과목 명: 시스템프로그래밍

담당 교수 명: 김 지 환

<<Project 3>>

**서강대학교 컴퓨터공학과**

**[학번]20141556**

**[이름]이성문**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 3

3.1 모듈 이름 : addes 3

3.1.1 기능 3

3.1.2 사용 변수 3

3.2 모듈 이름: add\_ref 3

3.2.1 기능 3

3.2.2 사용 변수 3

3.3 모듈이름: add\_num\_memory 4

3.3.1 기능 4

3.3.2 사용변수 4

3.4 모듈이름: run 4

3.4.1 기능 4

3.4.2 사용변수 4

3.5 모듈이름: Mnemonic 명령어들 4

3.5.1 기능 4

3.5.2 사용변수 4

4. 전역 변수 정의 4

4.1 각종 구조체들 4

4.2 memoryend, tempend

4.3 각종 register 값들 4

4.4 각종 flag들 5

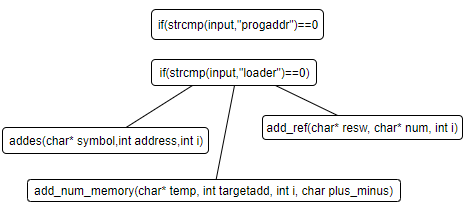
# 프로그램 개요

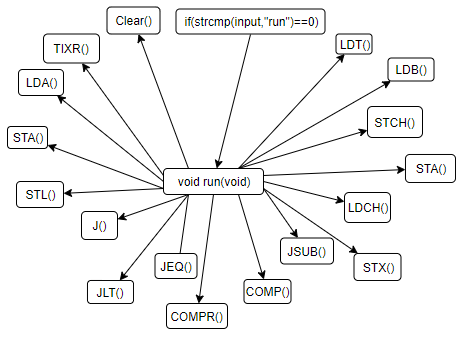
프로젝트 #1, #2에서 구현한 셀(shell)에 Linking과 loading 기능을 추가하는 프로그램이다. 프로젝트 #2에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모리에 올리는 일을 수행한다.

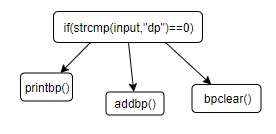
# 프로그램 설명

## 프로그램 흐름도

그림 1> 프로그램 흐름도







중심이 되는 strcmp는 main()에 존재하며 위와 같이 여러 함수들을 호출한다. 먼저 Input이 progaddr일 경우 그 address를 프로그램 시작 초기값으로 설정한다. 그리고 Input이 loader일 경우의 if절로 들어가 프로그램을 메모리에 로딩한다. object파일 세 개를 파일입력으로 받으며 object파일 줄 맨앞의 문자에 따라 다른 요청을 처리한다. 외부참조 symbol가 내부에 있는 symbol을 es(external table이란 뜻) table에 추가한다. (addes()) 그리고 M(modification)일 경우 add\_num\_memory함수를 호출에 메모리값을 수정해준다.

Input이 run일 경우 프로그램을 실행해주는 역할을 하는데 이때 메모리값을 1바이트씩 불러들여 안에 저장된 코드값을 실행시켜주고 다음 메모리 값을 다시 불러오는 역할을 해준다. 여기서 우리는 이미 opcode값들을 알고있기 때문에 코드가 어떤 명령어를 나타내는지 알 수 있다. 그리고 각 명령어마다 처리하는 레지스터들이 다르므로 주요 명령어마다 지역함수를 생성하여 레지스터를 처리하게 하였다. Input이 bp일 경우 breakpoint를 설정하는 (addbp()), breakpoint를 출력하고 (printbp()) , breakpoint 목록을 초기화시켜준다.

# 모듈 정의

## 모듈 이름 : addes(char\* symbol, int address, int i)

### 기능

External symbol table에 parameter로 받는 symbol과 address값을 저장해준다. I 는 i번째 파일의 es table을 뜻한다. Es table은 Linked list의 자료구조를 이용하였다.

### 사용 변수

Symbol\* new : 전달받은 데이터들을 저장할 새로운 노드이다.

ESTAB\* es 변수 안에 symbol\* head 변수가 있어서 head에 linked list로 추가해준다.

## 모듈 이름: add\_ref(char\* new, char\* num, int i)

### 기능

이 함수는 Es table에 외부 참조(external reference ) symbol들을 저장해주는 역할을 한다. New는 변수 이름이고 num은 외부 참조 번호이다. I는 위 함수와 같다.

