[100][100], int tok\_num){// function for cmd assemble

if(tok\_num != 2){//wrong option

return -1;

}

FILE\* fp = NULL;

char filename[100], lstfilename[100], objfilename[100];

int step1,step2;

strcpy(filename, input[1]);

if(isdotAsm(filename) == 0){

return 0;

}

fp = fopen(filename, "r");

if(fp == NULL){//file not exist

printf("Cannot find filename \"%s\" in this directory.\n",filename);

return 0;

}

//assemble init

tmp\_symtab = symtab;

symtab = NULL;

st\_flag = 0;

Starting\_address = 0;

pLength = 0;

programName[0] = 0;

BASE\_line\_idx = 0;

BASE\_flag = 0;

BASE\_name[0] = 0;

End\_flag = 0 ;

lst\_idx = 0;

Modify\_idx = 0;

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

modify[i] = 0;

}

for(int i = 0 ; i< 2000;i++){

lstArr[i].Loc\_counter = 0;

lstArr[i].label\_flag = 0;

lstArr[i].Objcode = 0;

lstArr[i].printObjnum = 0;

lstArr[i].line[0] = 0;

lstArr[i].str[0] = 0;

}

//pass1 실행

step1 = pass1(fp);

if(step1 != 1){

//free tmp\_symtab

symtab = tmp\_symtab;

fclose(fp);

return 0;

}

//pass2 실행

step2 = pass2();

if(step2 != 1){

//free symtab

symtab = tmp\_symtab;

fclose(fp);

return 0;

}

// pass1 과 pass2 가 정상적으로 작동하였을때

fclose(fp);

//free\_symtab(symtab);

tmp\_symtab = NULL;

char tmp\_fname[100];

char\* tok\_filename;

strcpy(tmp\_fname,filename);

// create filename

tok\_filename = strtok(tmp\_fname,".");

strcpy(lstfilename, tok\_filename);

strcat(lstfilename,".lst");

strcpy(objfilename, tok\_filename);

strcat(objfilename,".obj");

//filename.lst 생성

fp = fopen(lstfilename,"w");

makeLstFile(fp);

fclose(fp);

//filename.obj 생성

fp = fopen(objfilename,"w");

makeObjFile(fp);

fclose(fp);

printf("[%s], [%s]\n",lstfilename,objfilename);

return 1;

}

void makeLstFile(FILE\* lstfile){// write information at .lst file

int i;

for(i = 0; i < lst\_idx; i++){

fprintf(lstfile, "%-8d\t",i \* 5 + 5); //print line num

if(lstArr[i].Loc\_counter == -1){

fprintf(lstfile, " \t");

}

else{

fprintf(lstfile, "%04X\t", lstArr[i].Loc\_counter);

}

fprintf(lstfile, "%s", lstArr[i].line);

if(lstArr[i].Objcode == -1){

fprintf(lstfile, "\n");

}

else{

for(int j = strlen(lstArr[i].line); j < 40;j++){

fprintf(lstfile," ");

}

switch (lstArr[i].printObjnum)

{

case 2:

fprintf(lstfile, "%02llX\n",lstArr[i].Objcode);

break;

case 4:

fprintf(lstfile, "%04llX\n",lstArr[i].Objcode);

break;

case 6:

fprintf(lstfile, "%06llX\n",lstArr[i].Objcode);

break;

case 8:

fprintf(lstfile, "%08llX\n",lstArr[i].Objcode);

break;

default:

fprintf(lstfile, "%02llX\n",lstArr[i].Objcode);

break;

}

}

}

}

void makeObjFile(FILE\* Objfile){// write information at .obj file

int i;

char str[300] = {0,},tmp\_oneline[100];

fprintf(Objfile,"H%-6s%06X%06X\n",programName,lstArr[0].Loc\_counter,pLength);

for(i = 0; i< lst\_idx;){

str[0] = '\0';

tmp\_oneline[0] = '\0';

if(lstArr[i].Loc\_counter == -1 || lstArr[i].Objcode == -1){

i++;

continue;

}

fprintf(Objfile,"T%06X",lstArr[i].Loc\_counter);

for(;i<lst\_idx;i++){

if(lstArr[i].Loc\_counter == -1){

continue;

