**과목명: 시스템프로그래밍**

**CSE4100-01**

**<<Project #3>>**

**서강대학교 컴퓨터공학부**

**20151523**

**김동현**

목 차

1. **프로그램 개요**
   1. 프로그램에 관하여
   2. 프로그램 구성
2. **프로그램 설명**
   1. 프로그램 흐름도
3. **모듈 정의**

3.1 모듈 이름 : void loader\_pass1(char [], char [], char [])

3.1.1 기능

3.1.2 사용 변수

3.2 모듈이름: void loader\_pass2(char filename[])

3.2.1 기능

3.2.2 사용변수

3.3 모듈이름: void Make\_Estab(int typetmp, char nametmp[], int addrtmp, int lentmp)

3.3.1 기능

3.3.2 사용변수

3.4 모듈이름: int Find\_Estab(char name[])

3.4.1 기능

3.4.2 사용변수

3.5 모듈이름: int S\_to\_Hex(char strtmp[])

3.5.1 기능

3.5.2 사용변수

3.6 모듈이름: void Print\_Load(void)

3.6.1 기능

3.6.2 사용변수

3.7 모듈이름: void Free\_Estab(void)

3.7.1 기능

3.8 모듈이름: void Print\_Bp(void)

3.8.1 기능

3.8.2 사용변수

3.9 모듈이름: void Clear\_Bp(void)

3.9.1 기능

3.9.2 사용변수

3.10 모듈이름: void Make\_Num(int reg[], int num, int \*ta)

3.10.1 기능

3.10.2 사용변수

3.11 모듈이름: void Run\_Prog(void)

3.11.1 기능

3.11.2 사용변수

3.12 모듈이름: int TA\_Format3(int reg[],int \*ta)

3.12.1 기능

3.12.2 사용변수

3.13 모듈이름: int TA\_Format2(int reg[],int \*reg1, int \*reg2)

3.13.1 기능

3.13.2 사용변수

3.14 모듈이름: void Make\_Bplist(int baddr)

3.14.1 기능

3.14.2 사용변수

4. 전역 변수 정의

4.1 progaddr

4.2 int runstartaddr, runendaddr, runfinaladdr

4.3 int bplist[]

4.4 int bpnum

5. 코드

**1. 프로그램 개요**

* 1. **프로그램에 관하여**

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.

* 1. **프로그램 구성**
* 주소 지정 명령어 ( progaddr )
* Linking Loader ( loader )
* 프로그램 실행 ( run )
* debug 명령어 ( bp )

**2. 프로그램 설명**

* 1. **프로그램 흐름도**

스크린샷이(가) 표시된 사진

자동 생성된 설명

**3. 모듈 정의**

## 3.1모듈 이름 : void loader\_pass1(char [], char [], char [])

### 3.1.1기능

linking loader 중 pass1의 기능을 수행하는 함수이다. .obj파일을 한줄 씩 읽어가며 ESTAB을 생성하고 duplicate symbol에 대한 에러체크를 수행한다. .obj 파일의 H레코드, D레코드만을 읽은 후 프로그램을 종료한다. 만약 정상적으로 종료되지 않고 duplicate symbol 등의 에러가 발생했다면 pass2를 실행시키지 않고 종료한다. 정상적으로 종료되었을 경우 ESTAB List를 출력하고 pass2를 실행한다.

### 3.1.2사용 변수

FILE \*fp1, \*fp2, \*fp3 : 각각 첫 번째, 두 번째, 세 번째 filename을 열어보고 유효한 파일의 개수를 세는 파일 포인터이다.

FILE \*fp: 각각의 filename들을 반복문 내에서 열어볼 파일 포인터.

char filename[3][30],strmp[100],nametmp[10],addrtmp[10],lengthtmp[10] : 각각 문자열 배열로 사용할 변수들, 임시 변수로 주로 사용.

int error\_flag : 에러 체크에 사용될 flag

int file\_cnt: file개수를 저장하는 변수

int I,j,k index 변수

int proglen : program 길이를 저장하는 변수.

int startaddr: 시작 주소를 저장하는 변수, Define 변수들을 입력 받을 때 사용한다.

## 3.2모듈이름: void loader\_pass2(char filename[])

### 3.2.1기능

### linking loader 중 pass2에 해당하는 함수. 에러체크는 ESTAB에 존재하지 않는 SYMBOL 등을 체크하였다. 이후 run드에도 사용될 progaddr나 proglength는 저장하여 사용한다. .obj파일을 읽어 실제로 loading하는 작업으로서 modification 코드 처리 등을 수행한다. 뿐만 아니라 T레코드 등을 읽어 실제 memory에 저장하여 가상의 sic/xe 머신을 구현한다.

### 3.2.2사용변수

FILE \*fp;

int csaddr; //프로그램 시작주소

int cslength=0; //프로그램 길이

int address; //주소값

int reference\_num; //reference number에 사용할 배열과 index

int reference\_arr[100];

int loctmp; //임시변수

int len; //길이를 저장하는 임시변수

int t\_len; //t record의 길이를 저장하는 변수

int obj\_code; //obj\_code를 저장하는 변수.

int modifi\_num; //modifi num을 저장한다.

int immediate\_num; //bit mask 등에 사용할 변수.

int edit\_num; //modification +05, -06 등에서의 숫자를 저장하는 변수.

int i; //index

int error\_flag=0; //에러체크용 플래그

int idx; //index

char modifi\_type; //modification type(+,-)을 저장하하는 변수

char nametmp[10]; //이름을 저장하는 임시변수.

char strtmp[100]; //str을 저장하는 임시변수.

char \*str; //fgets에 사용할 변수.

Estab\_Table \*tmp; //ESTAB에서 index로 사용할 임시변수.

## 3.3모듈이름: void Make\_Estab(int typetmp, char nametmp[], int addrtmp, int lentmp)

### 3.3.1기능

ESTAB을 만드는 함수이다. Hash function을 사용하여 편리하게 Symbol의 address를 구할 수 있다. 또한 Program Name에 대한 Label과 Symbol에 대한 Label을 구분해서 사용한다. 각각 프로그램의 길이를 저장한다.

### 3.3.2사용변수

inti dx;

Estab\_Table \*tmp, //idx로 사용할 변수.

Estab\_Table \*new //ESTAB에 추가할 노드 변수.

## 3.4모듈이름: int Find\_Estab(char name[])

### 3.4.1기능

ESTAB에서 적절한 주소값을 찾아 이를 리턴해주는 함수. 만약 SYMBOL이 존재하지 않는다면

-1을 리턴한다.

### 3.4.2사용변수

int idx;

Estab\_Table \*tmp; //idx로 사용할 변수.

## 3.5모듈이름: int S\_to\_Hex(char strtmp[])

### 3.5.1기능

문자열을 입력받아 이 문자열이 적절한 16진수 타입인지 확인하고 이를 16진수로 바꿔주는 함수. 만약 적절하지 않다면 -1을 리턴한다.

### 3.5.2사용변수

int i; //idx로 사용할 변수.

int hex=0; //hex값을 저장할 변수.

int digit=1; //digit변수로 사용할 변수.

## 3.6모듈이름: void Print\_Load(void)

### 3.6.1기능

Linking loading 수행 중 ESTAB List를 print하는 함수.

### 3.6.2사용변수

int i,j; //idx로 사용할 변수

int count=0; //ESTAB을 addr 순서대로 정렬할 때 사용할 변수

int totallen=0; //ESTAB내에 있는 프로그램들의 총 길이.

## 3.7모듈이름: void Free\_Estab(void)

### 3.7.1기능

ESTAB의 메모리를 해제한다.

3.7.2 사용변수

int i; //idx로 사용할 변수

Estab\_Table \*tmp; //idx로 사용할 변수

## 모듈이름: void Print\_Bp(void)

### 3.8.1기능

Breaking Point를 모두 출력하는 함수.

### 3.8.2사용변수

Symbol\_Table \*tmp //idx로 사용할 변수

## 3.9모듈이름: void Clear\_Bp(void)

### 3.9.1기능

Breaking Point를 비워주는 함수.

### 3.9.2사용변수

Symbol\_Table \*tmp //idx로 사용할 변수

## 3.10모듈이름: void Make\_Num(int reg[], int num, int \*ta)

### 3.10.1기능

memory에 레지스터에 있는 값을 적절히 끊어서 저장하는 함수.

### 3.10.2사용변수

없음.

## 3.11모듈이름: void Run\_Prog(void)

### 3.11.1기능

프로그램을 해당Breaking point까지 실제로 돌리는 함수. if else 문장 안의 조건들은 각각 opcode를 나타내며, 해당 opcode에 따라 reg[] 배열에 저장되어있는 register 값을 참조, 변화시키며 프로그램을 진행한다..

### 3.11.2사용변수

int op\_number //bitmask를 사용하여 메모리의 내용 중 opcode에 대한 부분만을 뽑아내어 opcode와 비교한 후 적절한 명령어를 찾는다.

int reg[] //index번호가 각각 A,X,L 등의 레지스터의 번호와 상응한다. 배열값은 해당 레지스터에 들어있는 값으로 간주한다.

int ta,reg1,reg2 //각각 target address, register1, register2 값을 저장

int I //idx변수

int swtmp //sw register에 들어갈 값을 임시로 저장.

## 3.12모듈이름: int TA\_Format3(int reg[],int \*ta)

### 3.12.1기능

Format3와 Format4의 경우에 적절한 target address를 출력하는 함수.

### 3.12.2사용변수

int nixbpe, disp //각각 nixbpe, display값으로 사용할 변수

## 3.13모듈이름: int TA\_Format2(int reg[],int \*reg1, int \*reg2)

### 3.13.1기능

Format2의 Target address를 계산해준다.

### 3.13.2사용변수

없음

## 3.14모듈이름: void Make\_Bplist(int baddr)

### 3.14.1기능

Breaking Point list를 linked list 형태로 만든다.

### 3.14.2사용변수

Symbol\_Table \*new, \*tmp;//각각 새로 넣을 node와 idx변수로 사용

**4. 전역변수 정의**

## progaddr

progaddress, 즉 시작주소를 저장한다..

## int runstartaddr, runendaddr, runfinaladdr

모두 run에서 사용되는 변수로 각각 breaking point의 시작 주소, 끝 주소, END의 주소이다.

## 4.3 int bplist[]

breaking point의 목록을 저장해놓는 배열이다.

## int bpnum

breaking point의 목록을 저장해놓는 배열에 대한 index이다.

## int firsttry

프로그램을 구동하는 첫 시도인지에 대한 판단 flag이다.

**5. 코드 설명**

/\*포함되는 파일\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

#include <sys/stat.h>

#define max\_memory 1048576 //It equals to 16 \* 65536 which is the whole number of the memory.

#define Header 1

#define Define 2

/\* Usage: It is the linked list which saves the information of the history.

\* Variable num : number of valid command input.

\* name: name of valid command input. \*/

typedef struct linkhistory{

int num;

char name[100];

struct linkhistory \*link;

}Link\_History;

/\* Usage: It is the linked list which saves the information of the opcode and mnemonic.