### 사용 변수

Ref\* renw : table에 새로 추가할 외부참조 변수이다.

Es[i].r\_head가 es 구조체안에 저장되있는 외부 참조 Linked list의 head이다.

## 모듈이름: add\_num\_memory(char\* temp, int targetadd, int I, char plus\_minus)

### 기능

이 함수는 Modification record의 내용을 메모리에서 수정해주는 역할을 한다. Targetadd가 수정할 주소이다. Plus\_minus는 + 혹은 –이다. targetaddress에서 주소값을 불러와 거기안에 내용이랑 수정할 내용을 더하거나 빼준다. 그리고 다시 메모리에 넣어준다. 외부 참조할 경우 그 문자열을 es table에서 검색해 그 주소값을 알아내 계산에 포함시켜준다.

### 사용변수

Char original[7] : 주소안에 저장되있는 원래의 메모리값이다.

Ref\* rtemp : 리스트를 탐색할 임시 노드이다.

Int add\_resw\_loc : 원하는 외부 변수를 es table에서 검색해 그 변수의 location값이다.

Char add\_address[7] : 더할 주소값이다. 문자열이다.

## 모듈이름: void run(void)

### 기능

Loader에 의해 로딩된 메모리를 읽어서 프로그램을 실행시켜주는 역할을 한다. 메모리를 불러들여와 opcode값을 알아내고 그 opcode값에 맞는 명령어의 함수로 들어가 레지스터값들을 변경시켜주는 역할을 한다. ..

### 사용변수

int Realstart : Run이 처음으로 실행됐을 경우 아직1이다. 0으로 설정된다.

Int tempend : breakpoint이다

Int memoryend : 마지막 location 값이다.

Int plus : 반복문안에서 PC값을 변화시키는 역할을 한다.

Int beforePC : breakpoint에서 PC값을 출력시킬 때 변하기 전 PC값을 출력하기 위한 변수이다.

Char now\_m\_value[3] : 현재 PC값에 해당하는 메모리의 내용이므로 한 바이트 내용이다. 이 내용을 가지고 opcode를 알아낸다.

## 모듈이름: Clear(), TIXR(), LDA(), STA(), STL(), J(), JLT(), JEQ(), COMPR(), COMP(), JSUB(), LDCH(), STX(), STA(), STCH(), LDB(), LDT()

### 기능

Clear는 메모리값에 해당하는 number의 register값을 0으로 초기화시켜준다. TIXR은 X레지스터에 1을 더한후 number에 해당하는 레지스터 값과 비교해서 같으면 0, 작으면 -1, 크면 1을 리턴해준다. LDA,STA,STL,LDCH,STX,STA,STCH,LDB,LDT는 모두 각자 해당되는 레지스터를 사용해 load하거나 register에 store하는 역할을 한다. J, JLT,JEQ는 L 레지스터를 이용해 PC값에 변화를 주어 location 을 변화시켜 이동하는 역할을 한다. COMPR, COMP는 두 메모리 값을 비교해 TIXR과 마찬가지로 결과를 리턴해준다.

### 사용변수

Int ni : Opcode를 구하는 도중 ni값을 구할 수 있다. 0 : SIC , 1 : immediate, 2 : indirect, 3: simple

Int format : 그 코드의 format을 표현한다.

Int x,b,p,e : 각각의 비트들을 표현한다.

Int ta : target address이다.

Int disp : displacement값이다.

Char temp\_t[3], temp[3] : 메모리에 저장할 임시 배열들이다.

## 모듈이름 : printbp()

### 기능

Bphead에 저장된 리스트를 형식에 맞게 모두 출력해준다.

### 사용변수

Bptb\* bphead : breakpoint가 저장되있는 자료구조의 head이다.

## 모듈이름 : bpclear()

### 기능

Bphead list에 저장된 모든 breakpoint를 없애주는 역할을 한다.

### 사용변수

Bptb\* bphead : breakpoint가 저장되있는 자료구조의 head이다.

Bptb\* temp : breakpoint를 메모리 해제할 임시 노드이다.

## 모듈이름 : addbp(char bp[7])

### 기능

Parameter로 받은 문자열을 breakpoint로 자료구조안에 저장해주는 역할을 한다. 이떄 저장할 때, breakpoint를 오름차순으로 저장한다. 그래야만 순차적으로 break할 수 있다 생각하였다.