}

if(lstArr[i].Objcode == -1){

break;

}

else if(lstArr[i].printObjnum == -1){

strcpy(tmp\_oneline,lstArr[i].str);

//fprintf(Objfile,"%02llX",lstArr[i].Objcode);

}

else{

switch(lstArr[i].printObjnum){

case 2:

sprintf(tmp\_oneline,"%02llX",lstArr[i].Objcode);

break;

case 4:

sprintf(tmp\_oneline,"%04llX",lstArr[i].Objcode);

break;

case 6:

sprintf(tmp\_oneline,"%06llX",lstArr[i].Objcode);

break;

case 8:

sprintf(tmp\_oneline,"%08llX",lstArr[i].Objcode);

break;

default:

sprintf(tmp\_oneline,"%llX",lstArr[i].Objcode);

break;

}

}

if(strlen(str) + strlen(tmp\_oneline) > 60){

i--;

break;

}

strcat(str,tmp\_oneline);

}

fprintf(Objfile,"%02X%s\n",(int)strlen(str)/2,str);

i++;

}

for(i = 0; i<Modify\_idx;i++){//modify

fprintf(Objfile, "M%06X05\n",modify[i]);

}

//if(st\_flag == 1)

fprintf(Objfile, "E%06X\n", lstArr[0].Loc\_counter);

}

int isdotAsm(char\* filename){// check if file extension end with ".asm"

char\* dotAsm = ".asm";

char\* ptr = strstr(filename, dotAsm);

if(ptr == NULL || strcmp(dotAsm,ptr) != 0){

printf("Valid filename extension of source file is \".asm\". Check your input.\n");

return 0;

}

return 1;

}

int cmd\_symbol(char input[100][100], int tok\_num){// function for cmd symbol

if(tok\_num != 1){

return -1;

}

if(symtab == NULL){

printf("SYMTAB not exist!\n");

return 1;

}

symbolNode\* pWalk;

pWalk = symtab;

while(pWalk != NULL){

printf("\t%s\t%04X\n",pWalk->Label,pWalk->Loc\_counter);

pWalk = pWalk->link;

}

return 1;

}

int add\_symTab(char asm\_tok[100][100], int tok\_num,int LOCCTR){

symbolNode\* pNew, \*pWalk, \*pPrev;

pNew = (symbolNode \*)malloc(sizeof(symbolNode));

pNew->Loc\_counter = LOCCTR;

strcpy(pNew->Label, asm\_tok[0]);

pNew->link = NULL;

if(symtab == NULL){

symtab = pNew;

return 1;

}

pWalk = symtab;

pPrev = pWalk;

while(pWalk != NULL){

if(strcmp(pWalk->Label, pNew->Label) == 0){

return -1; // same label error

}

if(pWalk == symtab){

if(strcmp(pWalk->Label,pNew->Label) > 0){

pNew->link = pWalk;

symtab = pNew;

return 1;

}

}

else if(strcmp(pWalk->Label,pNew->Label) > 0){

pNew->link = pPrev->link;

pPrev->link = pNew;

return 1;

}

pPrev = pWalk;

pWalk = pWalk->link;

}

pPrev->link = pNew;

return 1;

}

int find\_symbol(char\* operand){// function for find symbol

symbolNode\* pWalk;

pWalk = symtab;

while(pWalk != NULL){

if(strcmp(pWalk->Label,operand) == 0){

return pWalk->Loc\_counter;

}

pWalk = pWalk->link;

}

return -1;

}

int find\_register(char\* reg){// function for find register

if(strcmp(reg, "A") == 0 || strcmp(reg, "") == 0){

return 0;

}

else if(strcmp(reg, "X") == 0){

return 1;

}

else if(strcmp(reg, "L") == 0){

return 2;

}

else if(strcmp(reg, "B") == 0){

return 3;

}

else if(strcmp(reg, "S") == 0){

return 4;

}

else if(strcmp(reg, "T") == 0){

return 5;

}

else if(strcmp(reg, "F") == 0){

return 6;

}

else if(strcmp(reg, "PC") == 0){

return 8;

}

else if(strcmp(reg, "SW") == 0){

return 9;

}

// can not find register

return -1;

}

Opcode\_table\* find\_OpcodeNode(char\* mnemonic){//opcode hash table에서 입력받은 mnemonic 과 일치하는 opcode return

// find Opcode\_mnemonic

// if find input mnemonic in Opcode\_hash\_table, return 0.

