과목 명: 시스템프로그래밍

담당 교수 명: 김 지 환

<<Assignment 3>>

**서강대학교 컴퓨터학과**

**[20161601]**

**[송인아]**

목 차

1. 프로그램 개요 4

2. 프로그램 설명 5

2.1 프로그램 흐름도 5

3. 모듈 정의 7

3.1 모듈 이름 : command\_bp() 7

3.1.1 기능 7

3.1.2 사용 변수 7

3.2 모듈 이름: command\_loader() 7

3.2.1 기능 7

3.2.2 사용 변수 8

3.3 모듈이름: int load\_obj() 8

3.3.1 기능 8

3.3.2 사용변수 9

3.4 모듈이름: void free\_estab() 9

3.4.1 기능 9

3.4.2 사용변수 9

3.5 모듈이름: int create\_estab() 10

3.5.1 기능 10

3.5.2 사용변수 10

3.6 모듈 이름 : int read\_estab() 10

3.1.1 기능 10

3.1.2 사용 변수 11

3.7 모듈 이름: void insert\_estab() 11

3.2.1 기능 11

3.2.2 사용 변수 11

3.8 모듈이름: void what\_is\_op\_code() 11

3.3.1 기능 11

3.3.2 사용변수 11

3.9 모듈이름: void run\_run\_run() 12

3.4.1 기능 12

3.4.2 사용변수 12

3.10 모듈이름: void command\_run() 13

3.5.1 기능 13

3.5.2 사용변수 13

3.11 모듈이름: void read\_opcode() 13

3.5.1 기능 13

3.5.2 사용변수 13

3.12 모듈이름: int make\_hex\_neg() 13

3.5.1 기능 13

3.5.2 사용변수 13

4. 전역 변수 정의 14

4.1 es\_node\* estab[3] 14

4.2 int load\_check 14

4.3 int reg[10] 14

4.4 int progaddr 14

4.5 int csaddr 14

4.6 int cslth 14

4.7 int exeaddr,exeaddr\_run 15

4.8 int prog\_end 15

4.9 int bp[CODE\_MAX] 15

4.2 int bp\_index 15

4.3 int cur\_reg 15

# 프로그램 개요

; 해당 Assignment의 프로그램에 대한 기능 및 특이점에 대한 전반적인 개요에 대해서 설명한다.

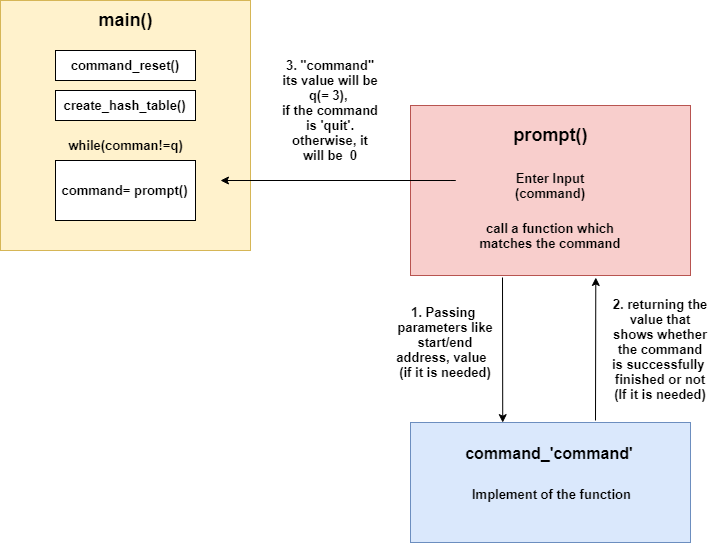
프로젝트 1,2로 진행했던 shell에 Linking과 Loading 을 추가 구현한 프로그램이다. link할 프로그램(obj file)을 입력받아 링크하고, 메모리에 obj 코드를 올려준다. 또한Load한 프로그램을 run 명령어를 통해 실제로 실행하고 연산 수행 결과를 출력해준다. 그리고 bp를 설정하여 디버깅 할 수 있는 기능도 수행한다.

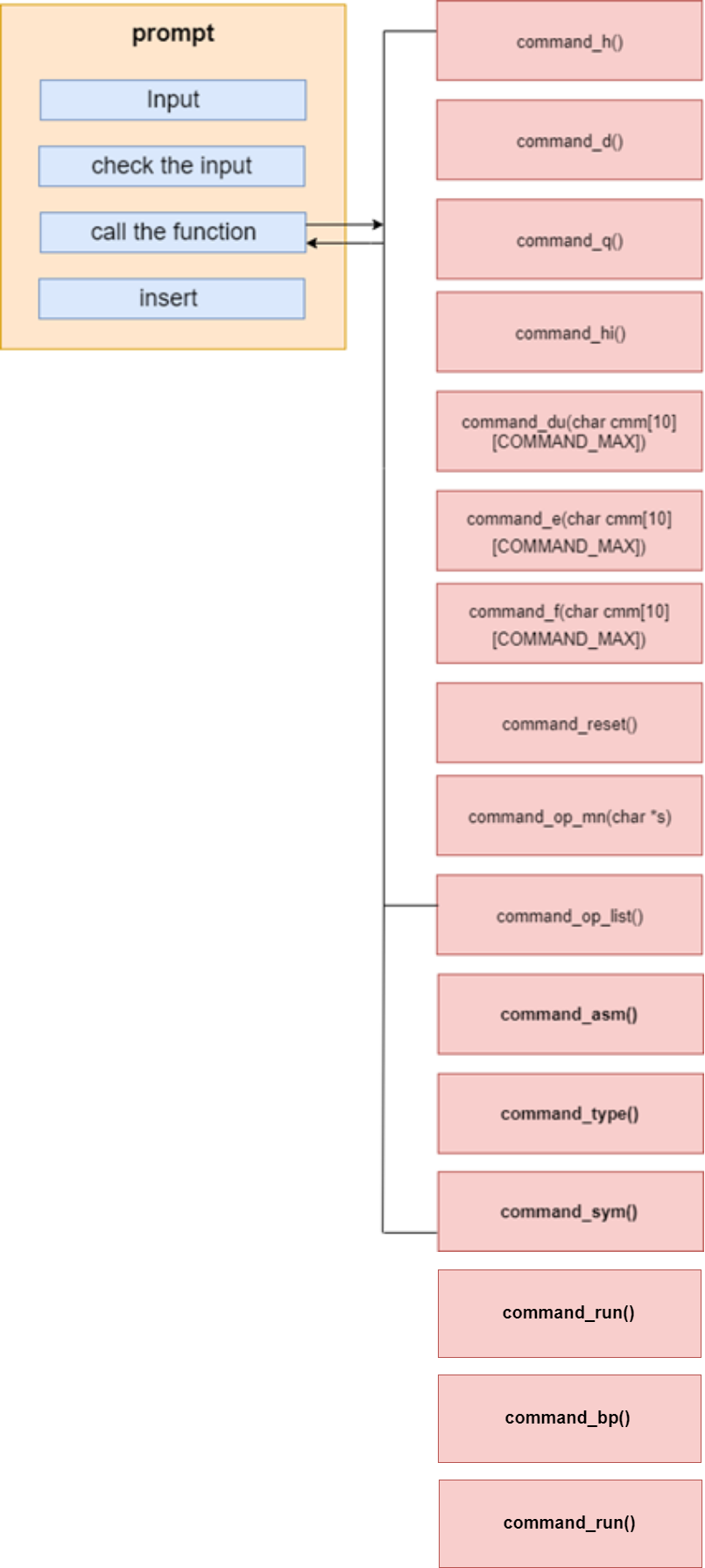
< 추가 구현 기능 >

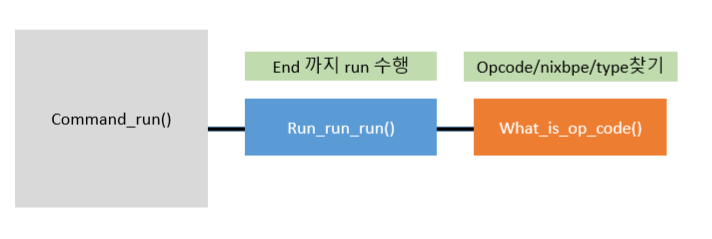
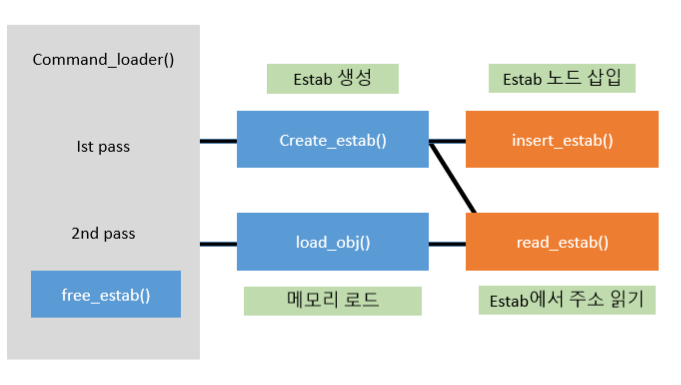
1. help = 추가된 명령어를 추가해준다.
2. prompt = 추가된 명령어를 호출해준다.
3. run = 메모리에 로드된 프로그램을 수행한다
4. bp = 입력받은 break point를 저장한다
5. linking loader = 입력받은 프로그램을 링크하고 로드한다.

# 프로그램 설명

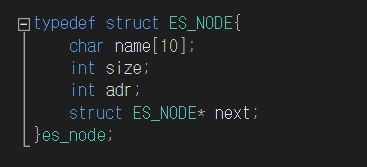
## 프로그램 흐름도







# 모듈 정의



* External Symbol Table 저장하는 구조체

## 모듈 이름 : void command\_bp(int adr)

### 기능

bp를 설정해주는 함수이다. adr이 -1인 경우 세팅된 bp를 출력해주고, adr이 -2인 경우에는 bp를 모두 초기화해준다. bp가 -1,-2가 아닌 경우에는 해당 값을 bp[bp\_index]에 저장하고, bp\_index++을 수행한다. 이 때 저장된 bp값들이 모두 오름차순이 되도록 정렬하여 저장한다.

### 사용 변수

int i,j : loop counter

## 모듈 이름 : int command\_loader(int num, char file[10][COMMAND\_MAX])

### 기능

loader 명령어 수행시 호출되는 함수로, 사용자로부터 입력받은 파일들을 Linking 하고 Loading 해주는 기능을 수행한다. 최대 파일 개수는 3개이다. num 에는 파일의 개수가, file 배열에는 각각 파일 이름이 저장되어 있다.