### 사용 변수

Bptb\* new : breakpoint를 저장할 새로운 노드이다.

Bptb\* bphead : breakpoint가 저장되있는 자료구조의 head이다.

## 모듈 이름 : printsymbol(void)

### 기능

Symbol table 에 저장된 데이터들을 순서대로 일정한 형식에 맞게 출력해준다.

### 사용 변수

Symbol\* temp: 데이터들을 출력하기 위해 저장할 임시 노드이다.

# 전역 변수 정의

## Struct ref

typedef struct \_REF{

char name[7]; : 외부참조 변수의 이름이다.

int ref\_num; : 외부참조 변수의 숫자이다. Reference num이다.

struct \_REF\* link;

}ref;

## Struct ESTAB

typedef struct{

char progname[10]; : prog 이름이다.

int address; : prog의 주소값이다.

int length; : prog의 코드 길이이다.

symbolt\* head; : 내부 참조 Head 포인터이다.

ref\* r\_head; : 외부 참조 symbol의 Head 포인터이다.

}ESTAB;

## Struct bptb

typedef struct \_BPTB{

char breakpoint[7]; : breakpoint 내용이다. 값. 16진수이다.

struct \_BPTB\* link;

}bptb;.

## ****Int memoryend** : 그 **프로그램의** location의 끝의 값을 나타낸다.**

## Int tempend : breakpoint당할 때 필요한 변수이다. Breakpoint를 나타낸다.

## Int A : A register 값

## Int X : X register 값

## Int L : L register 값

## Int PC : PC register 값

## Int B : B register 값

## Int S : S register 값

## Int T : T register 값

## Int flag : Comp, Compr, TIXR 에서 연산한 결과를 저장할 변수

## Int realend : 더 이상 breakpoint가 없어 정말 프로그램의 끝일 때 1

## Int realstart : run에 의해 프로그램이 처음으로 실행될때만 1

## Int progAddr : 명령어 progaddr에 의해 설정되는 프로그램 초기 주소값이다.

# 코드

///메인 안에 있는 새로 구현한 코드들이다.

else if(strcmp(input,"loader")==0)//loader 명령어

{

erflag=0;//초기화

char name[3][15];

char junk[10];

junk[0]='\0';

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

name[1][0]='\0';

name[2][0]='\0';

filenum=0;//입력되는 파일 개수

sscanf(input2,"%s %s %s %s %s",input,name[0],name[1],name[2],junk);

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

{

printf("Error\n");

continue;

}

if(name[1][0]=='\0')

filenum=1;

else if(name[2][0]=='\0')

filenum=2;

else

filenum=3;

if(junk[0]!='\0')

{

printf("Error\n");

continue;

}

es=(ESTAB\*)malloc(sizeof(ESTAB)\*filenum);

int totallength=0;//total length이다.

for(i=0;i<filenum;i++)//파일 개수만큼 읽어들인다.

{

FILE\* p=fopen(name[i],"r");

if(p==NULL)

{

printf("No %s FILE!!\n",name[i]);

erflag=1;

continue;

}

char H;

fscanf(p,"%c",&H);

if(H!='H')

{

printf("No Head Error\n");

continue;//다음파일로 스킴

}

char\* progname;//program name

progname=(char\*)malloc(sizeof(char)\*10);

fscanf(p,"%s",progname);

progname=rmspace(progname);//빈공간이 있을 수 있으니 빈칸 제거

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

{

if(strcmp(es[k].progname,progname)==0)

{

erflag=1;

break;

}

}

if(erflag==1)

{

printf("The program name is already existed.\n");

break;

}

strcpy(es[i].progname,progname);

char temp[7];

fscanf(p,"%6s",temp);//6자리만큼 받는다

int startaddress=to10(temp);//주소 출발값

fscanf(p,"%6s",temp);

int programlength;

programlength=to10(temp);//obj파일 길이

char dump;

fscanf(p,"%c",&dump);

char dumpstr[100];

es[i].address=startaddress+progAddr;//SET CSADDR

es[i].length=programlength;//SET LENGTH

es[i].head=NULL;

es[i].r\_head=NULL;

while(1)