\* Variable op\_num : opcode number of mnemonic.

\* op\_mnemonic : name of the mnemonic.

\* type : type of the mnemonic(1~4). \*/

typedef struct opcode\_table{

int op\_num;

char op\_mnemonic[20];

char type[10];

struct opcode\_table \*link;

}Opcode\_Table;

/\* symbol table 만들 때 사용.

\* 변수 순서대로 입력받음. \*/

typedef struct sym\_tab{

char symbol[10];

int locctr;

struct sym\_tab \*link;

}Symbol\_Table;

/\* Pass1에서의 H 와 D Record를 저장시 사용

\* type1일 경우 H 타입, type2일 경우 \*/

typedef struct l\_l{

int type;

char name[30];

int addr;

int length;

int startaddr;

struct l\_l \*link;

}Estab\_Table;

/\* Name : Basic\_State()

\* Parameter: None

\* Usage : Shows the basic state siscim> \*/

void Basic\_State(void);

/\* Name : Read\_Op()

\* Parameter: None

\* Usage : Read the opcode, and mnemonic

\* written in "opcode.txt". \*/

int Read\_Op(void);

/\* Name : Helplist()

\* Parameter: None

\* Usage : Print the helplist. \*/

void Helplist(void);

/\* Name : Make\_History()

\* Parameter: char command[]

\* Usage : Make the linked list which stores

\* the information of the history. \*/

void Make\_History(char \*);

/\* Name : Free\_History()

\* Parameter: None

\* Usage : Free the linked list which stores

\* the information of the history. \*/

void Free\_History(void);

/\* Name : Print\_History()

\* Parameter: None

\* Usage : Print the every history which

\* consists valid command input. \*/

void Print\_History(void);

/\* Name : Print\_Dir()

\* Parameter: None

\* Usage : Print every files which is in

\* the current directory. \*/

void Print\_Dir(void);

/\* Name : Reset\_Dump()

\* Parameter: None

\* Usage : Reset every memories into '0' \*/

void Reset\_Dump(void);

/\* Name : Dump\_Print()

\* Parameter: int start, int end

\* Usage : Print all the dump memories

\* having the address from start

\* to end. \*/

void Dump\_Print(int, int);

/\* Name : Dump\_Edit()

\* Parameter: int address, int value

\* Usage : Edit the value of specific

\* address. \*/

void Dump\_Edit(int, int);

/\* Name : Dump\_Fill()

\* Parameter: int start, int end, int value

\* Usage : Fill the value to address from

\* start to end. \*/

void Dump\_Fill(int, int, int);

/\* Name : Op\_Make()

\* Parameter: int op\_num,char mnemonic,char type

\* Usage : Make the linked list of opcode.

\* consisting opcode number, mnemonic,

\* name and type of the mnemonic. \*/

void Op\_Make(int, char [], char []);

/\* Name : Op\_Free()

\* Parameter: None

\* Usage : Free the linked list of opcode. \*/

void Op\_Free(void);

/\* Name : Op\_Mnemonic()

\* Parameter: char Mnemonic

\* Usage : Print the opcode of the mnemonic.\*/

void Op\_Mnemonic(char []);

/\* Name : Op\_List

\* Parameter: None

\* Usage : Print the list of opcode. \*/

void Op\_List(void);

/\* Name : opcodetable\_func\_make\_hash\_given()

\* Parameter: char Mnemonic

\* Usage : return hasing number of given

\* mnemonic using hash function. \*/

int opcodetable\_func\_make\_hash\_given(char []);

/\* filename에 해당하는 파일을 현재 디렉터리에서 찾아서 한줄씩 출력

\* 만약 없다면 에러출력. 성공시 1출력 실패시 0출력. \*/

int Type\_Print(char []);

/\* filename을 입력받아 symbol table을 작성한다. 정상적으로 종료시 1 출력 아닐시 -1\*/

int Make\_Symbol\_And\_Loc(char []);

/\* symbol과 locctr을 linked list에 추가. \*/

void Make\_Symtab(char [], int);

/\* Symbol Table Free한다. \*/

void Free\_Symtab(void);

/\* Symbol Table에 이미 들어있는 symbol인지 확인. \*/

int Is\_Used\_Sym(char []);

/\* Opcode중에 겹치는것이 있나 확인. \*/

int Is\_Op(char []);

/\*symbol table을 출력\*/

void Print\_Symtab(void);

/\*filename에 해당하는 asm파일을 읽어서 obj 파일과 lst 파일 생성. \*/

void Assemble(char []);

/\*opcode의 타입을 리턴\*/

int Type\_Return(char []);

/\*Symbol 리셋\*/

void Reset\_Symbol(void);

/\*Byte 다음에 오는 문자열 체크\*/

int Byte\_Check(char []);

/\* Symbol을 내림차순으로 바꿔 새로운 배열에 저장. \*/

void Change\_Sym(void);

/\* 각각 format1, format2, format3, format4에 대한

\* objcode를 생성해 준다. \*/

int format1(int);

int format2(int, char, char);

int format3(int, int, int);

int format4(int, int, int);

/\* Mnemonic의 opcode를 출력. \*/

int Opnum\_Return(char []);

/\* String이 Decimal인지 검사. \*/

int Is\_Dec(char []);

/\* Byte C''타입의 형식을 체크. \*/

int Check\_C(char []);

/\* 문자열이 Register인지 검사. \*/

int Is\_Reg(char []);

/\* filename1, filename2, filename3을 받아서 각각

\* 내용을 읽고 그 내용을 적절히 저장하여 load map을 화면에 출력하는

\* 함수. 만약 오류가 존재할 경우 에러 내용을 화면에 출력한다. \*/

void loader\_pass1(char [], char [], char []);

/\* filename을 읽어 Modification 등을 진행하여 dump에 메모리를 적절하게

\* 변경하여 저장하는 함수. \*/

void loader\_pass2(char []);

/\* ESTAB을 만드는 함수. 이름에 따른 Hash function을 사용한다.\*/

void Make\_Estab(int, char [], int, int);

/\* ESTAB에 이미 들어있는 Symbol인지 확인. 만약 존재한다면 \*/

int Find\_Estab(char []);

/\* String을 입력받아 이를 16진수 수로 바꿔주는 함수. \*/

int S\_to\_Hex(char []);

/\* Load Map을 출력하는 함수. \*/

void Print\_Load(void);

/\* ESTAB Free해주는 함수. \*/

void Free\_Estab(void);

/\* bp list를 print하는 함수. \*/

void Print\_Bp(void);

/\* bp list를 clear하는 함수. \*/

void Clear\_Bp(void);

/\* run에서 계산할때 넣을 숫자를 계산하는 함수. \*/

void Make\_Num(int [], int, int\*);

/\* run하는 함수. \*/

void Run\_Prog(void);

/\* Format 3,4의 계산을 하는 함수. \*/

int TA\_Format3(int [], int \*);

/\* Format 2의 계산을 하는 함수. \*/

int TA\_Format2(int [], int\*, int\*);

/\* Bplist 배열을 오름차순으로 만드는 함수. \*/

void Make\_Bplist(int );

/\* 소스 파일 \*/

#include "20151523.h"

Symbol\_Table\* Dec\_Table;

Link\_History\* Used\_Command = NULL; //linked list for history

Opcode\_Table\* Op\_Table[20]; //linked list for opcode

Symbol\_Table\* Sym\_Table[20];

Estab\_Table\* Es\_Table[20];

int History\_Num=1; //history variable

unsigned char memory[max\_memory]={0,}; //memory variable

int addr=0; //address variable

int LOCCTR;

int loc\_length;

int op\_length[100];

int loc\_save[10000];

int mod\_list[50]={0,};

int firsttry;

int progaddr;

int runstartaddr;

int runendaddr;

int runfinaladdr;

Symbol\_Table \*bp\_list=NULL;

int main(void){

char command[100],first\_str[100],second\_str[100],comma,fcomma,str\_error,com\_error;

char filename1[30],filename2[30],filename3[30];

int start,end;

int dump\_check,com\_check;

int eaddr, evalue;

int fstart,fend,fvalue;

int blank\_flag,opflag;

int i;

Symbol\_Table \*bptmp;

bptmp = bp\_list;

runstartaddr = 0;

runendaddr = 0;

runfinaladdr = 0;

opflag = Read\_Op();

progaddr = 0;

firsttry=0;

while(opflag){

/\*Initializing. \*/

blank\_flag=0;

comma='c';

fstart = -1;

fend = -1;

fvalue = -1;

fcomma = '\0';

eaddr=-1;

evalue=-1;

start=-1;

end=-1;

com\_check=-1;

str\_error='\0';

dump\_check=-1;

first\_str[0] = '\0';

second\_str[0] = '\0';

filename1[0] = '\0';

filename2[0] = '\0';

filename3[0] = '\0';

Basic\_State();

gets(command);

com\_check = sscanf(command,"%s %c",first\_str,&com\_error);

/\*Input command and determine whether the command is valid or not. \*/

for(i=0; i<(int)strlen(command); i++){

if(command[i] == '\0' || command[i] == ' ' || command[i] == '\t')

blank\_flag += 1;

}

if(blank\_flag == (int)strlen(command))

continue;

if((strcmp(first\_str,"q") == 0 || strcmp(first\_str,"quit") == 0) && com\_check == 1){ //quit program

break;

}

else if((strcmp(first\_str,"h") == 0 || strcmp(first\_str,"help") == 0) && com\_check == 1){ //help

Make\_History(command);

Helplist();

}

else if((strcmp(first\_str,"d") == 0 || strcmp(first\_str,"dir") == 0) && com\_check == 1){ //directory

Make\_History(command);

Print\_Dir();

}

else if((strcmp(first\_str,"hi") == 0 || strcmp(first\_str,"history") == 0) && com\_check == 1){ //history

Make\_History(command);

Print\_History();

}

else if(strcmp(first\_str,"reset") == 0 && com\_check == 1){

Make\_History(command);

Reset\_Dump(); //reset all the memories

}

else if(strcmp(first\_str,"du") == 0 || strcmp(first\_str,"dump") == 0){ //dump

dump\_check = sscanf(command,"%s %x %c %x %c",first\_str,&start,&comma,&end,&str\_error);

if(dump\_check == 1 && start == -1 && comma == 'c' && end == -1 && str\_error == '\0' && sscanf(command,"%s %s",first\_str,second\_str) == 1){

Make\_History(command);

Dump\_Print(addr,addr+159);

}

else if(dump\_check == 2 && comma == 'c' && end == -1 && str\_error == '\0'){

if(start<0 || start>=max\_memory)

printf("dump range check!\n");

else{

Dump\_Print(start,start+159);

Make\_History(command);

}

}

else if(dump\_check == 4 && comma == ',' && str\_error == '\0' ){

if(start<0 || start>=max\_memory || end>=max\_memory || start > end)

printf("dump range check!\n");

else{

Dump\_Print(start,end);

Make\_History(command);

}

}

else{

printf("invalid command!\n");