1. 파일 확인 : 해당 파일들은 command\_loader 가 호출되기 전에 prompt 함수에서 ‘.obj’ 파일임을 확인하고 넘어온 것이기 때문에 해당 파일이 존재하는지만 체크해준다. 파일일 존재하지 않는 경우 에러 메시지 출력 후 함수를 종료한다.
2. 1st pass : csaddr을 progaddr로 설정해주고 입력 받은 파일들을 하나씩 읽어가며 External Symbol table을 작성한다. 해당 기능은 create\_estab함수를 통해 수행된다.
3. 2nd pass : csaddr과 exeaddr을 progaddr로 설정해주고, 작성된 ESTAB을 바탕으로 파일을 한번씩 더 읽어 메모리에 Loading한다. 이는 load\_obj 함수를 통해 수행된다. 파일을 한 번씩 읽을 때마다 csaddr에 cslth를 더해주어 csaddr을 갱신해준다. 모든 파일을 읽었으면 prog\_end를 csaddr로 설정해준다. (마지막 csaddr은 progaddr+ 로드한 프로그램 총 사이즈)
4. Loading이 정상적으로 완료된 경우에는 load\_check를 1로 세팅해주고, 열었던 파일을 닫아준다.

### 사용 변수

* FILE\* f[3] : 입력받은 파일들을 읽기 위한 파일 포인터 배열
* int i,j : loop counter
* int result : Load의 성공여부를 저장하는 변수 (0= 실패, 1= 성공)

## 모듈 이름 : int load\_obj(FILE\* fp, int f\_index)

### 기능

2nd pass에서 obj 파일의 내용들을 메모리에 올리는 기능을 수행하는 함수이다. fp로 넘겨받은 파일을 한 줄 한 줄 읽으며 Loading을 수행한다.

1. D record = 무시
2. R record

* ref배열에는reference number를 인덱스로, 해당 external symbol의 주소값이 저장되어 있다.
* R record에 저장되어 있는 external symbol과 reference number를 strncpy로 받아와서 그 내용을ref 배열에 저장한다.

1. H record

* head record에서 프로그램 총 사이즈를 받아와서 cslth 값을 저장한다.

1. T record

* T record에서 시작 주소와, 사이즈를 읽어와서 메모리에 해당 text를 올린다.
* t\_adr에 text record 시작 주소를 저장하고, t\_size에 총 사이즈를 저장한다.
* t\_adr을 바탕으로 row, col을 계산하고 t\_size\*2 개의 digit만큼 메모리에 올린다.

1. M record

* 메모리 수정 주소와 길이를 받아와서 start\_address 와 length 를 세팅해준다.
* t\_adr, t\_size 에 각각 start\_address 와 length 의 int 변환값을 저장해준다.
* 이를 바탕으로 row/col t\_size가 홀수이면 col++을 해준다/
* M record에서 수정해줄 주소와 ‘+’/’-‘를 받아서 주소는 m\_adr에 저장하고, ‘-‘이면 m\_adr에 -1을 곱해준다.
* 수정할 주소의 obj 코드를 받아와서 int로 변환하고 m\_adr을 더해준 값을 다시 메모리에 올린다.

1. E record

* exeaddr이 있는 경우 이를 저장한다.

pc를 exeaddr로 설정해준다.

### 사용 변수

* int ref[LINE\_MAX] : external reference 주소값 저장해주는 배열/ ref[1]에는 해당 프로그램 시작 주소가 저장되어 있다.
* char c[LINE\_MAX] : 파일에서 한 줄씩 읽어오기 위한char 배열
* int obj : obj code의 int형 변환한 값이 저장된 변수
* int t\_adr: 메모리 시작 주소
* int m\_adr: modification 해야할 주소를 저장하고 있는 변수
* int i : loop counter
* int row, col : 주소를 받아와서 memory row/col을 계산하여 저장한다.

## 모듈 이름 : void free\_estab()

### 기능

ESTAB을 free시켜주는 함수이다. estab[3] 내용을 하나 하나 free 시켜준다.

### 사용 변수

* int i : loop counter
* es\_node\* tmp, \*tmp2 : 노드 하나하나 접근하기 위한 es\_node 포인터 변수

## 모듈 이름 : int create\_estab(FILE\* fp, int f\_index)

### 기능

Loading 1st pass 때 수행되는 함수로, 입력받은 파일을 한 줄 한 줄 읽어가며 H,D record에 있는 내용을 바탕으로 estab을 작성하는 기능을 수행한다.

1. H record: 해당 프로그램의(Control Section) 이름과, 시작주소, 길이(cslth)를 저장한다. strncpy 함수를 통해 obj 파일 형식에 맞춰 각각의 정보들을 얻어온다. 이후 estab에 중복된 이름의 control section이 존재하는지 확인한 뒤, 중복된 경우라면 에러메시지 출력 후 함수를 종료한다. 그렇지 않은 경우 insert\_estab() 함수를 통해 해당 정보를 estab에 추가한다.
2. D record: 마찬가지로 strncpy 함수를 통해 D record에 정의된 symbol들의 이름과 주소를 받아온다. 이후 해당 symbol이 이미 존재하는지 확인한 뒤, 중복된 경우라면 에러메시지를 출력 후 함수를 종료하고, 그렇지 않은 경우에는 insert\_estab() 함수로 해당 정보를 ESTAB에 저장한다.
3. 1~2를 모두 수행한 경우, csaddr 에 cslth 를 더해주어 다음 프로그램이 로드될 시작 주소를 계산한다.

### 사용 변수

* char name[10] : 프로그램 이름을 저장하기 위한 char배열
* char c[LINE\_MAX] : fgets로 파일에서 한 줄 읽어오기 위한 char 배열
* int adr, size / char tmp[LINE\_MAX] : estab에 넣을 정보를 저장하는 변수들

## 모듈 이름 : int read\_estab(char\* name)

### 기능

estab에 name이라는 이름의 symbol이 있으면 그 symbol의 주소를 반환해주는 함수이다. 해당 이름의 symbol이 없는 경우는 return -1을 한다.

### 사용 변수

* int i : loop counter
* es\_node\* tmp : estab의 노드들에 접근하기 위한 포인터

## 모듈 이름 : void insert\_estab(int f\_index, char\* name, int size, int adr)

### 기능

해당 f\_index를 key로 하는 estab에 naem, size, adr을 저장하고 있는 노드를 추가해주는 함수이다. 새로운 노드를 할당하여 넘겨받은 정보들을 채워준 뒤, 경우에 맞게 해당 노드를 estab에 연결한다.

### 사용 변수

* es\_node\* node : 새로 estab에 연결할 노드
* es\_node \* tmp : node를 연결하기 위해 사용하는 es\_node 포인터

## 모듈 이름 : void what\_is\_op\_code(char\* obj, int\* op, int\* n,int\* i, int\* x,int\* b, int\* p, int\* e, int\* type)

### 기능

메모리에 있는 opj code 3-digit(char\* obj)을 받아와서 opcode, n,i,x,b,p,e 와 type을 세팅해주는 함수이다. obj 를 make\_hex함수로 int로 변환한 뒤, 비트 연산자를 사용하여 각각 n,i,x,b,p,e비트값을 세팅해준다. type은 read\_opcode 함수를 통해 세팅한다.

### 사용 변수

int tmp, tmp2 = obj 내용을 int형으로 바꾼 값을 저장하는 변수

## 모듈 이름 : void run\_run\_run(int end)

### 기능

end를 bp로하여 run을 수행하는 함수이다.

1. pc가 prog\_end보다 작을 때까지 메모리를 이동한다.
2. 현재 pc에 대한 row/col을 계산한다.
3. pc부터 메모리의 3-digit(12bit)의 내용을 받아와서 what\_is\_op\_code() 함수에 넘긴다.
4. 이후 계산된 opcode,n,i,b,p,e와 type을 가지고 해당 instruction과 target\_adr을 구한다.
5. pc<=end && end<pc+type 인지 확인하고 이 조건이 맞지 않는 경우 break하고 그렇지 않은 경우pc += type을 하여 다음 instruction을 가리키게 한다.
6. format 이 2인 경우 reg1 과 reg2를 설정해주고 필요한 instruction을 opcode로 구별하여 수행한 뒤, continue한다
7. format이 3,4인 경우 opj code 뒷부분(앞의 3digit을 제외한 부분)을 가지고 target\_adr과 operand, imm을 구한다. 이 때, ni 비트에 따라 필요한 값들을 부분적으로 계산한다.
8. 이후 opcode 를 가지고 연산을 구별하고 그에 맞는 기능을 수행한다.

루프가 끝나면 출력이 필요한 레지스터를 출력해준다

### 사용 변수

* int rg1,rg2 : format 2 instruction수행시 사용되는 레지스터 넘버
* int target\_adr : 매개변수로 받은 주소의 실제 주소값
* int operand : target\_adr의 메모리에 저장된 값
* int op,n,i,x,b,p,e : obj 코드의 opcode/n/i/b/p/e 비트
* int type : 해당 instruction의 type
* int row/col/ina/song : memory 배열 접근을 위한 변수
* int imm: immediate mode에서의 operand

## 모듈 이름 : void command\_run()

### 기능

### run 명령어 수행시 호출되는 함수로, load\_check=0이면 아직 로드된 프로그램이 없는 것이므로 에러메시지 출력 후 종료한다.

### load\_check=1인 경우 cur\_bp 인덱스에 있는 bp를 매개변수로 run\_run\_run() 호출하여 run을 수행한다.

### 사용 변수

* 없음

## 모듈 이름 : void read\_opcode(int n, int\* type)

### 기능

OPTAB에서 opcode가 n인 명령어를 찾아 type을 세팅해주는 함수이다.

### 사용 변수

* int i : loop counter
* op\* tmp : optab 노드들에 접근하기 위한 op\* 변수

## 모듈 이름 : int make\_hex\_neg(char\* s)

### 기능

### s 문자열에 저장된 hex를 int 형으로 반환해주되, 음수인 경우 음수로 반환해주는 함수이다. make\_hex() 함수를 사용하여 MSB가 8~9 or A~F 인 경우 음수로 처리해준다.

### 사용 변수

* int i : loop counter
* int tmp : make\_hex(s) 값을 저장하는 변수

# 전역 변수 정의

## es\_node\* estab[3]

External Symbol Table을 위한 헤드포인트이다. hash table size는 3이다.