{

fscanf(p,"%c",&H);

if(H=='E')//END일경우

break;

else if(H=='D')//D record 일 경우

{

while(1){

char\* resw;

resw=(char\*)malloc(sizeof(char)\*7);

int resw\_add;

resw[0]='\*';

fscanf(p,"%6s%6s",resw,temp);

resw=rmspace(resw);

resw\_add=to10(temp);//그 변수의 주소 입력

resw\_add+=es[i].address;//초기 주소 더해준다.

fscanf(p,"%c",&dump);//끝일 때 브레이크

addes(resw,resw\_add,i);//ES TABLE에 추가

if(dump=='\n')

break;

else

{

fseek(p,-1,1);

}

}

}

else if(H=='T')//T record 일 경우

{

fscanf(p,"%[^\n]",dumpstr);

fscanf(p,"%c",&dump);

}

else if(H=='M')//Modification record 일 경우

{

fscanf(p,"%[^\n]",dumpstr);

fscanf(p,"%c",&dump);

}

else if(H=='R')

{

while(1){

char\* rtemp;

char\* resw;

resw=(char\*)malloc(sizeof(char)\*7);

rtemp=(char\*)malloc(sizeof(char)\*3);

fscanf(p,"%2s",rtemp);

fscanf(p,"%6s",resw);

resw=rmspace(resw);

add\_ref(resw,rtemp,i);

fscanf(p,"%c",&dump);

if(dump=='\n')

break;

else

fseek(p,-1,1);

}

}

else

{

fscanf(p,"%[^\n]",dumpstr);

fscanf(p,"%c",&dump);

}

}

progAddr=es[i].address+es[i].length;

totallength+=es[i].length;

fclose(p);

}

if(erflag==1)

continue;

printf("Control Symbol\n");

printf("section name Address Length\n");

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

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

{

printf("%s %04X %04X\n",es[i].progname,es[i].address,es[i].length);

symbolt\* temp;

temp=es[i].head;

while(temp!=NULL)

{

printf(" %s %04X\n",temp->symbol,temp->loc);

temp=temp->link;

}

}

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

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

memoryend=es[0].address+totallength-1;

//////////////////////////////여기까지 table에 저장

//이제부터 Pass2

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

{

FILE\* p=fopen(name[i],"r");

char dumpstr[50];

char dump;

char H;

fscanf(p,"%[^\n]",dumpstr);

fscanf(p,"%c",&dump);

while(1)

{

fscanf(p,"%c",&H);

if(H=='T')//T recorde일때 메모리에 로딩

{

int startadd;

char temp[7];

int linelength;

fscanf(p,"%6s",temp);

startadd=to10(temp);//시작 주소

startadd+=es[i].address;

fscanf(p,"%2s",temp);

linelength=to10(temp);//줄 길이

for(int k=0;k<linelength;k++)

{

fscanf(p,"%2s",temp);

m[startadd+k][0]=temp[0];

m[startadd+k][1]=temp[1];

}//Loading 작업이다.

fscanf(p,"%c",&dump);

}

else if(H=='E')

break;

else if(H=='M')

{

int targetadd;//변경할 주소값

char temp[7];

char plus\_minus;//+ 혹은 -

fscanf(p,"%6s",temp);

targetadd=to10(temp);

targetadd+=es[i].address;

fscanf(p,"%\*c%\*c%c",&plus\_minus);

fscanf(p,"%2s",temp);

if(temp[0]<='9')//숫자로 들어왔을 경우

{

add\_num\_memory(temp,targetadd,i,plus\_minus);//메모리 수정

}

else//symbol 그 자체가 들어왔을 경우

{

//add\_sym\_memory(temp,targetadd,i);//메모리 수정

}

fscanf(p,"%c",&dump);

}

else

{

fscanf(p,"%[^\n]",dumpstr);

fscanf(p,"%c",&dump);

}

}

}

//L=memoryend;

addh(input2);

}

else if(strcmp(input,"run")==0)//run 명령어

{

if(realstart==1)//처음 시작할경우

L=memoryend;

if(strcmp(inputn,"run")!=0)//run말고 또다른 뒤에 명령어가 잇을경우

{

printf("Error\n");

continue;

}

if(es==NULL)//loader를 하지않았을때

{

printf("Error. No loaded Memory.\n");

continue;

}

run();

addh(input2);