}

}

else if(strcmp(first\_str,"e") == 0 || strcmp(first\_str,"edit") == 0){ //edit

dump\_check = sscanf(command,"%s %x %c %x %c",first\_str,&eaddr,&comma,&evalue,&str\_error);

if(dump\_check == 4 && comma == ',' && evalue != -1 && eaddr != -1 && str\_error == '\0'){

if(evalue <= 255 && evalue >= 0 && eaddr >= 0 && eaddr < max\_memory){

Dump\_Edit(eaddr,evalue);

Make\_History(command);

}

else{

printf("Edit range check!\n");

}

}

else{

printf("invalid command!\n");

}

}

else if(strcmp(first\_str,"f") == 0 || strcmp(first\_str,"fill") == 0){ //fill

dump\_check = sscanf(command,"%s %x %c %x %c %x %c",first\_str,&fstart,&comma,&fend,&fcomma,&fvalue,&str\_error);

if(dump\_check == 6 && comma == ',' && fcomma == ',' && str\_error == '\0'){

if(fstart <= fend && fstart >= 0 && fend < max\_memory && fvalue >= 0 && fvalue <= 255){

Dump\_Fill(fstart,fend,fvalue);

Make\_History(command);

}

else{

printf("fill range check!\n");

}

}

else{

printf("invalid command!\n");

}

}

else if(strcmp(command,"opcodelist") == 0 && com\_check == 1){ //opcode list

Op\_List();

Make\_History(command);

}

else if(strcmp(first\_str,"opcode") == 0){ //opcode mnemonic

dump\_check = sscanf(command,"%s %s %c",first\_str,second\_str,&str\_error);

if(dump\_check == 2 && str\_error == '\0'){

Op\_Mnemonic(second\_str);

Make\_History(command);

}

else{

printf("invalid command!\n");

}

}

else if(strcmp(first\_str,"type") == 0){

dump\_check = sscanf(command,"%s %s %c",first\_str,second\_str,&str\_error);

if(dump\_check == 2 && str\_error == '\0'){

Type\_Print(second\_str);

Make\_History(command);

}

}

else if(strcmp(first\_str,"symbol") == 0 && com\_check == 1){

Print\_Symtab();

Make\_History(command);

}

else if(strcmp(first\_str,"assemble") == 0){

dump\_check = sscanf(command,"%s %s %c",first\_str,second\_str,&str\_error);

if(dump\_check == 2 && str\_error == '\0'){

Free\_Symtab();

Reset\_Symbol();

Assemble(second\_str);

}

else{

printf("Invalid command!\n");

}

Make\_History(command);

}

else if(strcmp(first\_str,"progaddr") == 0){

//progaddr꼴

if(com\_check == 1){

progaddr = 0;

Make\_History(command);

}

else{

dump\_check = sscanf(command,"%s %s %c",first\_str,second\_str,&str\_error);

if(dump\_check == 2 && str\_error == '\0'){

dump\_check = sscanf(command,"%s %X %c",first\_str,&eaddr,&str\_error);

if(dump\_check == 2 && str\_error == '\0'){

progaddr = eaddr;

Make\_History(command);

}

else{

printf("Invalid command!\n");

}

}

else{

printf("Invalid command!\n");

}

}

//progaddr이랑 progaddr address 꼴 두개 처리 해야함

}

else if(strcmp(first\_str,"loader") == 0){

dump\_check = sscanf(command,"%s %s %s %s %c",first\_str,filename1,filename2,filename3,&str\_error);

if(dump\_check >= 2 && dump\_check <= 4 && str\_error == '\0'){

Free\_Estab();

runstartaddr = progaddr; //run 시작주소 저장

//printf("%X\n",runstartaddr);

loader\_pass1(filename1, filename2, filename3);

runfinaladdr = progaddr; //run 끝난주소 저장

//printf("%X\n",runendaddr);

Make\_History(command);

}

else{

printf("Invalid command!\n");

}

//최소 한개에서 최대 3개까지 입력가능

}

else if(strcmp(first\_str,"run") == 0 && com\_check == 1){

Make\_History(command);

if(runfinaladdr == 0){

printf("Nothing loaded!\n");

continue;

}

bptmp = bp\_list;

if(bptmp != NULL){

runendaddr = bptmp->locctr;

Run\_Prog();

runstartaddr = runendaddr;

printf("\tStop at checkpoint[%04X]\n",bptmp->locctr);

bptmp = bptmp->link;

bp\_list = bp\_list->link;

}

else{

runendaddr = runfinaladdr;

Run\_Prog();

firsttry = 0;

printf("\tEnd Program\n");

}

/\*

bptmp = bp\_list;

while(bptmp != NULL){

runendaddr = bptmp->locctr;

bptmp = bptmp->link;

Run\_Prog();

runstartaddr = runendaddr;

}

runendaddr = runfinaladdr;

Run\_Prog();

\*/

//run함수 구현 아직 안함

}

else if(strcmp(first\_str,"bp") == 0){

dump\_check = sscanf(command,"%s %s %c",first\_str,second\_str,&str\_error);

if(dump\_check == 1 && str\_error == '\0'){

Print\_Bp();

Make\_History(command);

}

else if(dump\_check == 2 && (strcmp(second\_str,"clear") == 0) && str\_error == '\0'){

Clear\_Bp();

Make\_History(command);

}

else if(dump\_check == 2){

dump\_check = sscanf(command,"%s %x",first\_str,&eaddr);

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

Make\_Bplist(eaddr);

//bplist[bpnum++] = eaddr;

Make\_History(command);

}

//bp address랑 bp clear 두개 짜야함

}

else{

printf("invalid command!\n");

}

}

Free\_History();

Op\_Free();

Free\_Symtab();

Free\_Estab();

return 0;

}

/\* Printing the basic state. \*/

void Basic\_State(void){

printf("sicsim> ");

}

/\* Reading "opcode.txt". \*/

int Read\_Op(void){

FILE \*fp;

int numtmp;

char mnemonictmp[20];

char typetmp[10];

int ret;

int i;

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

Op\_Table[i] = NULL;

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

if(fp == NULL){

printf("There is no opcode list in the directory\n");

return 0;

}

while(ret != EOF){

ret = fscanf(fp,"%x%s%s",&numtmp,mnemonictmp,typetmp);

if(ret == 3)

Op\_Make(numtmp,mnemonictmp,typetmp);

}

return 1;

}

/\* Print the helplist. \*/

void Helplist(void){

printf("h[elp]\n");

printf("d[ir]\n");

printf("q[uit]\n");

printf("hi[story]\n");

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

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

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

printf("reset\n");

printf("opcode mnemonic\n");

printf("opcodelist\n");

printf("assemble filename\n");

printf("type filename\n");

printf("symbol\n");

printf("progaddr [address]\n");

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

printf("run\n");

printf("bp [address]\n");

printf("bp clear\n");

}

/\* Insert new node into the history list. \*/

void Make\_History(char \*command){

Link\_History\* tmp;

Link\_History\* new;

new = (Link\_History\*)malloc(sizeof(Link\_History));

new->num = History\_Num++;

strcpy(new->name, command);

new->link = NULL;

if(Used\_Command == NULL){

Used\_Command = new;

}

else{

for(tmp = Used\_Command; tmp->link != NULL; tmp = tmp->link)

;

tmp->link = new;

}

}

/\* Free history linked list. \*/

void Free\_History(void){

Link\_History \*tmp;

while((tmp = Used\_Command) != NULL){

Used\_Command = Used\_Command->link;

free(tmp);

}

}

/\* Print the history list. \*/

void Print\_History(){

Link\_History\* tmp;

for(tmp = Used\_Command; tmp != NULL; tmp = tmp->link){

printf("%7d\t%s\n",tmp->num,tmp->name);

}

}

/\* Print all the files in the directory. \*/

void Print\_Dir(void){

DIR \*dirp;

struct dirent \*direntp;

struct stat file\_info;

int line=0;

char nametmp[100];

dirp = opendir(".");

while((direntp = readdir(dirp)) != NULL){

strcpy(nametmp, direntp->d\_name);

if(strcmp(nametmp,".") == 0 || strcmp(nametmp,"..") == 0)

continue;

lstat(nametmp,&file\_info);

printf("%20s",nametmp);

line++;

if(S\_ISDIR(file\_info.st\_mode))

printf("/");

else if(S\_IEXEC & file\_info.st\_mode)

printf("\*");

if(line % 3 == 0)

printf("\n");

}

printf("\n");

closedir(dirp);

}

/\* Reset all the memory into 0. \*/

void Reset\_Dump(void){

int i;

for(i=0; i<max\_memory; i++)

memory[i] = 0;

return ;

}

/\* Print dump. \*/

void Dump\_Print(int start,int end){

int i,j,k,addrtmp,ascii\_start;

if(end >= max\_memory)

end = max\_memory-1;

addrtmp = end+1;

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

for(i=0; i<start%16; i++)

printf(" ");

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

printf("%02X ",memory[i]);

if(i == end){

for(j=0; j<( 16-((end+1) % 16) ) % 16; j++){

printf(" ");

}

}

if((i+1) % 16 == 0 || i == end){

printf("; ");

ascii\_start = i - (i%16);

for(k=ascii\_start; k<ascii\_start+16; k++){

if(memory[k] >= 32 && memory[k] <= 127)

printf("%c",memory[k]);

else

printf(".");

}

printf("\n");

if(i < end)

printf("%05X ",((i+1)/16) \* 16);

}

}

addr = addrtmp % max\_memory;

}

/\* Edit the specific address's value. \*/

void Dump\_Edit(int eaddr,int evalue){

memory[eaddr] = evalue;

}

/\* Fill some value of address. \*/

void Dump\_Fill(int fstart,int fend,int fvalue){

int i;

for(i=fstart; i<=fend; i++){

memory[i] = fvalue;

}

}

/\* Make the linked list of a opcode. \*/

void Op\_Make(int numtmp,char mnemonictmp[],char typetmp[]){

int idxtmp;

Opcode\_Table \*new;

Opcode\_Table \*tmp;

/\* Memory Allocation. \*/

new = (Opcode\_Table\*)malloc(sizeof(Opcode\_Table));

new->op\_num = numtmp;

strcpy(new->op\_mnemonic, mnemonictmp);

strcpy(new->type, typetmp);

new->link = NULL;

idxtmp = opcodetable\_func\_make\_hash\_given(mnemonictmp);

if(Op\_Table[idxtmp] == NULL){

Op\_Table[idxtmp] = new;

}

else{

for(tmp = Op\_Table[idxtmp]; tmp->link != NULL; tmp = tmp->link)

;

tmp->link = new;

}

}

/\* Free the linked list of a opcode. \*/

void Op\_Free(void){

int i;

Opcode\_Table \*tmp;

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

while((tmp = Op\_Table[i]) != NULL){

Op\_Table[i] = Op\_Table[i]->link;

free(tmp);

}

}

}

/\* Find the opcode of specific mnemonic. \*/

void Op\_Mnemonic(char mnemonic[]){

int ret;

Opcode\_Table \*tmp;

ret = opcodetable\_func\_make\_hash\_given(mnemonic);

for(tmp = Op\_Table[ret]; tmp != NULL; tmp = tmp->link){

if(strcmp(tmp->op\_mnemonic,mnemonic) == 0){

printf("opcode is %02X\n",tmp->op\_num);

return ;