## int load\_check

run 명령어 수행시, 메모리에 로드된 프로그램이 있는지 확인하기 위한 변수이다. 값이 0이면 메모리에 로드된 파일이 없는 것이고, 1이면 로드된 프로그램이 있다는 것이다.

## int reg[10]

Register 9개를 담고 있는 배열이다. 각각 Register number를 인덱스로 접근한다

## int progaddr

Load할 프로그램이 올라갈 첫 주소이다. progaddr 명령어를 통해 수정이 가능하다.

## int csaddr

Control Section의 시작 주소를 저장하는 변수이다.

## int cslth

Control Section의 총 사이즈를 저장하는 변수이다.

## int exeaddr, exeaddr\_run

Run 명령어로 프로그램 수행시 처음으로 실행되는 instruction이 저장된 주소이다. 이 주소부터 run을 시작한다.

## int prog\_end

Load된 프로그램의 마지막 주소를 저장하는 변수이다. Run 수행 시, 최대 이 주소까지 수행한다.

## int bp[CODE\_MAX]

Break Point를 입력받아 저장하는 배열이다.

## int bp\_index

bp의 개수를 저장한다.

## int cur\_bp

마지막으로 run이 수행된 bp의 index(bp 배열) 를 저장한다.

# 코드

**<20161601.h>**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/stat.h>

//명령어글자 최대 길이

#define COMMAND\_MAX 256

#define MEMORY\_ROW 0xfffff

#define MEMORY\_COL 35

#define HASH\_MAX 20

#define LINE\_MAX 256

#define CODE\_MAX 0xfffff

//예외처리

#define WRONG\_ADR 1

#define WRONG\_VAL 2

#define WRONG\_PARAMETER 3

#define WRONG\_CMM 4

#define INPUT\_SIZE\_OVER 5

#define WRONG\_FILE 6

#define ASSEMBLE\_FAIL 7

#define NO\_SYMBOL 8

//명령어 num

#define q 3

#define VAR 9

#define CONST 10

#define BASE 11

#define DIREC 12

#define COMMENT 13

#define A 0

#define X 1

#define L 2

#define B 3

#define S 4

#define T 5

#define F 6

#define PC 8

#define SW 9

//Linked List

typedef struct LinkedList {

int cnt;

struct Node\* front;

struct Node\* rear;

}ll;

typedef struct Node {

char command[COMMAND\_MAX];

struct Node\* next;

}node;

typedef struct OP {

int op\_num;

char op\_name[20];

char op\_type[10];

struct OP\* next;

}op;

typedef struct SYM{

int adr;

char name[30];

struct SYM\* next;

}sym;

typedef struct CODE{

int adr;

int num;

char inst[20];

char operand[30];

char operand2[30];

int type;

char obj[100];

}code;

typedef struct ES\_NODE{

char name[10];

int size;

int adr;

struct ES\_NODE\* next;

}es\_node;

void command\_h();

void command\_d();

void command\_q();

void command\_hi();

int command\_du(char cmm[10][COMMAND\_MAX],int cnt);

int command\_e(char cmm[10][COMMAND\_MAX]);

int command\_f(char cmm[10][COMMAND\_MAX]);

void command\_reset();

int command\_op\_mn(char\* s);

void command\_op\_list();

int hash\_key(char\* s);

void create\_hash\_table();

//assemble

int command\_file(char\* name);

int command\_asm(char\* name);

int command\_sym();

int create\_lst(FILE\* fp,char\* inst, int line);

int create\_obj(FILE\* fp,char\* prog\_name);

int insert\_code(char\* inst, char\* operand, int line);

int store\_symbol(int adr, int size, char\* s, int line);

int comp\_symbol(char \*name, char\* tmp);

int insert\_symbol(char\* name, int adr);

int read\_symbol(char\* name);

void free\_symbol();

int read\_reg(char\* s);

void print\_estab();

//linking loader

void command\_bp(int adr);

int command\_loader(int num, char file[10][COMMAND\_MAX]);

int load\_obj(FILE\* fp, int f\_index);

void free\_estab();

int create\_estab(FILE\* fp, int f\_index);

int read\_estab(char\* name);

void insert\_estab(int f\_index, char\* name, int size, int adr);

void what\_is\_op\_code(char\* obj, int\* op, int\* n, int\* i,int\* x, int\* b, int\* p , int\* e, int\* type);

void run\_run\_run(int end);

void command\_run();

void read\_opcode(int n, int\* type);

int make\_hex\_neg(char\* s);

int prompt();

void print\_error(int error);

void insert(char cmm[COMMAND\_MAX]);

char\* remove\_space(char\* s);

int one\_digit\_to\_int(char c);

int make\_hex(char\* s);

char make\_char(int n);

int make\_int(char\* s);

**<20161601.c>**

#include "20161601.h"

op\* oplist[HASH\_MAX]; //opcode hash table을 위한 linked list

ll\* hlist=NULL; //history 저장 linked list

sym\* symtable[26]; //symbol table

code code\_set[CODE\_MAX];

es\_node\* estab[3]; //estab (최대 파일 개수 3)

int RUN\_FLAG;

int memory[MEMORY\_ROW][MEMORY\_COL];

int lc[MEMORY\_ROW];

int cur\_adr=0; // dump 수행시 시작 메모리 번지

int start\_check=0, end\_check=0, load\_check=0;

int a=0,t=0,f=0,l=0,sw=0,pc=0,base=0,x=0,s=0,nobase=1; // 레지스터

int reg[10];

int last\_line=0,total\_size=0, code\_max=0; //코드 라인 개수, 총 코드 사이즈, 코드 한 줄 최대 길이

int progaddr=0, csaddr=0 , cslth=0, exeaddr=0,exeaddr\_run=0, prog\_end=0;

int bp[CODE\_MAX], bp\_index=0, cur\_bp=0;

int main() {

int command = 0;// 명령어

command\_reset(); //memory 초기화

create\_hash\_table(); //hash table 생성

while (command != q) {

command = prompt(); //q 명령어 입력 전까지 prompt 수행

}

return 0;

}

//명령어 입력 받고, 해당 명령어에 해당하는 함수 호출quit 명령어를 받았을 때만 q(3) return 하고 , 나머지는 return 0

int prompt(){

int i=0,flag=0, j, cut=0;

char command\_words[10][COMMAND\_MAX]; // (공백 제거된)명령어를 이루는 문자열 저장 배열

char command[COMMAND\_MAX]; // 입력받은 실제 명렁어

char cmm[COMMAND\_MAX];

char \*tmp =NULL,ch=-1, tmp\_arr[COMMAND\_MAX];

//초기화

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

command\_words[i][0]='\0';

}

i=0;

printf("sicsim> ");

// 명령어 입력 받음(공백 포함)

while(1){

scanf("%c", &ch);

if(ch=='\n')

break;

command[i++]=ch;

//input 최대 글자수 초과인 경우

if(i==COMMAND\_MAX){

print\_error(INPUT\_SIZE\_OVER);

flag=1;

}

}

//input size 초과인 경우 종료

if(flag==1)

return 0;

command[i]='\0';

strcpy(command, remove\_space(command));

i=0; // 여기서부터 i = 명령어에 포함된 문자열 덩어리 개수

strcpy(cmm,command);

tmp= strtok(cmm," \r\n\t"); // ' ' 기준으로 자른 문자열

strcpy(command\_words[i++],tmp);

if( strcmp(command\_words[0], "loader")==0)

cut= 1;

while(i<6){

if( cut==1)

tmp= strtok(NULL, " \r\t\n"); //공백 단위로 매개변수 구분하는 경우(loader)

else

tmp = strtok(NULL,","); // ',' 단위로 자른 문자열

if(tmp==NULL)

break;

tmp=remove\_space(tmp);

strcpy(command\_words[i++],tmp);

}

//잘못된 명령어(i 의 최대값 = 4)

if(i>4){

print\_error(WRONG\_CMM);

return 0;

}

//해당 명령어에 해당하는 함수 수행

if( strcmp(command, "q") ==0 || strcmp(command,"quit")==0){

command\_q();

return q;

}

if( strcmp(command, "h") ==0 || strcmp(command,"help")==0)

command\_h();

else if( strcmp(command,"d") ==0 || strcmp(command,"dir")==0)

command\_d();

else if( strcmp(command\_words[0],"du")==0 || strcmp(command\_words[0],"dump")==0){

if(i>3){ //너무 많은 parameter 받은 경우

print\_error(WRONG\_PARAMETER);

return 0;

}

if(!command\_du(command\_words, i)) // 예외 발생 시 종료( hisory 기록 안됨)

return 0;

}

else if(strcmp(command\_words[0],"e")==0 || strcmp(command\_words[0],"edit")==0){

//잘못된 형식으로 입력한 경우 예외처리 후 함수 종료

if(i!=3){

print\_error(WRONG\_PARAMETER);

return 0;

}

if(!command\_e(command\_words))

return 0;

}

else if(strcmp(command\_words[0],"f")==0 || strcmp(command\_words[0],"fill")==0){

if(i!=4){

print\_error(WRONG\_PARAMETER);

return 0;

}

if(!command\_f(command\_words))

return 0;

}

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

command\_reset();

else if( strcmp(command\_words[0], "opcode")==0){

if(i!=2){

print\_error(WRONG\_PARAMETER); //너무 많은 mnemonic을 받았거나, 띄어쓰기가 포함된 경우

return 0;

}

if(!command\_op\_mn(command\_words[1])) //해당op\_mn이 없는 경우

return 0;

}

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

command\_op\_list();

else if(strcmp(command\_words[0],"type")==0){

//파일 이름 잘못 입력한 경우

if(i!=2){

print\_error(WRONG\_PARAMETER);

return 0;

}

//해당 이름의 파일이 없는 경우 history에는 저장

command\_file(command\_words[1]);

}

else if(strcmp(command\_words[0], "assemble")==0){

//파일 이름 잘못 입력한 경우

if(i!=2){

print\_error(WRONG\_FILE);

return 0;

}

//파일 이름 확인하기 (.asm 파일인지/ 올바른 형식인지)

tmp= strtok(command\_words[1], ".");

if( tmp ==NULL){

print\_error(WRONG\_FILE);

return 0;

}

//파일이름 ('.' 앞 문자열) 저장하고 tmp에 확장자명 저장

strcpy(command\_words[1],tmp);

tmp = strtok(NULL," \r\n\t");

//확장자명이 asm 이 아니거나 파일 입력을 잘못한 경우

if(strcmp(tmp, "asm")!=0){

print\_error(WRONG\_FILE);

return 0;