}

else if(strcmp(input,"bp")==0)//breakpoint 설정 명령어

{

char bp[6];

sscanf(input2,"%s %s",input,bp);

if(strcmp(inputn,"bp")==0)//bp만 입력됐을 경우

{

printbp();

continue;

}

else if(strcmp(bp,"clear")==0)//bp clear

{

bpclear();

continue;

}

addbp(bp);

A=0;

X=0;

L=0;

PC=0;

S=0;

T=0;

bptemp=bphead;

realstart=1;

realend=0;

addh(input2);

}

else//올바르지않은 입력일 경우 에러

{

printf("Error\n");

}

}

deletem();//메모리 해제

return 0;

}

/////여기서부터 지역함수이다.

void addes(char\* symbol,int address,int i)//external symbol table에 추가해주는 함수이다.

{

symbolt\* new;

new=(symbolt\*)malloc(sizeof(symbolt));

strcpy(new->symbol,symbol);

new->loc=address;

new->link=NULL;

if(es[i].head==NULL)//비어있을 경우

{

es[i].head=new;

}

else//안비어있을 때

{

symbolt\* temp;

temp=es[i].head;

while(temp->link!=NULL)

temp=temp->link;

temp->link=new;

}

return;

}

void add\_ref(char\* resw,char\* num,int i)//EST 구조체에 외부 참조 변수들을 저장해준다.

{

ref\* rnew;

rnew=(ref\*)malloc(sizeof(ref));

strcpy(rnew->name,resw);

rnew->ref\_num=to10(num);

rnew->link=NULL;

if(es[i].r\_head==NULL)

es[i].r\_head=rnew;

else

{

ref\* temp;

temp=es[i].r\_head;

while(temp->link!=NULL)

temp=temp->link;

temp->link=rnew;

}

return;

}

void add\_num\_memory(char\* temp,int targetadd,int i,char plus\_minus)//Memory에 modifciation 내용을 수정해주는 함수

{

char original[7];

strcpy(original,m[targetadd]);

strcat(original,m[targetadd+1]);

strcat(original,m[targetadd+2]);

ref\* rtemp;

rtemp=es[i].r\_head;

int add\_resw\_loc;

int k;

if(to10(temp)==1)

{

add\_resw\_loc=es[i].address;

}

else{

char find[7];//외부 참조 심볼

while(rtemp!=NULL)

{

if(rtemp->ref\_num==to10(temp))//number가 같으면

{

strcpy(find,rtemp->name);

break;

}

rtemp=rtemp->link;

}//find 찾음

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

printf("Null 값입니다.\n");

//int add\_prog\_loc;//프로그램의 더할 주소값

for(k=0;k<filenum;k++)

{

symbolt\* stemp;

stemp=es[k].head;

while(stemp!=NULL)

{

if(strcmp(stemp->symbol,find)==0)

{

add\_resw\_loc=stemp->loc;

//add\_prog\_loc=es[k].address;

break;

}

stemp=stemp->link;

}

}

}

char add\_address[7];

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

add\_address[k]='0';

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

int ntemp;

if(plus\_minus=='+')//+일때

ntemp=to10(original)+add\_resw\_loc;//+add\_prog\_loc;//더해줃나

else//-일때

ntemp=to10(original)-add\_resw\_loc;

int templen=strlen(to16(ntemp));

for(k=0;k<templen;k++)

{

add\_address[6-templen+k]=to16(ntemp)[k];//더할 주소 배열 완성

}

//이제 다시 메모리에 로딩

m[targetadd][0]=add\_address[0];

m[targetadd][1]=add\_address[1];

m[targetadd+1][0]=add\_address[2];

m[targetadd+1][1]=add\_address[3];

m[targetadd+2][0]=add\_address[4];

m[targetadd+2][1]=add\_address[5];//loading 완료

}

/\*void add\_sym\_memory(char\* temp,int targetadd,int i)

{

}\*/

void addbp(char bp[7])//breakpoint 추가해주는 함수

{

/\*if(to10(bp)>memoryend)

{

printf("Not allowed ranged breakpoint.\n");

return;

}\*/

bptb\* new;

new=(bptb\*)malloc(sizeof(bptb));

strcpy(new->breakpoint,bp);

new->link=NULL;

if(bphead==NULL)//head가 NULL일경우

{

bphead=new;

}

else

{

bptb\* temp;

bptb\* pre;

pre=NULL;

temp=bphead;

while(temp!=NULL&&to10(new->breakpoint)>to10(temp->breakpoint))