// if not print caution message and return 1.

int i;

Opcode\_table\* pFind;

for(i = 0; i< 20 ; i++){

pFind = opTable[i];

while(pFind != NULL){

if(strcmp(pFind->key, mnemonic) == 0){

//printf("opcode is %X\n",pFind->value);

return pFind;

}

pFind = pFind->link;

}

}

// if input mnemonic can not found in opTable.

return NULL;

}

void free\_symtab(symbolNode\* remove){//free symbolnode

symbolNode\* pRemove;

while(remove != NULL){

pRemove = remove;

//

//printf("%d\n",pRemove->num);

//

remove = remove->link;

free(pRemove);

}

}

## Loader\_command.h

#include "20161565.h"

int cmd\_progaddr(char input[100][100], int tok\_num){// progaddr [address] : address를 program 시작주소로 지정해준다.

if(tok\_num >2){

return -1;

}

int prog\_addr = strtol(input[1],NULL,16); // [address]는 string으로 입력되므로 16진수로 변환

if(prog\_addr < 0 || prog\_addr > 0xFFFFF){// prog\_addr range check

program\_address = 0;

printf("Program address out of range.\n");

return 0;

}

program\_address = EXEADDR;

return 1;

}

void init\_estab(){//estab을 초기화 한다.

pLength = 0;

for(int i=0;i<4;i++){

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

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

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

}

estab\_Max[i] = 0;

}

}

void add\_estab(int file\_num,char\* symbol,int addr){// estab[file\_num]의 맨끝에 external symbol과 주소를 추가한다.

for(int i =0 ; i<6;i++){ // 6자리에 ' '이나 '\n'이 들어갈 경우 '\0'으로 바꾸어줌

if(symbol[i] == ' ' || symbol[i] == '\n'){

symbol[i] = '\0';

}

}

strcpy(estab[file\_num][estab\_Max[file\_num]].name,symbol);

estab[file\_num][estab\_Max[file\_num]].address = addr;

estab\_Max[file\_num]++;

}

int return\_estabAddr(char\* name){// 해당 name과 일치하는 estab의 address를 반환한다.

for(int i = 0; i < 4; i++){

for(int j = 0 ; j < estab\_Max[i]; j++){

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

return estab[i][j].address;

}

}

}

return -1;

}

int cmd\_loader(char input[100][100],int tok\_num){// 명령어 loader [filename1.obj] .. 에 관한 명령을 실행한다.

if(tok\_num == 1 || tok\_num > 4){//input check

return -1;

}

FILE\* fp[4];

int file\_number;

int pass1,pass2;

init\_estab();

for(int i=0; i< tok\_num-1;i++){// filename.obj 파일을 읽어 fp[i]에 저장한다.

fp[i] = fopen(input[i+1],"r");

if(fp[i] == NULL){

printf("\"%s\"is not exist in current directory.\n", input[i+1]);

return 0;

}

}

file\_number = tok\_num - 1;

pass1 = loader\_pass1(fp,file\_number);

if(pass1 == 0){

init\_estab();

loader\_flag = -1;

return 0;

}

for(int i=0;i<file\_number;i++){

fclose(fp[i]);

fp[i] = fopen(input[i+1],"r");

if(fp[i] == NULL){

printf("\"%s\"is not exist in current directory.\n", input[i+1]);

return 0;

}

}

pass2 = loader\_pass2(fp,file\_number);

if(pass2 == 0){

init\_estab();

loader\_flag = -1;

return 0;

}

print\_loadMap(file\_number);

for(int i=0;i<file\_number;i++){

fclose(fp[i]);

}

loader\_flag =1;

return 1;

}

int loader\_pass1(FILE\* fp[], int file\_number){// loader 2pass중 pass1 에 해당하는 동작을 수행한다.

int CSADDR = 0;

int CSLTH;

char str[100];

char prog\_name[10];