}

//어셈블이 실패한 경우 (history 에는 저장x)

if(!command\_asm(command\_words[1])){

free\_symbol();

return 0;

}

}

else if(strcmp(command, "symbol")==0){

//잘못된 입력

if( i!=1){

print\_error(WRONG\_CMM);

return 0;

}

if(!command\_sym()){

print\_error(NO\_SYMBOL);

}

}

else if( strcmp(command\_words[0], "progaddr")==0){

//잘못된 입력

if( i!=2){

print\_error(WRONG\_CMM);

return 0;

}

if( make\_hex(command\_words[1])==-1){

print\_error(WRONG\_PARAMETER);

return 0;

}

progaddr= make\_hex(command\_words[1]);

}

else if( strcmp(command\_words[0], "loader")==0){

//파일 입력 안한경우

if( i==1){

print\_error(WRONG\_PARAMETER);

return 0;

}

//파일 이름 확인하기 (.obj 파일인지/ 올바른 형식인지)

for(j=1; j<i; j++){

strcpy(tmp\_arr, command\_words[j]);

tmp= strtok(tmp\_arr, ".");

if( tmp ==NULL){

print\_error(WRONG\_FILE);

return 0;

}

//tmp에 확장자명 저장

tmp = strtok(NULL," \r\n\t");

//확장자명이 obj 이 아니거나 파일 입력을 잘못한 경우

if(strcmp(tmp, "obj")!=0){

print\_error(WRONG\_FILE);

return 0;

}

}

if(!command\_loader(i-1, command\_words))

return 0;

}

else if(!strcmp(command,"run")){

command\_run();

}

else if( strcmp(command\_words[0],"bp") ==0){

//bp 출력

if( i==1)

command\_bp(-1);

//잘못된 형식

else if(i>2){

printf("Wrong Command\n");

return 0;

}

//bp clear

else if(strcmp(command\_words[1],"clear")==0)

command\_bp(-2);

// else if( make\_hex(command\_words[1])==-1 || make\_hex(command\_words[1])>cslth)

// return 0;

else

command\_bp(make\_hex(command\_words[1]));

}

//입력 받은 명령어가 history도 아니면, 정의되지 않은 명령어 -> 예외처리

else if( strcmp(command, "hi") !=0 && strcmp(command,"history")!=0){

print\_error(WRONG\_CMM);

return 0;

}

//예외 없이 잘 수행한 명령어들은 hlist에 추가

insert(command);

if( strcmp(command, "hi") ==0 || strcmp(command,"history")==0)

command\_hi();

return 0;

}

char\* remove\_space(char\* s) {

int i;

if (s == NULL)

return s;

//앞쪽 공백 제거

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

if (s[i] != ' ' && s[i] != '\t' && s[i] != '\n')

break;

}

s += i;

//뒤쪽 공백 제거

for (i = strlen(s); i > 0; i--) {

if (s[i] != '\0' && s[i] != ' ' && s[i] != '\t' && s[i] != '\n')

break;

s[i] = '\0';

}

return s;

}

void print\_error(int error) {

if (error == WRONG\_ADR)

printf("Wrong address\n");

else if (error == WRONG\_VAL)

printf("Wrong Value\n");

else if (error == WRONG\_PARAMETER)

printf("Wrong parameter\n");

else if (error == WRONG\_CMM)

printf("Wrong Command\n");

else if (error == INPUT\_SIZE\_OVER)

printf("Input size is over\n");

else if(error==WRONG\_FILE)

printf("Wrong file name\n");

else if(error==ASSEMBLE\_FAIL)

printf("Assemble is failed\n");

else if(error==NO\_SYMBOL)

printf("Symbol table does not exist\n");

}

void insert(char cmm[COMMAND\_MAX]) {

//새로운 노드 할당

node\* newNode = (node\*)malloc(sizeof(node));

newNode->next = NULL;

strcpy(newNode->command, cmm);

//처음으로 노드 추가하는 경우

if (hlist == NULL) {

hlist = (ll\*)malloc(sizeof(ll));

hlist->front = hlist->rear = newNode;

hlist->cnt = 0;

}

else {

(hlist->rear)->next = newNode;

hlist->rear = newNode;

}

(hlist->cnt)++;

}

//한 digit(char) -> int

int one\_digit\_to\_int(char c){

int result=0;

if(c>='A' && c<='F')

result= c-'A'+10;

else if(c>='a' && c<='f')

result=c-'a'+10;

else if(c>='0' && c<='9')

result=c-'0';

//16진수 형식에 맞지 않으면 -16 return

else

result=-1;

return result;

}

int make\_hex(char\* s) {

int i, result = 0;

char c;

//앞에 0x(0X) 붙여준 경우그부분 짤라줌

if (s[0] == '0' && (s[1] == 'x' || s[1] == 'X'))

s += 2;

//한 digit씩 체크

for (i = 0; s[i]; i++) {

c = s[i];

result \*= 16;

if (c >= 'A' && c <= 'F')

result += c - 'A' + 10;

else if (c >= 'a' && c <= 'f')

result += c - 'a' + 10;

else if (c >= '0' && c <= '9')

result += c - '0';

//16진수 형식에 맞지 않으면 -1 return

else

return -1;

}

return result;

}

int make\_int(char\* s){

int i, result=0;

for(i=0; s[i]; i++){

result\*=10;

if(s[i]<'0' || s[i]>'9')

return -1;

result+=s[i]-'0';

}

return result;

}

char make\_char(int n) {

char c;

if (n<0 || n >= 16)

return '\0';

if (n<10)

c = n + '0';

else

c = (n - 10) + 'A';

return c;

}

//hash key 계산 ( 문자열 아스키 코드 값의 합)

int hash\_key(char\* s) {

int i, key = 0;

for (i = 0; i<strlen(s); i++)

key += s[i] - '0';

return key%HASH\_MAX;

}

void create\_hash\_table() {

int key, op\_num;

char op\_name[20], op\_type[10];

FILE\* fp = fopen("opcode.txt", "r");

op\* opcode = NULL;

//파일이 없는 경우

if (fp == NULL) {

printf("'opcode.txt' dose not exist\n");

return;

}

while (fscanf(fp, "%x %s %s", &op\_num, op\_name, op\_type) != EOF) {

//새로운 op node 생성

opcode = (op\*)malloc(sizeof(op));

opcode->op\_num = op\_num;

strcpy(opcode->op\_name, op\_name);

strcpy(opcode->op\_type, op\_type);

opcode->next = NULL;

key = hash\_key(op\_name);

//해당 위치에 처음으로 추가하는 경우

if (oplist[key] == NULL) {

oplist[key] = opcode;

}

else {

opcode->next = oplist[key];

oplist[key] = opcode;

}

}

}

//help 명령어 수행 함수

void command\_h() {

puts("h[elp]");

puts("d[ir]");

puts("q[uit]");

puts("hi[story]");

puts("du[mp] [start, end]");

puts("e[dit] address, value");

puts("f[ill] start, end, value");

puts("reset");

puts("opcode mnemonic");

puts("opcodelist");

puts("assemble filename");

puts("type filename");

puts("symbol");

puts("progaddr");

puts("loader");

puts("run");

puts("bp");

}

void command\_d() {

struct dirent\* file;

struct stat type;

DIR \*dir = opendir(".");

while (1) {

file = readdir(dir);

if (file == NULL)

break;

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

if (S\_ISDIR(type.st\_mode)) {

//. & .. 은 무시하고 넘어감

if (strcmp(file->d\_name, ".") == 0 || strcmp(file->d\_name, "..") == 0)

continue;

//directory 일 경우 /

printf("%s/\n", file->d\_name);

}

//실행파일의 경우 \*

else if (type.st\_mode & S\_IXUSR)

printf("%s\*\n", file->d\_name);

else

printf("%s\n", file->d\_name);

}

closedir(dir);

}

void command\_q() {

int i;

node\* tmp = NULL;

node\* tmp2 = NULL;

op\* t = NULL;

op\* t2 = NULL;

//hlist free

if (hlist != NULL) {

for (tmp = hlist->front; tmp != NULL;) {

tmp2 = tmp;

tmp = tmp->next;

free(tmp2);

}

free(hlist);

}

//oplist(hash\_table) free

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

for (t = oplist[i]; t != NULL;) {

t2 = t;

t=t->next;

free(t2);

}

}

free\_symbol();

}

void command\_hi() {

int i;

node\* tmp;

if (hlist == NULL) // 빈 리스트의 경우

return;

tmp = hlist->front;

for (i = 0; i<hlist->cnt; i++) {

printf("%d %s\n", i + 1, tmp->command);

tmp = tmp->next;

}

}

int command\_du(char cmm[10][COMMAND\_MAX], int cnt){

int i,j,tmp;

int s=0,end=0; // s: start, e: end

int srow=-1,scol=-1, ecol=-1, erow=-1; // 출력 row/col 범위

if( cnt>=2)

s= make\_hex(cmm[1]); //start 입력 받음

if(cnt ==3)

end= make\_hex(cmm[2]);//end 입력 받음

//입력 주소 체스 ( 입력받은 start/end가16진수 형식이 아니거나 범위 넘어간 경우 or s>e

if(s==-1||end==-1||s>0xfffff || end>0xfffff||( cnt==3 && s>end))//start > end 인 경우

{

print\_error(WRONG\_ADR);

return 0;

}

//start 안받은 경우

if(cnt<2){

s=cur\_adr;

if(s>0xfffff)

s=0;

}

//end 안 받은 경우

if(cnt<3){

end= s+159;

if(end>0xfffff)

end=0xfffff;

}

cur\_adr=end+1;

srow=s/16;

scol=(s%16)\*2;

erow=end/16;

ecol=(end%16)\*2;

for(i=srow; i<=erow; i++){

printf("%05X ", i\*16);

for(j=0; j<32; j+=2){

if((i==srow && j<scol)||(i==erow && j>ecol)) //범위아닌 메모리 값은 출력 x

printf(" ");

else

printf("%02X ", one\_digit\_to\_int(memory[i][j])\*16+ one\_digit\_to\_int(memory[i][j+1]));

}

printf("; ");

for(j=0; j<32; j+=2){

if((i==srow && j<scol)||(i==erow && j>ecol)) //범위아닌 메모리 값은 출력 x

printf(".");

else{

tmp=one\_digit\_to\_int(memory[i][j])\*16 + one\_digit\_to\_int(memory[i][j+1]); // 메모리에 저장된 값

if( tmp>=0X20 && tmp <=0X7E) // 20~7E 사이 값이면 해당 값의 아스키코드 출력

printf("%c",tmp);

else

printf(".");