}

}

printf("There is no mnemonic such as %s\n",mnemonic);

}

/\* Print the linked list of a opcodelist. \*/

void Op\_List(void){

int i;

Opcode\_Table \*tmp;

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

printf("%d : ",i);

if(Op\_Table[i] != NULL){

printf("[%s,%X] ",Op\_Table[i]->op\_mnemonic,Op\_Table[i]->op\_num);

if(Op\_Table[i]->link != NULL){

for(tmp = Op\_Table[i]->link; tmp != NULL; tmp = tmp->link){

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

}

}

}

printf("\n");

}

}

/\* Make the hash function. \*/

int opcodetable\_func\_make\_hash\_given(char s[])

{

int i,sum;

int len=strlen(s);

sum=0;

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

{

sum+=s[i];

}

return sum%20;

}

/\* Return the type of the opcode. \*/

int Type\_Print(char filename[]){

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

char buffer[200];

if(fp == NULL){

printf("%s does not exist!\n",filename);

return 0;

}

while(fgets(buffer,200,fp)){

printf("%s",buffer);

}

fclose(fp);

return 1;

}

/\* Symbol Table 만들고 LOCCTR 첨가. \*/

int Make\_Symbol\_And\_Loc(char filename[]){

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

char buffer[200];

char tmp1[40],tmp2[40],tmp3[40];

int hexnum;

int scan\_check=0,loctmp,byte\_num;

int error\_flag=0,startloc;

int op\_cnt=0,loc\_cnt=0,op\_tmp=0;

int line\_cnt=0;

if(fp == NULL){

printf("%s does not exist!\n",filename);

return -1;

}

/\* 첫줄에 start 들어오기 전에 다른 instruction 들어오면 오류처리. \*/

while(fgets(buffer,200,fp)){

scan\_check = sscanf(buffer,"%s %s %x",tmp1,tmp2,&hexnum);

if(strcmp(tmp2,"START") == 0 && scan\_check == 3){

LOCCTR = hexnum;

loc\_save[loc\_cnt++] = LOCCTR;

loctmp = LOCCTR;

startloc = LOCCTR;

line\_cnt += 5;

break;

}

else if(tmp1[0] == '.')

continue;

else{

printf("Other instructions cannot come before than start!\n");

error\_flag=1;

}

}

/\* 둘째줄부터 \*/

while(fgets(buffer,200,fp)){

tmp1[0] = '\0';

tmp2[0] = '\0';

tmp3[0] = '\0';

scan\_check = -1;

byte\_num = -1;

sscanf(buffer,"%s",tmp1);

if(tmp1[0] == '.')

continue;

line\_cnt += 5;

scan\_check = sscanf(buffer,"%s %s %s",tmp1,tmp2,tmp3);

if(Is\_Used\_Sym(tmp1) != -1 && scan\_check == 3){ //이미 쓴 symbol

printf("Error! Line: %d Symbol already used!\n",line\_cnt);

error\_flag=1;

}

if(strcmp(tmp1,"END") == 0)

break;

if(Is\_Op(tmp2) == 1){ //opcode의 형식이 맞은 경우

loctmp = LOCCTR;

Make\_Symtab(tmp1,loctmp);

loctmp = Type\_Return(tmp2);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp+loctmp>30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += loctmp;

LOCCTR += loctmp;

}

else if(Is\_Op(tmp2) == 2){ //+opcode의 형식

loctmp = LOCCTR;

Make\_Symtab(tmp1,loctmp);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp+4>30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += 4;

LOCCTR += 4;

}

else if(strcmp(tmp2,"BYTE") == 0){ //BYTE가 들어왔을 경우

loctmp = LOCCTR;

if(tmp3[0] == 'X'){

byte\_num = Byte\_Check(tmp3);

}

else if(tmp3[0] == 'C'){

byte\_num = Check\_C(buffer);

}

if(byte\_num == -1){ //syntax 오류

printf("Error! Line: %d BYTE syntax error!\n",line\_cnt);

error\_flag = 1;

continue;

}

Make\_Symtab(tmp1,loctmp);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp + byte\_num>30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += byte\_num;

LOCCTR += byte\_num; //

}

else if(strcmp(tmp2,"RESW") == 0){ //RESW가 들어왔을 경우

if(Is\_Dec(tmp3) == -1){

printf("Error! Line: %d RESW syntax error!\n",line\_cnt);

error\_flag = 1;

continue;

}

loctmp = LOCCTR;

Make\_Symtab(tmp1,loctmp);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp != 0){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

LOCCTR += 3 \* atoi(tmp3);

}

else if(strcmp(tmp2,"RESB") == 0){ //RESB가 들어왔을 경우

if(Is\_Dec(tmp3) == -1){

printf("Error! Line: %d RESB syntax error!\n",line\_cnt);

error\_flag = 1;

continue;

}

loctmp = LOCCTR;

Make\_Symtab(tmp1,loctmp);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp != 0){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

LOCCTR += atoi(tmp3);

}

else if(strcmp(tmp2,"WORD") == 0){ //WORD가 들어왔을 경우

if(Is\_Dec(tmp3) == -1 || atoi(tmp3) > 0xFFFFFF){

printf("Error! Line: %d WORD syntax error!\n",line\_cnt);

error\_flag = 1;

continue;

}

loctmp = LOCCTR;

Make\_Symtab(tmp1,loctmp);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp + 3>30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += 3;

LOCCTR += 3;

}

else if(strcmp(tmp2,"BASE") == 0){

continue;

}

/\*여기까지는 LABEL이 들어온 case, 즉 symbol table에 추가해야할 상황\*/

/\* 여기부터는 LABEL이 없는 경우\*/

else{

if(Is\_Op(tmp1) == 1){ //opcode의 형식이 맞은 경우

loctmp = Type\_Return(tmp1);

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp + loctmp > 30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += loctmp;

LOCCTR += loctmp;

}

else if(Is\_Op(tmp1) == 2){ //+opcode의 형식

loc\_save[loc\_cnt++] = LOCCTR;

if(op\_tmp + 4 > 30){

op\_length[op\_cnt++] = op\_tmp;

op\_tmp=0;

}

op\_tmp += 4;

LOCCTR += 4;

}

else if(strcmp(tmp1,"BASE") == 0){

continue;

}

else{

printf("%s\n",tmp1);

printf("Error! Line: %d Opcode doesn't exist!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

op\_length[op\_cnt++] = op\_tmp;

loc\_length = LOCCTR - startloc;

fclose(fp);

if(error\_flag == 1)

return -1;

else

return 1;

}

void Make\_Symtab(char symbol[], int locctr){

int idx;

Symbol\_Table \*new;

Symbol\_Table \*tmp = NULL;

new = (Symbol\_Table\*)malloc(sizeof(Symbol\_Table));

strcpy(new->symbol,symbol);

new->locctr = locctr;

new->link = NULL;

idx = opcodetable\_func\_make\_hash\_given(symbol);

//printf("symbol2: %s %04X\n",symbol,locctr);

if(Sym\_Table[idx] == NULL){

Sym\_Table[idx] = new;

}

else{

for(tmp = Sym\_Table[idx]; tmp->link != NULL; tmp = tmp->link)

;

tmp->link = new;

}

}

void Free\_Symtab(void){

int i;

Symbol\_Table \*tmp;

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

while((tmp = Sym\_Table[i]) != NULL){

Sym\_Table[i] = Sym\_Table[i]->link;

free(tmp);

}

}

}

/\* 안겹치면 -1 겹치면 locctr 리턴\*/

int Is\_Used\_Sym(char symbol[]){

int idx;

Symbol\_Table \*tmp;

idx = opcodetable\_func\_make\_hash\_given(symbol);

for(tmp = Sym\_Table[idx]; tmp != NULL; tmp = tmp->link){

if(tmp==NULL)

return -1;

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

return tmp->locctr;

}

return -1;

}

/\* 안겹치면 0 겹치면 1 +꼴일경우 2 리턴\*/

int Is\_Op(char name[]){

int idx;

int flag=0;

char plus\_deleted[20];

Opcode\_Table \*tmp;

/\*+뺀 mnemonic 저장.\*/

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

strcpy(plus\_deleted,name+1);

idx = opcodetable\_func\_make\_hash\_given(plus\_deleted);

for(tmp = Op\_Table[idx]; tmp != NULL; tmp = tmp->link){

if(strcmp(tmp->op\_mnemonic,plus\_deleted) == 0)

flag = 2;

}

}

else{

idx = opcodetable\_func\_make\_hash\_given(name);

for(tmp = Op\_Table[idx]; tmp != NULL; tmp = tmp->link){

if(strcmp(tmp->op\_mnemonic,name) == 0)

flag = 1;

}

}

if(flag == 0)

return 0;

else if(flag == 1)

return 1;

else if(flag == 2)

return 2;

else

return 0;

}

/\* 내림차순으로 출력. \*/

void Print\_Symtab(void){

int i,count=0;

Symbol\_Table \*tmp;

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

if(Sym\_Table[i] != NULL){

for(tmp = Sym\_Table[i]; tmp != NULL; tmp = tmp->link)

count += 1;

}

}

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

printf("\t%s\t%04X\n",Dec\_Table[i].symbol,Dec\_Table[i].locctr);

}

/\*assemble 함

loc\_length < 프로그램 길이 \*/

void Assemble(char filename[]){

int exe\_flag=0,error\_flag=0,i;

int mod\_cnt=0;

int line\_cnt=0; //line 수

int scan\_check=0;

int loc\_cnt=0, op\_tmp=0, op\_cnt=0;

int pc\_num=0,base\_num=0; //각각 pc counter, base counter

int obj\_code; //obj 계산시 사용

int numtmp,baseflag=0;

int lentmp; //저장용

char buffertmp[200];

char byte\_tmp1[40];

char syn\_check; //syntax 에러 체크시 사용

char \*str,buffer[200],tmp1[40],tmp2[40],tmp3[40],tmp4[2],tmp5[2];

char str\_tmp[40],str\_tmp2[40];

char \*file\_lst, \*file\_obj,file\_tmp[30];

char \*regtmp1, \*regtmp2;

FILE \*fp, \*fp2, \*fp3;

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

op\_length[i]=0;

for(i=0; i<10000; i++)

loc\_save[i]=0;

fp = fopen(filename,"r");

exe\_flag=Make\_Symbol\_And\_Loc(filename);

if(exe\_flag == -1)

return ;

strcpy(file\_tmp,filename);

file\_lst = strtok(file\_tmp, ".");

strcat(file\_lst,".lst");

fp2 = fopen(file\_lst,"w");

file\_obj = strtok(file\_lst,".");

strcat(file\_obj,".obj");

fp3 = fopen(file\_obj,"w");

/\* 맨 처음 START를 입력 받고 프로그램을 시작. \*/

while(fgets(buffertmp,200,fp)){

tmp1[0]='\0';

tmp2[0]='\0';

tmp3[0]='\0';

tmp4[0]='\0';

tmp5[0]='\0';

scan\_check = sscanf(buffertmp,"%s %s %s %s %s",tmp1,tmp2,tmp3,tmp4,tmp5);

line\_cnt += 5;

if(strcmp(tmp2,"START") == 0 && scan\_check>=3){

str = strtok(buffertmp,".\n");

fprintf(fp2,"%3d\t%04X %-50s\n",line\_cnt,loc\_save[loc\_cnt+1],str);

fprintf(fp3,"H%-6s%06X%06X",tmp1,loc\_save[loc\_cnt+1],loc\_length);

loc\_cnt++;

break;

}

else if(tmp1[0] == '.'){

continue;

}

else{

printf("Line 5 : Other instructions cannot come before than start!\n");

error\_flag = 1;

}

}

/\* 두 번쨰 줄부터의 실행

\* object 코드를 실시간으로 생성하여 \*.lst파일에 입력한다.