CSADDR = program\_address;

for(int i = 0; i < file\_number; i++){

fgets(str,100,fp[i]);

strncpy(prog\_name,str+1,6);

prog\_name[6] = '\0';

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

if(return\_estabAddr(prog\_name) != -1){//해당 symbol이 중복일경우

printf("\"%s\"is duplicated symbol.\n",prog\_name);

return 0;

}

add\_estab(i,prog\_name,CSADDR);

while(fgets(str,100,fp[i]) !=NULL){

char recordType = str[0];

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

if(recordType == 'D'){

int rel\_addr;

char\* ptr;

char ext\_symbol[10], relAddr\_str[10];

for(ptr = str+1; \*(ptr) != '\0'; ptr += 12){

strncpy(ext\_symbol,ptr,6);

ext\_symbol[6]= '\0';

strncpy(relAddr\_str,ptr+6,6);

relAddr\_str[6]= '\0';

rel\_addr = strtol(relAddr\_str,NULL,16);

if(return\_estabAddr(ext\_symbol) != -1){

printf("\"%s\"is duplicated symbol.\n",ext\_symbol);

return 0;

}

add\_estab(i, ext\_symbol, rel\_addr + CSADDR);

}

}

else

continue;

}

CSADDR += CSLTH;

program\_Length[i] = CSLTH;

pLength += program\_Length[i];

}

return 1;

}

int loader\_pass2(FILE\* fp[], int file\_number){// loader 2pass중 pass2 에 해당하는 동작을 수행한다.

int CSADDR = 0;

int CSLTH;

char str[100];

char refName[100][10]={'\0',};

CSADDR = program\_address;

for(int i =0;i < file\_number;i++){

fgets(str,100,fp[i]);

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

while(fgets(str,100,fp[i]) !=NULL){ // 'T' record type

char recordType = str[0];

//printf("%c\n",recordType);

if(recordType == '.' || recordType == 'D')

continue;

if(recordType == 'T'){

char str\_addr[10];

char str\_len[10];

int text\_Addr, text\_len;

strncpy(str\_addr,str+1,6);

str\_addr[6] = '\0';

strncpy(str\_len, str+7,2);

str\_len[2] ='\0';

text\_Addr = strtol(str\_addr,NULL,16) + CSADDR;

text\_len = strtol(str\_len,NULL,16);

for(int j = 0; j < text\_len; j++){

char tmp[10];

strncpy(tmp,str + 9 + 2 \* j, 2);

tmp[2] = '\0';

memory[text\_Addr + j] = (unsigned char) strtol(tmp,NULL,16);

}

}

else if(recordType == 'M'){//'M' record type

char str\_modify\_address[10];

char str\_len[10];

char tmp[10];

int modify\_address;

int obj\_value;

strncpy(str\_modify\_address,str+1,6);

str\_modify\_address[6] = '\0';

strncpy(str\_len,str+7,2);

str\_len[2] = '\0';

modify\_address = strtol(str\_modify\_address,NULL,16)+CSADDR;

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

sprintf(tmp + j\*2, "%02X", memory[modify\_address + j]);

}

tmp[6] = '\0';

obj\_value = strtol(tmp,NULL,16);

if(tmp[0] >= '8' || tmp[0] < '0'){// signed bit == 1, 2의 보수

obj\_value = -(0xFFFFFF - obj\_value +1);

}

strncpy(tmp, str+10,2);

tmp[2] = '\0';

int ref\_idx;

int ref\_Addr;

char operator = str[9];

ref\_idx = atoi(tmp);

ref\_Addr = return\_estabAddr(refName[ref\_idx]);

if(ref\_Addr == -1){

printf("Symbol \"%s\"is not defind.\n", refName[ref\_idx]);

return 0;

}

if(operator == '+'){

obj\_value += ref\_Addr;

}

else if(operator == '-'){

obj\_value -= ref\_Addr;

}

if(obj\_value < 0){

obj\_value -= 0xFF000000;

}

char objValue[10];

sprintf(objValue,"%06X", obj\_value);

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

char tmp\_mem[10];

strncpy(tmp\_mem, objValue + j \* 2, 2);

tmp\_mem[2] = '\0';

memory[modify\_address+j] = (unsigned char) strtol(tmp\_mem,NULL,16);