}

}

printf("\n");

}

return 1;

}

int command\_e(char cmm[10][COMMAND\_MAX]){

int adr,val;

adr= make\_hex(cmm[1]);

val= make\_hex(cmm[2]);

if(adr<0 || adr >0Xfffff){

print\_error(WRONG\_ADR);

return 0;

}

if(val>0xff){

print\_error(WRONG\_VAL);

return 0;

}

memory[adr/16][(adr%16)\*2]=cmm[2][0];

memory[adr/16][(adr%16)\*2+1]=cmm[2][1];

return 1;

}

int command\_f(char cmm[10][COMMAND\_MAX]){

int i,j,s,end,val;

int srow,scol,erow,ecol;

s=make\_hex(cmm[1]);

end=make\_hex(cmm[2]);

val=make\_hex(cmm[3]);

//잘못된 주소를 입력한 경우 (16진수 타입이 아님, end가 범위를 벗어남, start> end)

if(s==-1 || end==-1 ||end>0xfffff|| s>end ){

print\_error(WRONG\_ADR);

return 0;

}

//value로 1byte 벗어나는 값을 입력한 경우

if(val>0xff){

print\_error(WRONG\_VAL);

return 0;

}

srow=s/16;

scol=(s%16)\*2;

erow=end/16;

ecol=(end%16)\*2;

for(i=srow; i<=erow; i++){

for(j=0; j<32; j+=2){

if((i==srow && j<scol )||( i==erow && j>ecol))

continue;

memory[i][j] = cmm[3][0];

memory[i][j+1]=cmm[3][1];

}

}

return 1;

}

void command\_reset(){

int i,j;

for(i=0; i<MEMORY\_ROW; i++){

for(j=0; j<32; j++){

memory[i][j]='0';

}

}

}

int command\_op\_mn(char \*s){

int key,flag=-1;

op\* tmp=NULL;

key=hash\_key(s);

for(tmp=oplist[key]; tmp!=NULL; tmp=tmp->next){

// 입력받은 명령어를 찾은 경우

if(strcmp(tmp->op\_name,s)==0){

printf("opcode %s is %X\n", s, tmp->op\_num);

flag=1;

}

}

//해당 명령어가 존재하지 않는 경우

if(flag==-1){

print\_error(WRONG\_PARAMETER);

return 0;

}

return 1;

}

void command\_op\_list(){

int i,flag=-1;

op\* tmp;

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

printf("%02d : ",i);

if(oplist[i]==NULL) //비어있는 경우

printf("empty");

for(tmp=oplist[i]; tmp!=NULL; tmp=tmp->next)

{

if(flag==-1){

printf("[%s,%X]", tmp->op\_name,tmp->op\_num);

flag=1;

}

else

printf("-> [%s,%X]", tmp->op\_name,tmp->op\_num);

}

flag=-1;

printf("\n");

}

}

int command\_file(char\* name){

FILE\* fp= fopen(name, "r");

char c[LINE\_MAX];

//해당 이름의 파일이 현재 디렉토리에 없는 경우 에러메시지

if(fp==NULL){

print\_error(WRONG\_FILE);

return 1;

}

//파일이 존재하면 내용 출력

while(fgets(c,LINE\_MAX,fp)!=NULL)

printf("%s",c);

fclose(fp);

return 1;

}

int command\_asm(char\* name){

FILE \*fp;

FILE \*fp\_lst, \*fp\_obj;

char asmfile[LINE\_MAX],\*lstfile,objfile[LINE\_MAX];

char c[LINE\_MAX],\*tmp, \*tmp2,prog\_name[7]={"INAINA"};

char inst[5][LINE\_MAX];

int i,line=-1,tnum;

//name.asm 파일 열기

strcpy(asmfile, name);

strcat(asmfile, ".asm");

fp=fopen(asmfile, "r");

if(fp==NULL){

print\_error(WRONG\_FILE);

return 0;

}

free\_symbol();

nobase=1;

lc[0]=0;

//first pass

while(fgets(c, LINE\_MAX,fp)!=NULL){

line++;

tnum=3;

lc[line+1]=lc[line];

if(c[0]=='.'){

code\_set[line].type=COMMENT;

continue;

}

//symbol 아닌경우, 코드 한줄을 2부분으로(tnum=2) 나눈다(inst parameter)

if(c[0]==' '||c[0]=='\t' || c[0]=='\n')

tnum=2;

if( code\_max < (int)strlen(c))

code\_max= (int)strlen(c);

//tmp2에는 tmun-1번 문자열을 짜르고 난 뒤의 남은 문자열을 저장할 것

tmp2=c;

for(i=0; i<tnum-1; i++){

tmp=strtok\_r(tmp2," \r\t\n",&tmp2);

if(tmp==NULL)

break;

strcpy(inst[i],remove\_space(tmp));

}

tmp2=remove\_space(tmp2);

//symbol인 경우

if(tnum==3){

if(!insert\_code(inst[1],tmp2, line)){ // 에러발생시 종료

printf("Error! Line %d\n", (line+1)\*5);

return 0;

}

//프로그램 이름이면

if(line==0 && strcmp(inst[1],"START")==0)

strncpy(prog\_name,inst[0],6);

else if(!insert\_symbol(inst[0],lc[line])){

printf("Error! Line %d\n", (line+1)\*5);

return 0;

}

}

//그외

else if(tnum==2){

if(!insert\_code(inst[0],tmp2,line)){

printf("Error! Line %d\n", (line+1)\*5);

return 0;

}

}

}

//END 없는 경우

if( end\_check==0){

printf("There dose not exist END directive\n");

return 0;

}

//lst, obj 파일 생성

lstfile=name;

strcpy(objfile, name);

strcat(lstfile, ".lst");

fp\_lst=fopen(lstfile,"wt");

strcat(objfile,".obj");

fp\_obj=fopen(objfile, "wt");

//파일 생성 오류시 에러 메시지 출력 후 종료

if(fp\_lst==NULL || fp\_obj ==NULL){

printf("File Error\n");

return 0;

}

//2nd pass

last\_line=line-1;

total\_size= lc[last\_line]-lc[0];

code\_max= code\_max + 4;

fclose(fp);

fp=fopen(asmfile,"r");

//lst 파일 생성

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

fgets(c, LINE\_MAX, fp);

if(c[0]!='.'){

if(!create\_lst(fp\_lst, c, i)){

printf("Error! Line %d\n", (i+1)\*5);

printf("Error line => %s\n", c);

fclose(fp\_lst);

fclose(fp\_obj);

fclose(fp);

//파일 삭제

remove(lstfile);

remove(objfile);

return 0;

}

else

fprintf(fp\_lst, "\n");

}

else

fprintf(fp\_lst,"%d\t%s",(i+1)\*5, c);

}

//obj 파일 생성

create\_obj(fp\_obj,prog\_name);

start\_check=0;

end\_check=0;

fclose(fp\_lst);

fclose(fp\_obj);

fclose(fp);

printf("output file : [%s], [%s]\n",lstfile,objfile);

return 1;

}

int command\_sym() {

int i,flag=1;

sym\* tmp;

for (i = 25; i >= 0; i--) {

if(symtable[i]!=NULL)

flag=0;

for (tmp = symtable[i]; tmp != NULL; tmp = tmp->next) {

//마지막 라인의 경우 엔터 제거

if (i == 0 && tmp->next == NULL)

printf("\t%s\t%04X", tmp->name, tmp->adr);

else

printf("\t%s\t%04X\n", tmp->name, tmp->adr);

}

}

//sybol table 이 비어있는 경우

if (flag==1)

return 0;

return 1;

}

int create\_obj(FILE\* fp, char\* prog\_name) {

int i, j = 0, k, size = 0, start = -1, end = -1;

fprintf(fp, "H%s", prog\_name);

for (i = (int)strlen(prog\_name); i<6; i++)

fprintf(fp, " ");

fprintf(fp, "%06X%06X\n", lc[0], total\_size);

for (i = 1; i <= last\_line;) {

//obj code 없는 문장

if (code\_set[i].type>4 && code\_set[i].type != CONST) {

i++;

continue;

}

//상수 중간에 짤라서 출력한 경우, 남은 obj 코드 출력

if (end != -1) {

size = (int)strlen(code\_set[i].obj) / 2 - end;

end = -1;

j = 1;

}

while (i + j <= last\_line) {

//obj code가 없는 경우

if (code\_set[i + j].type == DIREC || code\_set[i + j].type == BASE || code\_set[i + j].type == COMMENT) {

j++;

continue;

}

//변수를 선언한 경우 변수 선언이 끝나고 난 뒤부터 시작

if (code\_set[i + j].type == VAR) {

while (code\_set[i + j].type == VAR) {

if (i + j<last\_line)

j++;

}

break;

}

//상수인 경우

if (code\_set[i + j].type == CONST) {

//0x1E 넘어가면 나눠서 출력

if ((int)strlen(code\_set[i + j].obj) / 2 + size >30) {

end = 30 - size;

size = 30;

break;

}

else {

size += (int)strlen(code\_set[i + j].obj) / 2;

j++;

}

continue;

}

if (size + code\_set[i + j].type>30)

break;

size += code\_set[i + j].type;

j++;

}

//사이즈 만큼 objcode 출력

fprintf(fp, "T%06X%02X", code\_set[i].adr, size);

if (start != -1) {

for (k = start; k<(int)(strlen(code\_set[i].obj) / 2); k++)

fprintf(fp, "%c%c", (code\_set[i].obj)[2 \* k], (code\_set[i].obj)[2 \* k + 1]);

start = -1;

i++;

}

for (k = i; k<i + j; k++) {

if ((code\_set[k].obj)[0] != '\0')

fprintf(fp, "%s", code\_set[k].obj);

}

//상수가 중간에 짤려서 출력된 경우

if (end != -1) {

for (k = 0; k<end; k++) {

fprintf(fp, "%c%c", (code\_set[i + j].obj)[2 \* k], (code\_set[i + j].obj)[2 \* k + 1]);

}

start = k;

}

fprintf(fp, "\n");

i += j;

size = 0;

j = 0;

}

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

if(code\_set[i].type==4){

//immediate mode 아닌 경우

if((code\_set[i].operand)[0]!='#'){

fprintf(fp,"M%06X05\n", code\_set[i].adr+1);