\* 또한 \*.obj 파일에도 실시간으로 입력한다.

\* 반복문이 도는 동안 계속 저장될 변수로는 line 수, locctr 등이

\* 저장된다.\*/

while(fgets(buffertmp,200,fp)){

tmp1[0]='\0';

tmp2[0]='\0';

tmp3[0]='\0';

tmp4[0]='\0';

tmp5[0]='\0';

obj\_code=0;

numtmp=-1;

syn\_check='\0';

pc\_num = loc\_save[loc\_cnt+1];

scan\_check=0;

/\* 주석 제외 \*/

sscanf(buffertmp,"%s",tmp1);

if(tmp1[0] == '.')

continue;

strcpy(buffer,buffertmp);

str = strtok(buffertmp,".\n");

strtok(buffer,"\n");

scan\_check = sscanf(str,"%s %s %s %s %s",tmp1,tmp2,tmp3,tmp4,tmp5);

line\_cnt += 5;

/\* END가 나올경우 프로그램 종료 \*/

if(strcmp(tmp1,"END") == 0){

fprintf(fp2,"%3d\t %-50s\n",line\_cnt,str);

break;

}

/\* 앞에 label 있고 두번째께 opcodelist에 존재할 경우. \*/

else if(Is\_Op(tmp2) == 1){

/\* Symbol의 앞에 #이 올 경우

\* 순서대로 PC Addressing, Base Relative

\* 숫자가 들어올 경우 \*/

if(tmp3[0] == '#'){

if(Is\_Used\_Sym(tmp3+1) != -1){

numtmp = Is\_Used\_Sym(tmp3+1);

/\*Inside the range of PC addressing mode. \*/

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp2),0x12,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp2),0x14,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

sscanf(tmp3+1," %d %c",&numtmp,&syn\_check);

if(numtmp != -1 && syn\_check == '\0'){

if(numtmp >= -2048 && numtmp <= 4095){

obj\_code = format3(Opnum\_Return(tmp2),0x10,numtmp);

fprintf(fp2,"%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

/\* Operand 앞에 @가 붙어

\* Indirect Addressing일 경우. \*/

else if(tmp3[0] == '@'){

if(Is\_Used\_Sym(tmp3+1) != -1){

numtmp = Is\_Used\_Sym(tmp3+1);

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp2),0x22,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp2),0x24,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Operand가 Symbol Table에

\* 존재하는 Symbol일 경우. \*/

else if(Is\_Used\_Sym(tmp3) != -1){

/\*Operand에 Symbol Table에 존재하는 Label 입력\*/

if(Type\_Return(tmp2) == 3){

numtmp = Is\_Used\_Sym(tmp3);

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp2),0x32,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp2),0x34,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Operand 꼴이 다른 opcode의 경우\*/

else{

/\*레지스터 입력 또는 BUFFER,X의 형태\*/

if(Is\_Reg(tmp3) != -1){

if(Type\_Return(tmp2) == 2){

obj\_code = format2(Opnum\_Return(tmp2),Is\_Reg(tmp3),0);

fprintf(fp2,"%3d\t%04X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%04X",obj\_code);

op\_tmp += 2;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Operand가 비어있는 경우. \*/

else if(tmp3[0] == '\0'){

if(Type\_Return(tmp2) == 1){

obj\_code = format1(Opnum\_Return(tmp2));

fprintf(fp2,"%3d\t%04X %-50s %02X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 1 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%02X",obj\_code);

op\_tmp += 1;

}

else if(Type\_Return(tmp2) == 2){

obj\_code = format2(Opnum\_Return(tmp2),0,0);

fprintf(fp2,"%3d\t%04X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%04X",obj\_code);

op\_tmp += 2;

}

else if(Type\_Return(tmp2) == 3){

obj\_code = format3(Opnum\_Return(tmp2),0x30,0);

fprintf(fp2,"%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Symbol, X꼴 또는 register, register꼴. \*/

else{

strcpy(str\_tmp,tmp3);

strcat(str\_tmp,tmp4);

strcat(str\_tmp,tmp5);

strcpy(str\_tmp2,str\_tmp);

lentmp = (int)strlen(str\_tmp);

regtmp1 = strtok(str\_tmp2,",");

regtmp2 = strtok(NULL,",");

if(Type\_Return(tmp2) == 2 && Is\_Reg(regtmp1) != - 1 && Is\_Reg(regtmp2) != -1){

obj\_code = format2(Opnum\_Return(tmp2),Is\_Reg(regtmp1),Is\_Reg(regtmp2));

fprintf(fp2, "%3d\t%-4X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%04X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 2;

}

else if(str\_tmp[lentmp-1] == 'X' && str\_tmp[lentmp-2] == ','){

file\_lst = strtok(str\_tmp,",");

numtmp = Is\_Used\_Sym(file\_lst);

if(numtmp != -1 && Type\_Return(tmp2) == 3){

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp2),0x3A,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp2),0x3C,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\*Type2 A,B 형식\*/

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

}

/\* 앞에 label 있고 두번째께 (+)opcode 형태로 존재할경우

\* 각각 앞에 #, @, Symbol이 operand 위치에 올 경우. \*/

else if(Is\_Op(tmp2) == 2){

if(tmp3[0] == '#'){

numtmp = Is\_Used\_Sym(tmp3+1);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp2+1), 0x11, numtmp);

mod\_list[mod\_cnt++] = loc\_save[loc\_cnt] - loc\_save[0]+1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

sscanf(tmp3+1,"%d %c",&numtmp,&syn\_check);

if(numtmp != -1 && syn\_check == '\0'){

if(numtmp >= 0 && numtmp <= 0xFFFFF){

obj\_code = format4(Opnum\_Return(tmp2+1), 0x11, numtmp);

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: address range check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

else if(tmp3[0] == '@'){

numtmp = Is\_Used\_Sym(tmp3+1);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp2+1),0x21,numtmp);

mod\_list[mod\_cnt++] = pc\_num - loc\_save[0]+1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: address range check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else if(Is\_Used\_Sym(tmp3) != -1){

numtmp = Is\_Used\_Sym(tmp3);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp2+1),0x31,numtmp);

mod\_list[mod\_cnt++] = loc\_save[loc\_cnt] - loc\_save[0]+1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Symbol, X의 꼴. \*/

else{

strcpy(str\_tmp,tmp3);

strcat(str\_tmp,tmp4);

strcat(str\_tmp,tmp5);

lentmp = (int)strlen(str\_tmp);

if(str\_tmp[lentmp-1] == 'X' && str\_tmp[lentmp-2] == ','){

file\_lst = strtok(str\_tmp,",");

numtmp = Is\_Used\_Sym(file\_lst);

if(numtmp != -1 && Type\_Return(tmp2+1) == 3){

obj\_code = format4(Opnum\_Return(tmp2+1),0x39,numtmp);

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

/\* Byte 꼴. \*/

else if(strcmp(tmp2,"BYTE") == 0){

if(tmp3[0] == 'X'){

numtmp = Byte\_Check(tmp3);

}

else if(tmp3[0] == 'C'){

numtmp = Check\_C(buffer);

}

else{

printf("Error! Line %d: Invalid byte syntax!\n",line\_cnt);

error\_flag = 1;

continue;

}

if(numtmp != -1){

fprintf(fp2,"%3d\t%04X %-50s ",line\_cnt,loc\_save[loc\_cnt++],buffer);

if(tmp3[0] == 'C'){

sscanf(buffer,"%s %s %[^'] %\*c %[^']",tmp1,tmp2,tmp3,byte\_tmp1);

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

fprintf(fp2,"%2X",byte\_tmp1[i]);

}

fprintf(fp2,"\n");

if(op\_tmp + numtmp > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

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

fprintf(fp3,"%2X",byte\_tmp1[i]);

}

op\_tmp += numtmp;

}

else{

for(i=2; i<numtmp\*2+2; i++){

fprintf(fp2,"%c",tmp3[i]);

}

fprintf(fp2,"\n");

if(op\_tmp + numtmp > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

for(i=2; i<numtmp\*2+2; i++){

fprintf(fp3,"%c",tmp3[i]);

}

op\_tmp += numtmp\*2;

}

}

else{

printf("Error! Line %d: Byte format check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else if(strcmp(tmp2, "RESW") == 0){

fprintf(fp2,"%3d\t%04X %-50s\n",line\_cnt,loc\_save[loc\_cnt++],str);

if(op\_tmp != 0){

op\_tmp=0;

}

}

else if(strcmp(tmp2, "RESB") == 0){

fprintf(fp2,"%3d\t%04X %-50s\n",line\_cnt,loc\_save[loc\_cnt++],str);

if(op\_tmp != 0){

op\_tmp=0;

}

}

else if(strcmp(tmp2, "WORD") == 0){

if(tmp3[0] <= '9' && tmp3[0] >= '0'){

if(Is\_Dec(tmp3) == -1){

printf("Error! Line %d: Word format check!\n",line\_cnt);

}

sscanf(tmp3,"%d %c",&numtmp,&syn\_check);

if(numtmp != -1 && syn\_check == '\0'){

numtmp = atoi(tmp3);

if(numtmp >= 0x000000 && numtmp <= 0xFFFFFF){

fprintf(fp2,"%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,numtmp);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",numtmp);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Word format check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Word format check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Word format check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* 여기서부터는 LABEL 없는 경우.