}

}

else if(recordType == 'R'){ // 'R' record type

int ref\_idx;

char tmp[10];

strcpy(refName[1], estab[i][0].name);

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

strncpy(tmp, str+j,2);

tmp[2] = '\0';

ref\_idx = atoi(tmp);

strncpy(tmp, str + j + 2, 6);

tmp[6] = '\0';

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

if(tmp[k] == ' ' || tmp[k] == '\n'){

tmp[k] = '\0';

}

}

strcpy(refName[ref\_idx], tmp);

}

}

else if( recordType == 'E'){

if(str[1] != '\n'){

char tmp[10];

strncpy(tmp, str+1,6);

tmp[6] = '\0';

EXEADDR = CSADDR + strtol(tmp,NULL,16);

}

CSADDR += CSLTH;

}

}

}

return 1;

}

void print\_loadMap(int file\_num){// loader가 성공적으로 끝났다면 loadmap을 출력한다.

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

printf("section name\n");

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

for(int i = 0; i < file\_num;i++){

for(int j = 0; j < estab\_Max[i]; j++){

if(j == 0){

printf("%-7s %04X %04X\n",estab[i][j].name,estab[i][j].address,program\_Length[i]);

}

else{

printf(" %6s %04X\n",estab[i][j].name,estab[i][j].address);

}

}

}

int total\_length = 0;

for(int i = 0; i< file\_num;i++){

total\_length += program\_Length[i];

}

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

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

}

## Run\_command.h

#include "20161565.h"

void init\_reg(){// register를 초기화한다. PC는 프로그램 시작주소, L 은 프로그램 길이로 초기화한다.

store\_reg.A = 0x0;

store\_reg.X = 0x0;

store\_reg.L = 0x0;

store\_reg.B = 0x0;

store\_reg.T = 0x0;

store\_reg.F = 0x0;

store\_reg.F2 = 0x0;

store\_reg.PC = 0x0;

store\_reg.SW = 0x0;

for(int i=0;i<20;i++)

store\_reg.reg[i] =0;

store\_reg.PC = program\_address;

store\_reg.reg[8] = program\_address;

store\_reg.L = pLength;

store\_reg.reg[2] = pLength;

}

int cmd\_run(char input[100][100],int tok\_num){// 명령어 run에 대한 처리를 담당한다.

if(tok\_num != 1){

return -1;

}

if(loader\_flag == 1){

init\_reg();

loader\_flag = 0;

}

int prog\_endAddr = program\_address + pLength;

while(store\_reg.reg[8] < prog\_endAddr){

run\_instruction(memory[store\_reg.reg[8]]);

for(int i = 0; i < bp\_idx;i++){

if(store\_reg.reg[8] == bp[i]){

print\_Reg();

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

return 1;

}

}

}

print\_Reg();

printf("\t\tEnd Program\n");

return 1;

}

void run\_instruction(int opcode){// 한 instruction에 대한 연산을 수행한다.

int format = 0;

int check\_format = opcode/0x10;

switch(check\_format){

case 0xC:

case 0xF:

format = 1;

break;

case 0xA:

case 0xB:

format = 2;

break;

default:

format = 3;

if(memory[store\_reg.reg[8] + 1] & 0b10000)//if e bit is set

format = 4;

}

if(format == 1 || opcode == 0){//format 1 없음. 해당 memory가 비어있다면

store\_reg.reg[8]++;

return;

}

else if(format == 2){

int r1,r2;

r1 = memory[store\_reg.reg[8] + 1] / 0x10;

r2 = memory[store\_reg.reg[8] + 1] % 0x10;

switch(opcode){

case 0xB4: //CLEAR

store\_reg.reg[r1] = 0;

break;

case 0xA0: //COMPR

if(store\_reg.reg[r1] == store\_reg.reg[r2]){

store\_reg.reg[9] = '='; //reg[9] :SW

}

else if(store\_reg.reg[r1] > store\_reg.reg[r2]){

store\_reg.reg[9] = '>'; //reg[9] :SW

}

else{ //store\_reg.reg[r1] < store\_reg.reg[r2]

store\_reg.reg[9] = '<'; //reg[9] :SW

}

break;

case 0xB8://TIXR

store\_reg.reg[1]++; // X++;

if(store\_reg.reg[1] == store\_reg.reg[r1]){

store\_reg.reg[9] = '='; //reg[9] :SW

}

else if(store\_reg.reg[1] > store\_reg.reg[r1]){

store\_reg.reg[9] = '>'; //reg[9] :SW

}

else{ //store\_reg.reg[1] < store\_reg.reg[r2]

store\_reg.reg[9] = '<'; //reg[9] :SW

}

break;