}

}

}

//End Record obj 코드 생성된 첫 inst의 주소 출력

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

if (code\_set[i].num != -1) {

fprintf(fp, "E%06X\n", code\_set[i].adr);

break;

}

}

return 1;

}

int create\_lst(FILE\* fp, char\* inst, int line) {

int opcode, i, offset = 123456789;

int nixbpe[6];

int obj[8]; //object code

char\* operand;

if (inst == NULL)

return 0;

//엔터 제거

if (inst[strlen(inst) - 1] == '\n') {

//줄간격 맞추기

for (i = (int)strlen(inst) - 1; i <= code\_max; i++)

inst[i] = ' ';

inst[i] = '\0';

}

//line 출력

fprintf(fp, "%d\t", (line + 1) \* 5);

//lc 출력 안해도 되는 경우

if (inst[0] == '.' || code\_set[line].type == COMMENT || line == last\_line)

fprintf(fp, " \t");

//LoC

else if (lc[line]>0xffff)

fprintf(fp, "%05X ", lc[line]);

else

fprintf(fp, "%04X ", lc[line]);

fprintf(fp, "%s\t", inst); //Inst

opcode = code\_set[line].num;

operand = code\_set[line].operand;

pc = code\_set[line + 1].adr; //PC는 다음 inst adr

//COMMENT인 경우

if (code\_set[line].type == COMMENT) {

return 1;

}

//object code 생성할 필요 없는 경우

if (opcode == -1) {

if (code\_set[line].type == CONST) {

fprintf(fp, "%s", code\_set[line].obj);

}

return 1;

}

if(strcmp(code\_set[line].inst,"LDB")==0){

base=0;

//LDB 연산은 오직 immediate mode 에서만 가능 #

if(operand[0]=='#'){

//symbol이면 Loc 그대로 입력

if( read\_symbol(operand+1)!=-1)

base = read\_symbol(operand+1);

else if(make\_int(operand+1)!=-1)

base = make\_int(operand+1);

else

return 0;

}

nobase=0;

}

//obj code 생성!

//nixbpe 모두 0으로 초기화

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

nixbpe[i] = 0;

//format 1 or 2

if (code\_set[line].type == 1 || code\_set[line].type == 2) {

//format 1의 겨우 operand 있으면 오류

if (code\_set[line].type == 1 && operand[0] != '\0')

return 0;

(code\_set[line].obj)[0] = make\_char(opcode / 16);

(code\_set[line].obj)[1] = make\_char(opcode % 16);

(code\_set[line].obj)[2] = '\0';

//format 2인 경우

if (code\_set[line].type == 2) {

//parameter 1개인 inst

if (!strcmp(code\_set[line].inst, "CLEAR") || !strcmp(code\_set[line].inst, "TIXR") || !strcmp(code\_set[line].inst, "SVC")) {

if ((code\_set[line].operand2)[0] != '\0')

return 0;

(code\_set[line].obj)[2] = '0' + read\_reg(code\_set[line].operand);

(code\_set[line].obj)[3] = '0';

}

//parameter 2개

else {

if ((code\_set[line].operand2)[0] == '\0')

return 0;

//reg, n 형식 연산

if (!strcmp(code\_set[line].inst, "SHIFTL") || !strcmp(code\_set[line].inst, "SHIFTR")) {

if (make\_int(code\_set[line].operand2) == -1 || make\_int(code\_set[line].operand2)>0xf)

return 0;

(code\_set[line].obj)[2] = '0' + read\_reg(code\_set[line].operand);

(code\_set[line].obj)[3] = (code\_set[line].operand2)[0];

}

else {

(code\_set[line].obj)[2] = '0' + read\_reg(code\_set[line].operand);

(code\_set[line].obj)[3] = '0' + read\_reg(code\_set[line].operand2);

}

}

(code\_set[line].obj)[4] = '\0';

}

fprintf(fp, "%s", code\_set[line].obj);

return 1;

}

//operand 없는 경우( RSUB 아닌 경우 에러)

if (operand[0] == '\0') {

if (strcmp(code\_set[line].inst, "RSUB") != 0)

return 0;

(code\_set[line].obj)[0] = make\_char((opcode + 3) / 16);

(code\_set[line].obj)[1] = make\_char((opcode + 3) % 16);

strcpy(code\_set[line].obj + 2, "0000");

fprintf(fp, "%X0000", opcode + 3);

return 1;

}

//immediate mode

if (operand[0] == '#') {

nixbpe[1] = 1;

operand++; //#제거

//숫자인 경우

if (read\_symbol(operand) == -1) {

//decimal

if (make\_int(operand) != -1)

offset = make\_int(operand);

//잘못된 형식

else

return 0;

//range 넘어가는 경우

if (code\_set[line].type == 3 && offset>0xFFF)

return 0;

if (code\_set[line].type == 4 && offset>0xFFFFF)

return 0;

}

}

//indirect

else if (operand[0] == '@') {

operand++;

nixbpe[0] = 1;

//해당 symbol없는 경우

if (read\_symbol(operand) == -1)

return 0;

}

//simple addressing

else {

nixbpe[0] = 1;

nixbpe[1] = 1;

//format 4인 경우 offset 에 read adr 저장

if (code\_set[line].type == 4) {

nixbpe[5] = 1;

offset = read\_symbol(operand);

}

}

//offset 계산해야 하는 경우(immediate mode 중상수 대입을 제외하곤 모두 offset 계산필요)

if (offset == 123456789) {

pc = lc[line + 1];

offset = read\_symbol(operand) - pc;

//base relative

if (offset>0xFFF || offset\*(-1)> 0xFFF) {

//base 값이 없는 경우

if(nobase==1)

return 0;

nixbpe[3] = 1;

offset = read\_symbol(operand) - base;

//offset 음수인 경우 에러

if (offset<0)

return 0;

}

//PC

else {

nixbpe[4] = 1;

//음수인 경우 2의 보수의 decimal 값으로 ( a+(-a) = 0x1000 -> -a= 0x1000-a)

if (offset<0)

offset = 0x1000 + offset;

}

}

//index 모드인 경우

if (code\_set[line].type>2 && strcmp(code\_set[line].operand2, "X") == 0) {

//simple addressing인 경우에만 index 모드 가능하다

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

return 0;

nixbpe[2] = 1;

offset += x;

}

//최종offset 체크

if ((code\_set[line].type == 4 && offset>0xFFFFF) || (code\_set[line].type == 3 && offset >0xFFF))

return 0;

//obj code 계산

obj[0] = opcode / 16;

obj[1] = opcode % 16 + nixbpe[0] \* 2 + nixbpe[1];

obj[2] = 0;

//format 3

if (code\_set[line].type == 3) {

obj[3] = offset / 0x100;

obj[4] = (offset % 0x100) / 0x10;

obj[5] = (offset % 0x100) % 0x10;

obj[6] = '\0';

for (i = 2; i<6; i++)

obj[2] = obj[2] \* 2 + nixbpe[i];

//obj code 저장

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

(code\_set[line].obj)[i] = make\_char(obj[i]);

(code\_set[line].obj)[6] = '\0';

}

//format 4

else {

nixbpe[5]=1;

for (i = 2; i<6; i++)

obj[2] = obj[2] \* 2 + nixbpe[i];

obj[3] = offset / 0x10000;

obj[4] = (offset % 0x10000) / 0x1000;

obj[5] = ((offset % 0x10000) % 0x1000) / 0x100;

obj[6] = (((offset % 0x10000) % 0x1000) % 0x100) / 0x10;

obj[7] = (((offset % 0x10000) % 0x1000) % 0x100) % 0x10;

for (i = 0; i<8; i++)

(code\_set[line].obj)[i] = make\_char(obj[i]);

(code\_set[line].obj)[8] = '\0';

}

fprintf(fp, "%s", code\_set[line].obj);

return 1;

}

int store\_symbol(int adr , int size, char\* s,int line){

int i,len;

int tmp2;

char\* tmp;

tmp=strtok(s,"'");

//문자인 경우

if(!strcmp(tmp,"C")){

//문자인데 WORD로 선언한 경우 에러

if(size==3)

return 0;

tmp = strtok(NULL,"'");

//올바른 형식이 아닌 경우

if(strtok(NULL, " \r\t\n")!=NULL)

return 0;

len=(int)strlen(tmp);

if(len>30)

return 0;

//아스키 코드값 저장(최대상수문자열길이는30= 3byte)

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

tmp2=(int)(tmp[i]);

(code\_set[line].obj)[2\*i]=make\_char(tmp2/16);

(code\_set[line].obj)[2\*i+1]=make\_char(tmp2%16);

}

(code\_set[line].obj)[2\*len]='\0';

lc[line+1]=lc[line]+strlen(tmp);

return 1;

}

//16진수

else if(!strcmp(tmp, "X")|| !strcmp(tmp,"x")){

tmp = strtok(NULL,"'");

//올바른 입력 아닌 경우

if(tmp==NULL || strtok(NULL," \r\t\n")!=NULL)

return 0;

tmp2=make\_hex(tmp);

//올바른 16진수 형태가 아닌경우

if( tmp2==-1||( size==1 &&tmp2>0xff ) || (size==3 && tmp2>0xffffff ))

return 0;

for(i=0; tmp[i];i++){

(code\_set[line].obj)[i]= tmp[i];

}

(code\_set[line].obj)[i]='\0';

lc[line+1]=lc[line]+size;

}

//형식에 맞지 않음

else

return 0;

return 1;

}

int insert\_code(char\* inst, char\* operand, int line){

int key,flag=-1;

op\* tmp=NULL;

char para1[LINE\_MAX],\*para2=NULL;

strcpy(code\_set[line].inst, inst);

if(\*operand!=0)

strcpy(code\_set[line].operand,operand);

else

(code\_set[line].operand)[0]='\0';

(code\_set[line].operand2)[0]='\0';

(code\_set[line].obj)[0]='\0';

code\_set[line].adr=lc[line];

code\_set[line].type=3;

code\_set[line].num=-1;

//directives인 경우

if(strcmp(inst, "START")==0){

lc[0]=make\_int(operand);

lc[1]=lc[0];

code\_set[line].type = DIREC;

start\_check=1;

return 1;

}

if(strcmp(inst, "END")==0){

code\_set[line].type = DIREC;

end\_check=1;

return 1;

}

//상수인 경우

if(strcmp(inst, "WORD")==0){

code\_set[line].type = CONST;

lc[line+1]=lc[line]+3;

if(!store\_symbol(lc[line], 3, operand,line))

return 0;

return 1;