\* Operand 위치에 순서대로 #, @

\* 들어올 경우. \*/

else{

if(Is\_Op(tmp1) == 1){

if(tmp2[0] == '#'){

if(Is\_Used\_Sym(tmp2+1) != -1){

numtmp = Is\_Used\_Sym(tmp2+1);

/\*Inside the range of PC addressing mode. \*/

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp1),0x12,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp1),0x14,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

sscanf(tmp2+1," %d %c",&numtmp,&syn\_check);

if(numtmp != -1 && syn\_check == '\0'){

if(numtmp >= -2048 && numtmp <= 4095){

obj\_code = format3(Opnum\_Return(tmp1),0x10,numtmp);

fprintf(fp2,"%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

else if(tmp2[0] == '@'){

if(Is\_Used\_Sym(tmp2+1) != -1){

numtmp = Is\_Used\_Sym(tmp2+1);

//pc\_num += 3;

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp1),0x22,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp1),0x24,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\* Operand 없는 opcode의 경우\*/

else if(tmp2[0] == '\0'){

if(Type\_Return(tmp1) == 1){

obj\_code = format1(Opnum\_Return(tmp1));

fprintf(fp2,"%3d\t%04X %-50s %02X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 1 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%02X",obj\_code);

op\_tmp += 1;

}

else if(Type\_Return(tmp1) == 2){

obj\_code = format2(Opnum\_Return(tmp1),0,0);

fprintf(fp2,"%3d\t%04X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%04X",obj\_code);

op\_tmp += 2;

}

else if(Type\_Return(tmp1) == 3){

obj\_code = format3(Opnum\_Return(tmp1),0x30,0);

fprintf(fp2,"%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else if(Is\_Used\_Sym(tmp2) != -1 && tmp3[0] == '\0'){

/\*Operand에 Symbol Table에 존재하는 Label 입력\*/

if(Type\_Return(tmp1) == 3){

numtmp = Is\_Used\_Sym(tmp2);

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp1),0x32,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp1),0x34,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

/\*레지스터 입력 또는 BUFFER,X의 형태\*/

if(Is\_Reg(tmp2) != -1){

if(Type\_Return(tmp1) == 2){

obj\_code = format2(Opnum\_Return(tmp1),Is\_Reg(tmp2),0);

fprintf(fp2,"%3d\t%04X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%04X",obj\_code);

op\_tmp += 2;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\*Symbol, X의 꼴\*/

else{

strcpy(str\_tmp,tmp2);

strcat(str\_tmp,tmp3);

strcat(str\_tmp,tmp4);

strcpy(str\_tmp2,str\_tmp);

lentmp = (int)strlen(str\_tmp);

regtmp1 = strtok(str\_tmp2,",");

regtmp2 = strtok(NULL,",");

if(Type\_Return(tmp1) == 2 && Is\_Reg(regtmp1) != - 1 && Is\_Reg(regtmp2) != -1){

obj\_code = format2(Opnum\_Return(tmp1),Is\_Reg(regtmp1),Is\_Reg(regtmp2));

fprintf(fp2, "%3d\t%-4X %-50s %04X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 2 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%04X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 2;

}

else if(str\_tmp[lentmp-1] == 'X' && str\_tmp[lentmp-2] == ','){

file\_lst = strtok(str\_tmp,",");

numtmp = Is\_Used\_Sym(file\_lst);

if(numtmp != -1 && Type\_Return(tmp1) == 3){

if((numtmp - pc\_num) >= -2048 && (numtmp - pc\_num) <= 2047){

obj\_code = format3(Opnum\_Return(tmp1),0x3A,numtmp-pc\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else if((numtmp - base\_num) >= 0 && (numtmp - base\_num) <= 4095 && baseflag == 1){

obj\_code = format3(Opnum\_Return(tmp1),0x3C,numtmp-base\_num);

fprintf(fp2, "%3d\t%04X %-50s %06X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 3 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%06X",obj\_code);

op\_tmp += 3;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

}

/\* (+)opcode의 형태의 입력.

\* 순서대로 #, @, Symbol 자체

\* 의 입력이 들어올 경우.\*/

else if(Is\_Op(tmp1) == 2){

if(tmp2[0] == '#'){

numtmp = Is\_Used\_Sym(tmp2+1);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp1+1), 0x11, numtmp);

mod\_list[mod\_cnt++] = loc\_save[loc\_cnt] - loc\_save[0] + 1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

sscanf(tmp2+1,"%d %c",&numtmp,&syn\_check);

if(numtmp != -1 && syn\_check == '\0'){

if(numtmp >= 0 && numtmp <= 0xFFFFF){

obj\_code = format4(Opnum\_Return(tmp1+1), 0x11, numtmp);

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: address range check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

else if(tmp2[0] == '@'){

numtmp = Is\_Used\_Sym(tmp3+1);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp1+1),0x21,numtmp);

mod\_list[mod\_cnt++] = loc\_save[loc\_cnt] - loc\_save[0] + 1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: address range check!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else if(Is\_Used\_Sym(tmp2) != -1){

numtmp = Is\_Used\_Sym(tmp2);

if(numtmp != -1){

obj\_code = format4(Opnum\_Return(tmp1+1),0x31,numtmp);

mod\_list[mod\_cnt++] = loc\_save[loc\_cnt] - loc\_save[0] + 1;

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

/\*Symbol, X의 꼴. \*/

else{

strcpy(str\_tmp,tmp2);

strcat(str\_tmp,tmp3);

strcat(str\_tmp,tmp4);

lentmp = (int)strlen(str\_tmp);

if(str\_tmp[lentmp-1] == 'X' && str\_tmp[lentmp-2] == ','){

file\_lst = strtok(str\_tmp,",");

numtmp = Is\_Used\_Sym(file\_lst);

if(numtmp != -1 && Type\_Return(tmp1+1) == 3){

obj\_code = format4(Opnum\_Return(tmp1+1),0x39,numtmp);

fprintf(fp2, "%3d\t%04X %-50s %08X\n",line\_cnt,loc\_save[loc\_cnt++],str,obj\_code);

if(op\_tmp + 4 > 30 || op\_tmp == 0){

fprintf(fp3,"\nT%06X%02X",loc\_save[loc\_cnt-1],op\_length[op\_cnt++]);

op\_tmp=0;

}

fprintf(fp3,"%08X",obj\_code);

op\_tmp += 4;

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag = 1;

continue;

}

}

}

/\* Base 명령어가 들어올 경우

\* Base flag 를 1로 set 하고

\* 그 옆에있는 LABEL의 LOCCTR을

\* base register로 저장. \*/

else if(strcmp(tmp1,"BASE") == 0){

baseflag=1;

if(Is\_Used\_Sym(tmp2) != -1){

base\_num = Is\_Used\_Sym(tmp2);

fprintf(fp2,"%3d\t%4s %-50s\n",line\_cnt," ",str);

}

else{

printf("Error! Line %d: Invalid operand!\n",line\_cnt);

error\_flag=1;

continue;

}

}

else{

printf("Error! Line %d: Object code doesn't exist!\n",line\_cnt);

error\_flag=1;

continue;

}

}

}

for(i=0; i<mod\_cnt; i++){

fprintf(fp3, "\nM%06X05",mod\_list[i]);

}

fprintf(fp3,"\nE%06X\n",loc\_save[0]);

Change\_Sym();

fclose(fp);

fclose(fp2);

fclose(fp3);

if(error\_flag == 1){

Free\_Symtab();

Reset\_Symbol();

strcpy(file\_tmp,filename);

file\_lst = strtok(file\_tmp, ".");

strcat(file\_lst,".lst");

remove(file\_lst);

file\_obj = strtok(file\_lst,".");

strcat(file\_obj,".obj");

remove(file\_obj);

}

else{

strcpy(file\_tmp,filename);

file\_lst = strtok(file\_tmp, ".");

printf("\toutput file : [%s.lst], [%s.obj]\n",file\_lst,file\_lst);

}

}

/\*name에 알맞는 opcode type을 리턴함 각각 1,2, 3/4\*/

int Type\_Return(char name[]){

int idx;

Opcode\_Table \*tmp;

if(name[0] == '\0')

return 0;

idx = opcodetable\_func\_make\_hash\_given(name);

for(tmp = Op\_Table[idx]; tmp != NULL; tmp = tmp->link){

if(strcmp(tmp->op\_mnemonic,name) == 0){

if(strcmp(tmp->type,"1") == 0){

return 1;

}

else if(strcmp(tmp->type,"2") == 0){

return 2;

}

else if(strcmp(tmp->type,"3/4") == 0){

return 3;

}

}

}

return 0;

}

/\* Symbol Table을 리셋한다. \*/

void Reset\_Symbol(void){

int i;

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

Sym\_Table[i] = NULL;

}

}

/\* Byte의 길이를 체크. \*/

int Byte\_Check(char str[]){

int len=-1,i;

int numlen=0;

int flag=0;

len = (int)strlen(str);

if(str[0] == 'C'){

if(str[1] == '\'' && str[len-1] == '\''){

numlen = len - 3;

return numlen;

}

}

else if(str[0] == 'X'){

if(str[1] == '\'' && str[len-1] == '\''){

if(len % 2 == 0){

return -1;

}

for(i=2; i<len-1; i++){

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

}

else{

flag=1;

}

}

if(flag == 0)

return (len-3)/2;

else

return -1;

}

else

return -1;

}

return -1;

}

/\* Symbol\_Table을 Decreasing Order로 새로운 배열에 저장한다. \*/

void Change\_Sym(void){

int i,j;

int count=0;

Symbol\_Table \*tmp;

Symbol\_Table dectmp;

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

if(Sym\_Table[i] != NULL){

for(tmp = Sym\_Table[i]; tmp != NULL; tmp = tmp->link)

count += 1;

}

}

Dec\_Table = (Symbol\_Table\*)malloc(sizeof(Symbol\_Table) \* count);

j=0;

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

if(Sym\_Table[i] != NULL){

for(tmp = Sym\_Table[i]; tmp != NULL; tmp = tmp->link){

strcpy(Dec\_Table[j].symbol,tmp->symbol);

Dec\_Table[j].locctr = tmp->locctr;

j+=1;

}

}

}

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

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

if(strcmp(Dec\_Table[j].symbol,Dec\_Table[j+1].symbol) < 0){

dectmp = Dec\_Table[j];

Dec\_Table[j] = Dec\_Table[j+1];

Dec\_Table[j+1] = dectmp;

}

}

}

}

int format1(int opcode){

return opcode;

}

/\* Register에 들어온 값들을 조사하고 이를 통해 format2 형식의 objcode를 계산해준다.\*/

int format2(int opcode,char reg1, char reg2){

int sum=0;

if(reg1 == 'A'){

reg1 = 0;

}

else if(reg1 == 'X'){

reg1 = 1;

}

else if(reg1 == 'L'){

reg1 = 2;

}

else if(reg1 == 'B'){

reg1 = 3;

}

else if(reg1 == 'S'){

reg1 = 4;

}

else if(reg1 == 'T'){

reg1 = 5;

}

else if(reg1 == 'F'){

reg1 = 6;

}

if(reg2 == 'A'){

reg2 = 0;

}

else if(reg2 == 'X'){

reg2 = 1;

}

else if(reg2 == 'L'){

reg2 = 2;

}

else if(reg2 == 'B'){

reg2 = 3;

}

else if(reg2 == 'S'){

reg2 = 4;

}

else if(reg2 == 'T'){

reg2 = 5;

}

else if(reg1 == 'F'){

reg1 = 6;

}

opcode <<= 8;

sum |= opcode;

reg1 <<= 4;

sum |= reg1;

sum |= reg2;

return sum;

}

/\* format3 형식의 objcode를 계산한다. \*/

int format3(int opcode,int nixbpe, int dis\_addr){

int sum=0;

dis\_addr &= 0xFFF;

opcode <<= 16;

sum |= opcode;

nixbpe <<= 12;

sum |= nixbpe;

sum |= dis\_addr;

return sum;

}

/\* Format4 형식의 objcode를 계산한다. \*/

int format4(int opcode,int nixbpe, int dis\_addr){

int sum=0;

dis\_addr &= 0xFFFFF;

opcode <<= 24;

sum |= opcode;

nixbpe <<= 20;

sum |= nixbpe;

sum |= dis\_addr;

return sum;

}

/\* Mnemonic의 Opcode를 출력해준다. \*/

int Opnum\_Return(char name[]){

int idx;

Opcode\_Table \*tmp;

idx = opcodetable\_func\_make\_hash\_given(name);

for(tmp = Op\_Table[idx]; tmp != NULL; tmp = tmp->link){

if(strcmp(name,tmp->op\_mnemonic) == 0)

return tmp->op\_num;

}

return -1;

}

/\* 입력받은 문자열이 Decimal 인지 확인. \*/

int Is\_Dec(char s[]){

int i;

int len = strlen(s);

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

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

return -1;

}

return 1;

}

/\* Byte에서 C''형식에 대한 체크를 수행한다.