}

store\_reg.reg[8] +=2; //PC += 2;

}

else{ //format 3 or 4

int nixbpe[6]= {0,0,0,0,0,0};

int addr = find\_AddrorConst(store\_reg.reg[8], format);

find\_nixbpe(nixbpe, store\_reg.reg[8]);

store\_reg.reg[8] += format;

if(format == 3 && (addr > (1<<11)) && (nixbpe[4] == 1)){//PC relative & signed bit 1

addr -= (1 << 12); //2's complement

}

if(nixbpe[2] == 1){ // x bit set

addr += store\_reg.reg[1];

}

if(nixbpe[3] == 1){ // b bit set

addr += store\_reg.reg[3];

}

if(nixbpe[4] == 1){ // p bit set

addr += store\_reg.reg[8];

}

if(nixbpe[0] == 1 && nixbpe[1] == 0){ //indirect aadressing

addr = indirect\_addr(addr, 3);

}

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

case 0x14: //STL

store\_memory(addr, store\_reg.reg[2], 3);// memory <- L

break;

case 0x68://LDB

if(nixbpe[0] == 0 && nixbpe[1] == 1){

store\_reg.reg[3] = addr; // B <- addr

}

else{

store\_reg.reg[3] = indirect\_addr(addr,3);

}

break;

case 0x48://+JSUB

store\_reg.reg[2] = store\_reg.reg[8];

store\_reg.reg[8] = addr;

break;

case 0x00://LDA

if(nixbpe[0] == 0 && nixbpe[1] == 1){

store\_reg.reg[0] = addr; // B <- addr

}

else{

store\_reg.reg[0] = indirect\_addr(addr,3);

}

break;

case 0x28://COMP

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr,3);

}

if(store\_reg.reg[0] == addr){

store\_reg.reg[9] = '=';

}

else if(store\_reg.reg[0] > addr){

store\_reg.reg[9] = '>';

}

else{ //store\_reg.reg[0] < addr

store\_reg.reg[9] = '<';

}

break;

case 0x30://JEQ

if(store\_reg.reg[9] == '='){

store\_reg.reg[8] = addr;

}

break;

case 0x3C://J

store\_reg.reg[8] = addr;

break;

case 0x0C://STA

store\_memory(addr,store\_reg.reg[0],3);

break;

case 0x74://LDT

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr, 3);

}

store\_reg.reg[5] = addr;

break;

case 0xE0://TD

//다음 instruction으로 넘어가되 CC 는 '<'로 변경되었다고 가정한다.

store\_reg.reg[9] = '<';

break;

case 0xD8://RD

//input device로부터 아무것도 받지 못했다고 가정.

// 다음 명령어인 COMPR A,S의 결과 CC 가 '=' 이라고 가정한다.

store\_reg.reg[0] = 0; // A = 0

store\_reg.reg[9] = '=';

break;

case 0x54://STCH

store\_memory(addr, store\_reg.reg[0] % 0X10, 1);

break;

case 0x38://JLT

if(store\_reg.reg[9] == '<'){

store\_reg.reg[8] = addr;

}

break;

case 0x10://STX

store\_memory(addr, store\_reg.reg[1], 3);

break;

case 0x4C://RSUB

store\_reg.reg[8] = store\_reg.reg[2];//PC = l

break;

case 0x50://LDCH

if(!(nixbpe[0] == 0 && nixbpe[1] == 1)){

addr = indirect\_addr(addr,3);

}

store\_reg.reg[0] =(store\_reg.reg[0] & 0xFFFFFF00) + (addr/0x10000);

break;

case 0xDC://WD

//다음 instruction으로 넘어간다.

break;