}

if(strcmp(inst, "BYTE")==0){

code\_set[line].type = CONST;

lc[line+1]=lc[line]+1;

if(!store\_symbol(lc[line],1,operand,line))

return 0;

return 1;

}

//변수

if(strcmp(inst, "RESW")==0){

code\_set[line].type = VAR;

lc[line+1]= lc[line]+ make\_int(operand)\*3;

return 1;

}

if(strcmp(inst, "RESB")==0){

code\_set[line].type = VAR;

lc[line+1]=lc[line]+make\_int(operand);

return 1;

}

/\*

if(strcmp(inst, "BASE")==0){

code\_set[line].type = BASE;

return 1;

}\*/

//format 4인 경우, 체크하고 +제거

if(inst[0]=='+'){

flag=1;

inst++;

}

key= hash\_key(inst);

for(tmp=oplist[key]; tmp!=NULL; tmp=tmp->next){

// 입력받은 명령어를 찾은 경우

if(strcmp(tmp->op\_name,inst)==0){

//+ 입력 했는데 format 1이나 2인 경우 에러

if(flag==1 && strlen(tmp->op\_type)==1)

return 0;

//operand가 있으면(0이 아니면) parameter ','단위로 나누기

if( \*operand!=0){

strcpy(para1,operand);

strcpy(code\_set[line].operand,remove\_space(strtok(para1,",")));

para2=strtok(NULL," ");

}

//parameter가 2개인 경우 operand2에 저장

if(para2!=NULL){

strcpy(code\_set[line].operand2,remove\_space(para2));

//잘못된 형식(매개변수 너무많음)

if(strtok(NULL," \r\t\n")!=NULL)

return 0;

}

code\_set[line].num= tmp->op\_num;

//해당 inst type 저장(op\_type은char배열 타입)

if(strlen(tmp->op\_type)==1)

code\_set[line].type= (int)(tmp->op\_type)[0] -'0';

else if( flag==1)

code\_set[line].type= 4;

else

code\_set[line].type=3;

//Locaton count 계산

code\_set[line].adr=lc[line];

lc[line+1]=lc[line]+code\_set[line].type;

return 1;

}

}

//명령어 없는 경우 주석

code\_set[line].type=COMMENT;

return 1;

}

int comp\_symbol(char \*name, char\* tmp){

int i,len= (int)strlen(name);

if( len>(int)strlen(tmp))

len = (int)strlen(tmp);

for(i=1; i<len; i++){

if( name[i] > tmp[i])

return 0;

if(name[i]< tmp[i])

return 1;

}

if( (int)strlen(name) > (int)strlen(tmp))

return 0;

return 1;

}

//Loc가 ard인 'name' symbol을 symbol table 에 저장

int insert\_symbol(char\* name, int adr){

int key=name[0]-'A';

sym \*node, \*tmp=NULL, \*tmp2=NULL;

//이름 첫글자가 숫자이면 에러

if( name[0]>='0' && name[0]<'9')

return 0;

//새로운 노드 생성

node= (sym\*)malloc(sizeof(sym));

strcpy(node->name,name);

node->adr=adr;

//처음 삽입

if( symtable[key]==NULL){

node->next= symtable[key];

symtable[key] = node;

}

//정렬하여 삽입

else{

tmp2= NULL;

for(tmp=symtable[key]; tmp!=NULL; tmp=tmp->next){

if( comp\_symbol(name, tmp->name)==0)

break;

tmp2=tmp;

}

//맨뒤에 삽입

if(tmp==NULL){

tmp2->next= node;

node->next=NULL;

}

//맨처음 삽입

else if(tmp2==NULL){

node->next= tmp;

symtable[key]= node;

}

//중간에 삽입

else{

tmp2->next = node;

node->next= tmp;

}

}

return 1;

}

void free\_symbol(){

sym\* tmp, \*tmp2;

int i;

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

if(symtable[i]==NULL)

continue;

for(tmp= symtable[i]; tmp!=NULL;){

tmp2= tmp;

tmp=tmp->next;

free(tmp2);

}

symtable[i]=NULL;

}

}

//name에 해당하는 sym의 adr return (없으면 -1 return )

int read\_symbol(char\* name){

int key= name[0]-'A';

sym\* tmp;

if(symtable[key]!=NULL){

for(tmp=symtable[key]; tmp!=NULL; tmp=tmp->next){

if( strcmp(tmp->name, name)==0)

return tmp->adr;

}

}

return -1;

}

int read\_reg(char\* reg){

if(!strcmp(reg, "A"))

return A;

if(!strcmp(reg, "B"))

return B;

if(!strcmp(reg, "S"))

return S;

if(!strcmp(reg, "T"))

return T;

if(!strcmp(reg, "F"))

return F;

if(!strcmp(reg, "L"))

return L;

if(!strcmp(reg, "PC"))

return PC;

if(!strcmp(reg, "X"))

return X;

if(!strcmp(reg, "SW"))

return SW;

return -1;

}

int command\_loader(int num, char file[10][COMMAND\_MAX]){

FILE\* f[3];

int i, j,result= 1;

//같은 파일 이름 체크

//파일 열기

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

f[i]= fopen(file[i+1], "r");

if( f[i]==NULL){

printf("File does not exist!\n");

for(j=0; j<i; j++)

fclose(f[j]);

return 0;

}

}

if(result){

//first pass

csaddr= progaddr;

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

if(!create\_estab(f[i], i)){

result=0;

break;

}

}

print\_estab();

csaddr=progaddr;

exeaddr=progaddr;

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

fclose(f[i]);

f[i]=fopen(file[i+1],"r");

//second pass

if(result){

if(!load\_obj(f[i],i)){

result=0;

}

}

csaddr+= cslth;

}

}

prog\_end= csaddr;

for(i=0; i<num; i++)

fclose(f[i]);

free\_estab();

if( result==1)

load\_check=1;

return result;

}

int load\_obj(FILE\* fp, int f\_index){

int ref[LINE\_MAX];

char c[LINE\_MAX], tmp[LINE\_MAX], tmp2[10];

int obj,t\_adr, t\_size,m\_adr, i,row=0, col=0 ;

ref[1]= csaddr;

while(fgets(c, LINE\_MAX, fp)){

if(c[0]=='D')

continue;

if(c[0]=='R'){

i=0;

while(1){

if( 1+9\*i >= (int)strlen(c))

break;

strncpy(tmp,c+1+8\*i,2);

tmp[2]='\0';

strncpy(tmp2, c+3+8\*i, 6);

tmp2[6]='\0';

strcpy(tmp2, remove\_space(tmp2));

//external ref 저장 (reference number)

ref[make\_hex(tmp)]= read\_estab(tmp2);

i++;

}

}

if(c[strlen(c)-1]=='\n')

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

//head record

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

strncpy(tmp, c+13,6);

tmp[6]='\0';

cslth= make\_hex(tmp);

}

//text record

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

//시작주소 & 길이 저장

strncpy(tmp, c+1,6);

tmp[6]='\0';

t\_adr= make\_hex(tmp)+csaddr;

strncpy(tmp2, c+7,2);

tmp2[2]='\0';

t\_size = make\_hex(tmp2);

row= t\_adr/16;

col= 2\*(t\_adr%16);

for(i=0; i< t\_size\*2; i++){

if( col>31){

col=0;

row++;

}

memory[row][col]= c[i+9];

col++;

}

}

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

//수정주소사이즈

char start\_address[LINE\_MAX];

char length[10];

strncpy(start\_address,c+1,6);

start\_address[6]='\0';

t\_adr= make\_hex(start\_address) + csaddr;

strncpy(length,c + 7,2);

length[2] = '\0';

t\_size=make\_hex(length);

row=t\_adr / 16;

col=(t\_adr % 16) \* 2;

//수정 half byte개수가 홀수인 경우 col++

if (t\_size % 2 == 1)

col++;

//더하거나 빼줄 symbol 이름

char reference\_string[10];

strcpy(reference\_string, c + 10);

m\_adr = ref[make\_hex(reference\_string)];

if (m\_adr == -1)

return 0;

if (c[9] == '-')

m\_adr \*= -1;

char value\_string[100];

//start\_address에 수정할 부분(obj 코드) 얻어오기

for(i=0; i<t\_size; i++){

value\_string[i]= memory[row][col+i];

}

value\_string[t\_size] = '\0';

obj = make\_hex(value\_string);

obj= (obj + m\_adr) % (1 << t\_size \* 4);

unsigned int obj\_front = obj & ((1 << (t\_size \* 4)) - 1);

sprintf(value\_string,"%06X", obj\_front);

//메모리에 수정 obj load

for(i=0; i<t\_size; i++){

memory[row][col+i] = value\_string[i];

}

}

else if(c[0]=='E'){

//exeaddr 없는 경우

if( strlen(c) ==1)

break;

strncpy(tmp,c+1, 6);

tmp[6]='\0';

exeaddr=make\_hex(tmp)+ ref[1];

break;

}

}

pc= exeaddr;

exeaddr\_run = exeaddr;

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

reg[i]=0;

return 1;

}

int create\_estab(FILE\* fp, int f\_index){

char name[10];

char c[LINE\_MAX], tmp[LINE\_MAX];

int adr,size,i;

while(fgets(c,LINE\_MAX,fp)!=NULL){

if(c[0]=='R' || c[0]=='T'|| c[0]=='R')

continue;

//haad record => 프로그램 이름 & 시작 주소 & 길이 estab 추가

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

//CS 이름, 시작주소, 길이 저장

strncpy(name, c+1, 6);

name[6]='\0';

strcpy(name ,remove\_space(name));

strncpy(tmp, c+7, 6);

tmp[6]='\0';

adr = make\_hex(tmp)+csaddr;

strncpy(tmp, c+13, 6);

tmp[6]='\0';

size = make\_hex(tmp);

cslth=size;

//중복된 CS인 경우 오류

if(read\_estab(name)!=-1){

printf("There exists duplicate control sections\n");

return 0;

}

insert\_estab(f\_index, name, size,adr);

}

//define record => estab에 추가(이름, 주소)

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

//이름& 길이 저장

size=-1;

i=0;

while(1){

strncpy(name, c+1+ i\*12, 6);

name[6]='\0';

strncpy(tmp, c+7+i\*12, 6);

tmp[6]='\0';

strcpy(name ,remove\_space(name));

i++;

if(1+(12\*i) >=(int)strlen(c))

break;

adr = make\_hex(tmp)+csaddr;

if(read\_estab(name)!=-1){

printf("There exist duplicate external sybols\n");

return 0;