\* 만약 틀린 형식일 경우 -1을 리턴한다. \*/

int Check\_C(char s[]){

int len;

char tmp1[20],tmp2[20],tmp3[20],check[40];

char e\_check1,e\_check2;

sscanf(s,"%s %s %[^'] %c %[^'] %c",tmp1,tmp2,tmp3,&e\_check1,check,&e\_check2);

len = (int)strlen(check);

if(e\_check1 == '\'' && e\_check2 == '\''){

return len;

}

else{

return -1;

}

}

/\* 문자열이 Register인지 검사한다.

\* 아닐 경우 -1을 리턴한다. \*/

int Is\_Reg(char s[]){

int flag = -1;

if(strcmp(s,"A") == 0){

flag = 0;

}

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

flag = 1;

}

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

flag = 2;

}

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

flag = 8;

}

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

flag = 9;

}

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

flag = 3;

}

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

flag = 4;

}

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

flag = 5;

}

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

flag = 6;

}

return flag;

}

/\* linking loader에서 pass1 부분을 담당하는 함수. ESTAB을 생성하고 duplicate symbol에

\* 대한 에러체크를 수행한다. ESTAB을 만들 때에는 program symbol과 label등을 구분하여 저장한다. \*/

void loader\_pass1(char filename1[], char filename2[], char filename3[]){

FILE \*fp1, \*fp2, \*fp3;

FILE \*fp;

char filename[3][30];

char strtmp[100];

char nametmp[10],addrtmp[10],lengthtmp[10];

int error\_flag=0;

int file\_cnt =0 ;

int i,j,k;

int proglen=0; //프로그램 길이

int startaddr=0; //시작 주소, 나중에 Define들을 받을 때 뺴줘야함

fp1 = fopen(filename1,"r");

if(fp1 != NULL){

file\_cnt += 1;

strcpy(filename[0],filename1);

}

fp2 = fopen(filename2,"r");

if(fp2 != NULL){

file\_cnt += 1;

strcpy(filename[1],filename2);

}

fp3 = fopen(filename3,"r");

if(fp3 != NULL){

file\_cnt += 1;

strcpy(filename[2],filename3);

}

for(k=0 ;k<file\_cnt; k++){

fp = fopen(filename[k],"r");

/\* Header 읽기\*/

while(fgets(strtmp,100,fp)){

memset(nametmp,'\0',10);

memset(addrtmp,'\0',10);

memset(lengthtmp,'\0',10);

nametmp[0] = '\0';

addrtmp[0] = '\0';

lengthtmp[0] = '\0';

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

for(i=1; i<7; i++){

if(strtmp[i] == ' ')

break;

nametmp[i-1] = strtmp[i];

}

nametmp[i-1] = '\0';

for(i=7; i<13; i++)

addrtmp[i-7] = strtmp[i];

for(i=13; i<19; i++)

lengthtmp[i-13] = strtmp[i];

if(S\_to\_Hex(addrtmp) == -1 || S\_to\_Hex(lengthtmp) == -1){

error\_flag = 1;

printf("Error1\n");

break;

}

if(Find\_Estab(nametmp) != -1){

printf("Duplicated Symbol\n");

return ;

}

/\*Make\_Estab들어가기 전에 이미 쓰인 Symbol인지 검사 필요. \*/

Make\_Estab(Header,nametmp,S\_to\_Hex(addrtmp),S\_to\_Hex(lengthtmp));

break;

}

}

startaddr = S\_to\_Hex(addrtmp);

proglen = S\_to\_Hex(lengthtmp);

while(fgets(strtmp,100,fp)){

memset(nametmp,'\0',10);

memset(addrtmp,'\0',10);

memset(lengthtmp,'\0',10);

nametmp[0] = '\0';

addrtmp[0] = '\0';

lengthtmp[0] = '\0';

i=1;

j=0;

/\* D 레코드 읽기. \*/

if(strtmp[0] == 'D'){

while(i+11 <= strlen(strtmp)){

for(j=0; j<6; j++,i++){

if(strtmp[i] == ' ')

nametmp[j] = '\0';

else

nametmp[j] = strtmp[i];

}

nametmp[j] = '\0';

for(j=0; j<6; j++,i++){

addrtmp[j] = strtmp[i];

}

/\* 유효한 16진수 타입인지 에러체크. \*/

if(S\_to\_Hex(addrtmp) == -1){

error\_flag = 1;

printf("Error2\n");

break;

}

if(Find\_Estab(nametmp) != -1){

printf("Duplicated Symbol\n");

return ;

}

Make\_Estab(Define,nametmp,S\_to\_Hex(addrtmp) - startaddr,S\_to\_Hex(lengthtmp));

}

}

/\* T레코드 부터는 더이상 pass1에서 읽을 필요가 없음. \*/

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

progaddr += proglen;

break;

}

}

fclose(fp);

}

/\* ESTAB 출력. \*/

Print\_Load();

if(error\_flag == 0){

for(k=0; k<file\_cnt; k++){

loader\_pass2(filename[k]);

}

}

}

/\* linking loader의 pass2 부분에 해당하는 함수. 에러체크는

\* ESTAB에 존재하지 않는 symbol을 참조하는 경우를 체크하였다.

\* 이후 run등에도 사용될 progaddr나 proglength 등을 저장한다. \*/

void loader\_pass2(char filename[]){

FILE \*fp;

int csaddr; //프로그램 시작주소

int cslength=0; //프로그램 길이

int address; //주소값

int reference\_num; //reference number에 사용할 배열과 index

int reference\_arr[100];

int loctmp;

int len;

int t\_len;

int obj\_code;

int modifi\_num; //modifi num을 저장한다.

int immediate\_num; //bit mask 등에 사용할 변수.

int edit\_num;

int i;

int error\_flag=0;

int idx;

char modifi\_type;

char nametmp[10];

char strtmp[100];

char \*str;

Estab\_Table \*tmp;

csaddr = progaddr;

fp = fopen(filename,"r");

if(fp == NULL){

printf("File open error!\n");

return ;

}

while(fgets(strtmp,100,fp)){

str = strtok(strtmp,"\n");

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

sscanf(str+1,"%6x",&address);

break;

}

else if(str[0] == '.'){

continue;

}

else if(str[0] == 'H'){

sscanf(str+1,"%6s",nametmp);

loctmp = Find\_Estab(nametmp);

if(loctmp == -1){

error\_flag = 1;

printf("Symbol doesn't exist!\n");

break;

}

idx = opcodetable\_func\_make\_hash\_given(nametmp);

for(tmp = Es\_Table[idx]; tmp != NULL; tmp = tmp->link){

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

cslength = tmp->length;

break;

}

}

reference\_arr[1] = loctmp;

}

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

if((strlen(str)-1) % 8 == 0)

len = (strlen(str)-1) / 8;

else

len = ((strlen(str)-1) / 8) + 1;

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

sscanf(str+i\*8+1,"%2x%6s",&reference\_num,nametmp);

loctmp = Find\_Estab(nametmp);

if(loctmp == -1){

printf("Symbol doesn't exist!\n");

error\_flag = 1;

break;

}

reference\_arr[reference\_num] = loctmp;

}

}

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

sscanf(str+1,"%6x%2x",&address,&t\_len);

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

sscanf(str+9+i\*2,"%2x",&obj\_code);

memory[i+csaddr+address] = obj\_code;

}

}

/\* M타입일 경우 Format4의 05bit modify와 06bit modify를 구분해 계산한다.\*/

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

sscanf(str+1,"%6x%2x%c%x",&address,&modifi\_num,&modifi\_type,&reference\_num);

if(reference\_num < 1){

printf("Undefined Reference Number!\n");

error\_flag = 1;

break;

}

else{

immediate\_num = 0;

if(modifi\_num % 2 == 1){

immediate\_num += memory[csaddr+address]%16;

}

for(i=csaddr + address + modifi\_num%2; i<csaddr+address+3; i++){

immediate\_num <<= 4;

immediate\_num += memory[i] / 0x10;

immediate\_num <<= 4;

immediate\_num += memory[i] % 0x10;

}

if(modifi\_type == '+'){

immediate\_num += reference\_arr[reference\_num];

}

else if(modifi\_type == '-'){

immediate\_num -= reference\_arr[reference\_num];

}

else{

error\_flag = 1;

printf("Syntax Error! '+' or '-' should be used!\n");

break;

}

if(modifi\_num == 5){

edit\_num = 0xFFFFF;

}

else if(modifi\_num == 6){

edit\_num = 0xFFFFFF;

}

else{

error\_flag = 1;

printf("Modification number has to be 5 or 6!\n");

break;

}

immediate\_num &= edit\_num;

for(i=2; i>=modifi\_num%2; i--){

memory[i+csaddr+address] = immediate\_num%0x100;

immediate\_num >>= 8;

}

if(modifi\_num%2 == 1){

memory[csaddr+address] /= 16;

memory[csaddr+address] \*= 16;

memory[csaddr+address] += immediate\_num % 0x10;

}

}

}

else{

continue;

}

}

progaddr += cslength;

fclose(fp);

if(error\_flag == 1){

printf("Load Fail!\n");

}

}

/\*ESTAB을 생성하는 함수. 이 때 Hash function을 사용해 symbol을 찾기 편하게 한다. \*/

void Make\_Estab(int typetmp, char nametmp[], int addrtmp, int lentmp){

int idx;

Estab\_Table \*tmp, \*new;

new = (Estab\_Table\*)malloc(sizeof(Estab\_Table));

new->type = typetmp;

strcpy(new->name,nametmp);

new->addr = /\*S\_to\_Hex(addrtmp) + \*/progaddr;

new->startaddr = addrtmp;

new->length = lentmp;

new->link = NULL;

if(new->type == Define)

new->addr += addrtmp;

idx = opcodetable\_func\_make\_hash\_given(nametmp);

if(Es\_Table[idx] == NULL){

Es\_Table[idx] = new;

}

else{

for(tmp = Es\_Table[idx]; tmp->link != NULL; tmp = tmp->link)

;

tmp->link = new;

}

}

/\* ESTAB에서 적절한 주소값을 찾아 이를 리턴해주는 함수. \*/

int Find\_Estab(char name[]){

int idx;