}

}

sync\_reg();//store\_reg의 register array와 각 register간의 동기화

}

void store\_memory(int addr, int value, int byte){ // value를 memory의 해당 address에 byte만큼 저장한다.

char tmp[10], word[3];

if(value < 0){

value -= 0xFF000000;

}

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

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

strncpy(word,tmp + i \* 2 ,2);

word[2] = '\0';

memory[addr+i] = strtol(word, NULL, 16);

}

};

int indirect\_addr(int addr, int byte){//n=0,i=1 일때 indircet address를 반환한다.

int indir\_addr = memory[addr];

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

indir\_addr \*= 0x100;

indir\_addr += memory[addr+i];

}

return indir\_addr;

};

void sync\_reg(){//store\_reg의 register array와 각 register간의 동기화

store\_reg.A = store\_reg.reg[0];

store\_reg.X = store\_reg.reg[1];

store\_reg.L = store\_reg.reg[2];

store\_reg.B = store\_reg.reg[3];

store\_reg.S = store\_reg.reg[4];

store\_reg.T = store\_reg.reg[5];

store\_reg.F = store\_reg.reg[6];

store\_reg.F2 = store\_reg.reg[7];

store\_reg.PC = store\_reg.reg[8];

store\_reg.SW = store\_reg.reg[9];

}

int find\_AddrorConst(int PC,int format){ // format에 따른 address or constant를 반환한다.

int addr = memory[PC + 1] % 0x10;

addr \*= 0x100;

addr += memory[PC + 2];

if(format == 4){//format == 4

addr \*= 0x100;

addr += memory[PC+3];

}

return addr;

}

void find\_nixbpe(int nixbpe[], int PC){// array nixbpe[6]에 순서대로 n,i,x,b,p,e에 해당하는 bit condition을 저장한다.

int ni = memory[PC] % 0b100;

int xbpe = memory[PC + 1] / 0b10000;

if(ni & 0b10){

nixbpe[0] = 1;

}

if(ni & 0b01){

nixbpe[1] = 1;

}

if(xbpe & (0b1000)){//x bit set

nixbpe[2] = 1;

}

if(xbpe & (0b0100)){//b bit set

nixbpe[3] = 1;

}

if(xbpe & (0b0010)){//p bit set

nixbpe[4] = 1;

}

if(xbpe & (0b0001)){//e bit set

nixbpe[5] = 1;

}

}

void print\_Reg(){//register들을 출력한다.

printf("\tA : %06X X : %06X\n",store\_reg.A, store\_reg.X);

printf("\tL : %06X PC : %06X\n",store\_reg.L, store\_reg.PC);

printf("\tB : %06X S : %06X\n",store\_reg.B, store\_reg.S);

printf("\tT : %06X\n",store\_reg.T);

}

int cmd\_bp(char input[100][100],int tok\_num){// 명령어 bp에 대한 연산을 수행한다.

if(tok\_num > 2){

return -1;

}

else if(tok\_num == 1){

print\_bp();

}

else{

int i;

if(strcmp(input[1],"clear") == 0){// bp clear

//clear bp

for(i = 0; bp[i] != -1; i++){

bp[i] = -1;

}

bp\_idx = 0;

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

}

else{ // bp [address]

int bp\_address = strtoHex(input[1]);

if(bp\_address < 0 || bp\_address > 0xFFFFF){

printf("Invalid break point address.\n");

return 0;

}

insert\_bp(bp\_address);

printf("\t\t[ok] create breakpoint %X\n",bp\_address);

}

}

return 1;

}

void insert\_bp(int addr){ //명령어 bp [address] : address를 bp\_address 배열에 저장 후 오름차순으로 sort하는 함수

if(bp\_idx >= 100){

printf("Number of breakpoint no more than 100.\n");

return;

}

bp[bp\_idx++] = addr;

if (bp\_idx == 1) return ;

for(int i = bp\_idx-1 ; i > 0;i--){

if(bp[i] < bp[i-1]){

int tmp = bp[i];

bp[i] = bp[i-1];

bp[i-1] = tmp;

}

}

};

void print\_bp(){ //명령어 bp : 모든 breakpoint를 출력하는 함수

printf("\t\tbreakpoint\n");

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

for(int i = 0; i < bp\_idx; i++){

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

}

};