}

insert\_estab(f\_index, name, size,adr);

}

}

}

csaddr+=cslth;

return 1;

}

void insert\_estab(int f\_index, char\* name, int size, int adr){

es\_node\* node= (es\_node\*)malloc(sizeof(es\_node));

es\_node\* tmp;

//새로운 es\_node 할당

strcpy(node->name , name);

node->size= size;

node->adr=adr;

node->next=NULL;

//처음 삽입하는 경우

if(estab[f\_index]==NULL){

estab[f\_index]= node;

}

else{

for(tmp=estab[f\_index]; tmp->next;)

tmp=tmp->next;

tmp->next= node;

}

}

int read\_estab(char\* name){

int i;

es\_node \*tmp=NULL;

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

if(estab[i]==NULL)

continue;

for(tmp=estab[i]; tmp; tmp=tmp->next){

//같은 이름의 노드가 있는 경우 주소값 return

if( !strcmp(name, tmp->name))

return tmp->adr;

}

}

return -1;

}

void free\_estab(){

int i;

es\_node \*tmp, \*tmp2;

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

if(estab[i]==NULL)

continue;

for(tmp=estab[i]; tmp;){

tmp2= tmp;

tmp=tmp->next;

free(tmp2);

}

estab[i]=NULL;

}

}

void print\_estab(){

int i,len=0;

es\_node\* tmp;

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

printf("section name\n");

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

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

if( estab[i]==NULL)

continue;

for(tmp=estab[i]; tmp; tmp = tmp->next){

if(tmp->size!=-1){

len+=tmp->size;

printf("%s\t\t\t\t%04X\t\t%04X\n", tmp->name, tmp->adr, tmp->size);

}

else

printf("\t\t%s\t\t%04X\n",tmp->name, tmp->adr);

}

printf("\n");

}

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

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

}

void read\_opcode(int n, int\* type){

int i;

op\* tmp;

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

if(oplist[i]==NULL)

continue;

for(tmp= oplist[i]; tmp;tmp= tmp->next){

if(tmp->op\_num==n){

if(strlen(tmp->op\_type)==1)

\*type= tmp->op\_type[0]-'0';

else

\*type= 3;

return;

}

}

}

}

void what\_is\_op\_code(char\* obj, int\* op, int\* n, int\* i,int\* x, int\* b, int\* p , int\* e, int\* type){

int tmp, tmp2;

//opcode, n,i 계산

//tmp = make\_hex(obj);

sscanf(obj,"%03X",&tmp);

tmp2= tmp%16;

tmp= tmp/16;

\*op = (tmp&0xFC);

\*n= (tmp&0x02)>>1;

\*i= tmp&0x01;

\*x= (tmp2&0x8)>>3;

\*b= (tmp2&0x4)>>2;

\*p= (tmp2&0x2)>>1;

\*e= tmp2&0x1;

//operation, format 구하기

read\_opcode(\*op,type);

if( \*type==3 && \*e==1)

\*type=4;

return;

}

void command\_run(){

//로드한 파일이 없는 경우

if( load\_check==0){

printf("There are no files loaded!\n");

return;

}

//남은 bp 없는 경우 프로그램 끝까지

// if( cur\_bp==bp\_index){

// run\_run\_run(prog\_end);

// cur\_bp=0;

// }

//

// else{

// run\_run\_run(bp[cur\_bp++]);

// }

run\_run\_run(100);

return;

}

void run\_run\_run(int end1){

int rg1, rg2,row, col, op,n,i,x,b,p,e, type, k;

int operand, ina,song,start,end,end\_flag;

char opr[5];

char tmp[5];

int target\_adr=0,imm=0;

while(pc<prog\_end){

row= pc/16;

col= 2\*(pc%16);

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

if(col+k>=32){

tmp[k] = memory[row+1][(col+k)%32];

}

else

tmp[k] = memory[row][(col+k)];

}

tmp[3]='\0';

what\_is\_op\_code(tmp, &op,&n,&i,&x,&b,&p,&e,&type);

start = pc;

end = start+type;

end\_flag = 0;

if(!RUN\_FLAG){

for(int j=0;j<bp\_index;j++){

if(start<=bp[j]&&bp[j]<end){

RUN\_FLAG = 1;

end\_flag = 1;

}

}

}

else

RUN\_FLAG = 0;

if(end\_flag)

{

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

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

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

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

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

return;

}

pc+= type;

//format 1,2

if( type<3){

rg1=memory[row][col+2]-'0';

rg2=memory[row][col+3]-'0';

//CLEAR

if(op==0xB4){

reg[rg1]=0;

}

//COMPR

else if( op==0xA0){

if( reg[rg1]< reg[rg2])

reg[SW]=-1;

else if(reg[rg1]==reg[rg2])

reg[SW]=0;

else

reg[SW]=1;

}

//TIXR

else if(op==0xB8){

reg[X]++;

if( reg[X]< reg[rg1])

reg[SW]=-1;

else if(reg[X]==reg[rg1])

reg[SW]=0;

else

reg[SW]=1;

}

continue;

}

//operand 복사

for(k=0; k< 2\*type-3; k++){

if(col+3+k>=32){

tmp[k] = memory[row+1][(col+3+k)%32];

}

else

tmp[k] = memory[row][(col+3+k)];

}

tmp[k]='\0';

//target\_adr= make\_hex\_neg(tmp);

sscanf(tmp, "%X\n", &operand);

target\_adr=operand;

imm=operand;

if( e==0 && operand>=0x800){

operand-=0x1000;

}

if(b==1 && p==0)

operand+= reg[B];

else if(b==0 && p==1)

operand+=pc;

if(x==1)

operand+=reg[1];

target\_adr= operand;

//immediate

if( n==0 && i==1){

//if( b==0 && p==0)

operand = target\_adr;

}

//simple

if( n==1 && i==1){

row= operand/16;

col= (operand%16)\*2;

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

if( col+k>=32){

tmp[k] = memory[row+1][(col+k)%32];

}

else

tmp[k] = memory[row][(col+k)];

}

tmp[k]='\0';

operand= make\_hex\_neg(tmp);

}

//indirect

else if( n==1 && i==0){

row= operand/16;

col= (operand%16)\*2;

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

if( col+k>=32){

tmp[k] = memory[row+1][(col+k)%32];

}

else

tmp[k] = memory[row][(col+k)];

}

tmp[k]='\0';

sscanf(tmp,"%06X",&operand);

row= operand/16;

col= (operand%16)\*2;

target\_adr=operand;

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

if(col+k>=32){

tmp[k] = memory[row+1][(col+k)%32];

}

else

tmp[k] = memory[row][(col+k)];

}

tmp[k]='\0';

sscanf(tmp, "%06X", &operand);

}

ina= target\_adr/16;

song = (target\_adr%16)\*2;

//LDA

if( op==0x00){

reg[A]=operand;

}

//LDB

else if( op== 0x68){

reg[B]=operand;

}

//LDT 1073

else if(op==0x74){

reg[T]= operand;

}

//LDX

else if(op==0x04){

reg[X]=operand;

}

//STL

else if(op==0x14){

char charr[6];

sprintf(charr, "%06X", reg[L]);

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

if( song+k>=32)

memory[ina+1][(song+k)%32]= charr[k];

else

memory[ina][song+k]= charr[k];

}

}

//STX

else if(op==0x10){

char charr[6];

sprintf(charr, "%06X", reg[X]);

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

if( song+k>=32)

memory[ina+1][(song+k)%32]= charr[k];

else

memory[ina][song+k]= charr[k];

}

}

//JSUB

else if(op==0x48){

reg[L]=pc;

pc= target\_adr;

}

//COMP

else if(op==0x28){

if(reg[A]<operand)

reg[SW]=-1;

else if(reg[A]==operand)

reg[SW]=0;

else

reg[SW]=1;

}

//RSUB

else if(op==0x4C){

pc= reg[L];

}

//JEQ

else if(op==0x30){

if( reg[SW]==0){

pc=target\_adr;

}

}

//J

else if(op==0x3C){

pc=target\_adr;

}

//STA

else if(op==0x0C){

char charr[6];

sprintf(charr, "%06X", reg[A]);

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

if( song+k>=32)

memory[ina+1][(song+k)%32]= charr[k];

else

memory[ina][song+k]= charr[k];

}

}

//TD

else if(op==0xE0){

reg[SW]=1;

}

//RD

else if(op==0xD8){

}

//STCH

else if(op==0x54){

if(song>=32)

memory[ina+1][song%32] = reg[A]/16;

else

memory[ina][song] = reg[A]/16;

if(song+1>=32)

memory[ina+1][(song+1)%32] = reg[A]%16;

else

memory[ina][song+1] = reg[A]%16;

}

//LDCH

else if(op==0x50){

reg[A]=reg[A]&0x00;

opr[2]='\0';

if(song>=32)

opr[0] = memory[ina+1][song%32];

else

opr[0] = memory[ina][song];

if(song+1>=32)

opr[1] = memory[ina+1][(song+1)%32];

else

opr[1] = memory[ina][song+1];

int ch;

sscanf(opr, "%02X", &ch);

reg[A]+= ch;

}

//WD

else if(op==0xDC){

//hello~~

}

//JLT

else if(op==0x38){

if( reg[SW]==-1){

pc=target\_adr;

}

}

//JGT

else if(op==0x34){

if(reg[SW]==1){

pc=target\_adr;

}

}

}

//register 출력

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

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

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

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

printf(" End Program\n");

return;

}

int make\_hex\_neg(char \*s){

int i,exp=1,tmp=make\_hex(s);

if((s[0]>='A' && s[0]<='F')||(s[0]>='8' && s[0]<='9')){

for(i=0; i<strlen(s); i++)

exp\*=16;

tmp=exp-tmp;

tmp=-tmp;

}

return tmp;

}

void command\_bp(int adr){

int i,j;

//break point 출력

if( adr==-1){

printf("\n breakpoint\n");

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

for(i=0; i<bp\_index; i++)

printf(" %04X\n", bp[i]);

return;

}

//break point clear

if(adr==-2){

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

for(i=0; i<bp\_index;i++)

bp[i]=0;

bp\_index=0;

return;

}

//bp 정렬해서 삽입

bp[bp\_index++]= adr;

for(i=0; i<bp\_index-1; i++){

for(j=i+1; j<bp\_index; j++){

if( bp[i]>bp[j]){

int tmp= bp[i];

bp[i]=bp[j];

bp[j]=tmp;

}

}

}

printf(" [ok] create breakpoint %04X\n", adr);

return;

}