Estab\_Table \*tmp;

idx = opcodetable\_func\_make\_hash\_given(name);

for(tmp = Es\_Table[idx]; tmp != NULL; tmp = tmp->link){

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

return tmp->addr;

}

}

return -1;

}

/\* 문자열을 입력받아 적절한 format인지 확인하고, 이를 16진수로 바꿔주는 함수. \*/

int S\_to\_Hex(char strtmp[]){

int i;

int hex=0;

int digit=1;

for(i=5; i>=0; i--){

if(strtmp[i] <= '9' && strtmp[i] >= '0'){

hex += (strtmp[i]-'0') \* digit;

digit \*= 16;

}

else if(strtmp[i] <= 'F' && strtmp[i] >= 'A'){

hex += (strtmp[i]-55) \* digit;

digit \*= 16;

}

else

return -1;

}

return hex;

}

/\* Loading한 후 ESTAB을 출력해주는 함수. \*/

void Print\_Load(void){

int i,j;

int count=0;

int totallen=0;

Estab\_Table \*tmp,ordered[300],forswap;

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

if(Es\_Table[i]!=NULL){

for(tmp = Es\_Table[i]; tmp != NULL; tmp = tmp->link){

ordered[count].type = tmp->type;

strcpy(ordered[count].name,tmp->name);

ordered[count].addr = tmp->addr;

ordered[count].length = tmp->length;

count++;

}

}

}

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

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

if(ordered[j].addr > ordered[j+1].addr){

forswap = ordered[j];

ordered[j] = ordered[j+1];

ordered[j+1] = forswap;

}

}

}

printf("%-15s%-15s%-15s%-15s\n","control","symbol","address","length");

printf("%-15s%-15s%-15s%-15s\n","section","name"," "," ");

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

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

if(ordered[i].type == Header){

printf("%-15s%-15s%04X%-11s%04X\n",ordered[i].name," ",ordered[i].addr," ",ordered[i].length);

totallen += ordered[i].length;

}

else if(ordered[i].type == Define){

printf("%-15s%-15s%04X%-11s%15s\n"," ",ordered[i].name,ordered[i].addr," "," ");

}

else

printf("Print Error!\n");

}

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

printf("%-30s%-15s%04X\n"," ","total length",totallen);

progaddr -= totallen;

}

/\* ESTAB을 Free해주는 함수. \*/

void Free\_Estab(void){

int i;

Estab\_Table \*tmp;

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

while((tmp = Es\_Table[i]) != NULL){

Es\_Table[i] = Es\_Table[i]->link;

free(tmp);

}

}

}

/\* Bp를 출력해주는 함수. \*/

void Print\_Bp(void){

Symbol\_Table \*tmp;

printf("breakpoint\n");

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

for(tmp = bp\_list; tmp != NULL; tmp = tmp->link){

printf("%04X\n",tmp->locctr);

}

}

/\* Bp를 비워주는 함수. \*/

void Clear\_Bp(void){

Symbol\_Table \*tmp;

while((tmp = bp\_list) != NULL){

bp\_list = bp\_list -> link;

free(tmp);

}

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

}

/\*A: 0

\*X: 1

\*L: 2

\*B: 3

\*S: 4

\*T: 5

\*F: 6

\*PC: 8

\*SW: 9

위의 레지스터 번호를 사용한다. \*/

/\* memory에 레지스터에 들어있는 값을 적절하게 넣어주는 함수. \*/

void Make\_Num(int reg[], int num, int \*ta){

memory[\*ta] = (reg[num] & 0x00FF0000) >> 16;

memory[\*ta+1] = (reg[num] & 0x0000FF00) >> 8;

memory[\*ta+2] = (reg[num] & 0x000000FF);

}

/\* 프로그램을 break point까지 실제로 돌리는 함수. if else 문장 안에 들어있는 조건들은

\* 각각 opcode들이며, 해당 opcode에 따라 명령을 수행한다. \*/

void Run\_Prog(void){

//t loc;

int op\_number;

static int reg[10]={0,}; //register들을 저장하는 배열, 각각 index가 register의 번호를 의미한다.

int ta=0, reg1=0, reg2=0; //각각 target address, register1, register2를 저장한다.

int i;

int swtmp; //sw register에 들어갈 값을 임시로 저장하는 변수.

if(firsttry == 0){

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

reg[i]=0;

}

reg[8] = runstartaddr;

reg[2] = runfinaladdr;

firsttry = 1;

}

while(reg[8] < runendaddr){

op\_number = memory[reg[8]] & 0xfc;

if(op\_number == 0x14){

TA\_Format3(reg,&ta);

Make\_Num(reg,2,&ta);

}

else if(op\_number == 0x68){

reg[3] = TA\_Format3(reg,&ta);

}

else if(op\_number == 0x48){

TA\_Format3(reg,&ta);

reg[2] = reg[8];

reg[8] = ta;

}

else if(op\_number == 0x00){

reg[0] = TA\_Format3(reg,&ta);

}

else if(op\_number == 0x28){

swtmp = TA\_Format3(reg,&ta);

if(reg[0] > swtmp){

reg[9] = 1;

}

else if(reg[0] == swtmp){

reg[9] = 0;

}

else{

reg[9] = -1;

}

}

else if(op\_number == 0x30){

TA\_Format3(reg,&ta);

if(reg[9] == 0){

reg[8] = ta;

}

}

else if(op\_number == 0x3c){

TA\_Format3(reg,&ta);

reg[8] = ta;

}

else if(op\_number == 0x0c){

TA\_Format3(reg,&ta);

Make\_Num(reg,0,&ta);

}

else if(op\_number == 0xb4){

TA\_Format2(reg,&reg1,&reg2);

reg[reg1]=0;

}

else if(op\_number == 0x74){

reg[5] = TA\_Format3(reg,&ta);

}

else if(op\_number == 0xe0){

TA\_Format3(reg,&ta);

reg[9] = 1;

}

else if(op\_number == 0xd8){

TA\_Format3(reg,&ta);

reg[9] = 0;

}

else if(op\_number == 0xa0){

TA\_Format2(reg,&reg1,&reg2);

if(reg[reg1] > reg[reg2]){

reg[9] = 1;

}

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

reg[9] = 0;

}

else{

reg[9] = -1;

}

}

else if(op\_number == 0x54){

TA\_Format3(reg,&ta);

memory[ta] = reg[0] % 256;

}

else if(op\_number == 0xb8){

reg[1] += 1;

TA\_Format2(reg,&reg1,&reg2);

if(reg[1] > reg[reg1]){

reg[9] = 1;

}

else if(reg[1] == reg[reg1]){

reg[9] = 0;

}

else{

reg[9] = -1;

}

}

else if(op\_number == 0x38){

TA\_Format3(reg,&ta);

if(reg[9] == -1){

reg[8] = ta;

}

}

else if(op\_number == 0x10){

TA\_Format3(reg,&ta);

Make\_Num(reg,1,&ta);

}

else if(op\_number == 0x4c){

TA\_Format3(reg,&ta);

reg[8] = reg[2];

}

else if(op\_number == 0x50){

reg[0] &= 0xFFFFFF00;

TA\_Format3(reg,&ta);

reg[0] += memory[ta];

}

else if(op\_number == 0xdc){

TA\_Format3(reg,&ta);

reg[9]=0;

}

else{

reg[8] += 1;

}

}

printf("\t\tA : %06X X : %06X\n",reg[0],reg[1]);

printf("\t\tL : %06X PC: %06X\n",reg[2],reg[8]);

printf("\t\tB : %06X S : %06X\n",reg[3],reg[4]);

printf("\t\tT : %06X\n",reg[5]);

}

/\* Format 3형식의 Target Address를 계산해주는 함수. \*/

int TA\_Format3(int reg[],int \*ta){

int nixbpe,disp=0;

nixbpe = ((int)(memory[reg[8]] & 0x03) << 4) + ((int)(memory[reg[8]+1] & 0x0F0) >> 4);

disp = (int)((memory[reg[8]+1] & 0x0F) << 8) + (int)(memory[reg[8]+2] & 0xFF);

if((nixbpe & 0x01) != 0){

disp <<= 8;

disp |= memory[reg[8]+3];

}

//3형식과 4형식의 경우 구분해서 disp 만들어줌

if(((nixbpe & 0x01) == 0) && ((disp & 0x800) != 0)){

disp = (~disp);

disp+=1;

disp &= 0x07FF;

disp \*= -1;

}

else if(((nixbpe & 0x01) != 0) && ((disp & 0x80000) != 0)){

disp = (~disp);

disp+=1;

disp &= 0x07FFFF;

disp \*= -1;

}

\*ta = disp;

//X Register가 1로 set된 경우.

if((nixbpe & 0x08) != 0){

\*ta += reg[1];

}

//n,i 비트가 둘다 0 일경우 -> SIC

if((nixbpe & 0x20) == 0 && (nixbpe & 0x10) == 0){

\*ta += (int)((memory[reg[8]] & 0xF0) << 8);

reg[8] += 3;

return (memory[\*ta]<<16) + (memory[\*ta+1]<<8) + (memory[\*ta+2]);

}

else{//각각 3형식 4형식.

if((nixbpe & 0x01) == 0){

reg[8] = reg[8] + 3;

}

else{

reg[8] = reg[8] + 4;

}

//PC relative와 base relative 구분.

if((nixbpe & 0x02) != 0){

\*ta += reg[8];

}

else if((nixbpe & 0x04) != 0){

\*ta += reg[3];

}

//ni가 11 -> simple

if(((nixbpe & 0x20) != 0) && ((nixbpe & 0x10) != 0)){

return (memory[\*ta]<<16) + (memory[\*ta+1]<<8) + (memory[\*ta+2]);

}

//ni가 01 -> immediate

else if(((nixbpe & 0x20) == 0) && ((nixbpe & 0x10) != 0)){

return \*ta;

}

//ni가 10 -> indirect

else{

\*ta = (memory[\*ta]<<16) + (memory[\*ta+1]<<8) + (memory[\*ta+2]);

return (memory[\*ta]<<16) + (memory[\*ta+1]<<8) + (memory[\*ta+2]);

}

}

}

/\* Format2의 Target Address를 계산해준다. \*/

int TA\_Format2(int reg[],int \*reg1, int \*reg2){

\*reg1 = (memory[reg[8]+1] & 0xf0)>>4;

\*reg2 = (memory[reg[8]+1] & 0x0f);

reg[8] += 2;

return 1;

}

void Make\_Bplist(int baddr){

Symbol\_Table \*new, \*tmp;

new = (Symbol\_Table\*)malloc(sizeof(Symbol\_Table));

new->link = NULL;

new->locctr = baddr;

if(bp\_list == NULL){

bp\_list = new;

}

else if(bp\_list->link == NULL){

if(bp\_list->locctr < new->locctr){

bp\_list->link = new;

}

else{

new->link = bp\_list;

bp\_list = new;

}

}

else{

for(tmp = bp\_list; tmp->link != NULL; tmp = tmp->link){

if(baddr < tmp->link->locctr){

break;

}

}

new->link = tmp->link;

tmp->link = new;

}

}