{

pre=temp;

temp=temp->link;

}

if(pre==NULL)

{

new->link=temp;

bphead=new;

}

else{

pre->link=new;

new->link=temp;

}

}

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

}

void printbp(void)//저장된 breakpoint 출력

{

if(bphead==NULL)

{

printf("There is no breakpoint.\n");

return;

}

else

{

printf("\tbreakpoint\n");

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

bptb\* temp;

temp=bphead;

while(temp!=NULL)

{

printf("\t%s\n",temp->breakpoint);

temp=temp->link;

}

}

return;

}

void bpclear(void)

{

if(bphead==NULL)

{

printf("No breakpoint.\n");

}

else

{

bptb\* temp;

//bptb\* temp2;

temp=bphead;

while(temp!=NULL)

{

bptb\* temp2;

temp2=temp;

temp=temp->link;

free(temp2);

}

bphead=NULL;

bptemp=bphead;

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

return;

}

}

void run(void)//run해주는 함수

{

//bptb\* temp;

//temp=bphead;

//int tempend;//breakpoint

if(realstart==1)//처음으로 run이 실행될경우

{

if(bptemp!=NULL)//breakpoint가 있을경우

{

PC=es[0].address;

tempend=to10(bptemp->breakpoint);

bptemp=bptemp->link;

}//처음 시작인데 breakpoint가 없는 설정일경우

else

{

PC=es[0].address;

tempend=memoryend+2;

}

}

else//처음이 아닐 때

{

if(bptemp==NULL)//breakpoint 더이상 존재하지않을때

{

tempend=memoryend+3;

}

else//breakpoint존재할때

{

tempend=to10(bptemp->breakpoint);

bptemp=bptemp->link;

}

}

realstart=0;

int plus=0;//PC값을 더하기 위한 변수 // 지역함수에서 PC값이 안변하면 변하게해준다.

//L=memoryend-1;

if(PC<=tempend){

while(1)

{

beforePC=PC;

plus=0;

//printf("%s%s%s\n\n",m[PC],m[PC+1],m[PC+2]);

if(PC>tempend)

break;

if(PC>memoryend)

{

realend=1;

break;

}

//if(PC>=tempend)//breakpoint를 만날 경우

// break;

char now\_m\_value[3];

strcpy(now\_m\_value,m[PC]);//불러온 메모리 값

/\*if(to10(now\_m\_value)==0)//빈 메모리

{

now\_m++;

continue;//다음 메모리로 넘어간다.

}\*/

//처음으로 나오는 0이 아닌 메모리값은 mnemonic이다.

if((to10(now\_m\_value)/4)\*4==180)//Clear

{

//printf("CLear\n");

CLEAR();

PC++;

}

else if((to10(now\_m\_value)>0&&(to10(now\_m\_value)/4)\*4==0)&&(to10(m[PC+1])/16<16&&to10(m[PC+2])!=0))//수정 필요하다

{

LDA();

//printf("LDA\n");

}

else if((to10(now\_m\_value)/4)\*4==20)//STL

{

STL();

//printf("STL\n");

}

else if((to10(now\_m\_value)/4)\*4==60)//J

{

J();

plus++;

}

else if((to10(now\_m\_value)/4)\*4==48)//JEQ

{

JEQ();

plus++;

//printf("JEQ\n");

}

else if((to10(now\_m\_value)/4)\*4==40)//COMP

{

flag=COMP();

//printf("COMP\n");

}

else if((to10(now\_m\_value)/4)\*4==72)//JSUB

{

JSUB();

plus++;

//printf("JSUB\n");

}

else if((to10(now\_m\_value)/4)\*4==80)//LDCH

{

LDCH();

//printf("LDCH\n");

}

else if((to10(now\_m\_value)/4)\*4==76)//RSUB

{

//RSUB();

PC=L;

plus++;

//printf("RSUB\n");

}

else if((to10(now\_m\_value)/4)\*4==56)//JLT

{

JLT();

plus++;

//printf("JLT\n");

}

else if((to10(now\_m\_value)/4)\*4==12)//STA

{

STA();

//printf("STA\n");

}

else if((to10(now\_m\_value)/4)\*4==84)//STCH

{

STCH();

//printf("STCH\n");

}

else if((to10(now\_m\_value)/4)\*4==16)//STX

{

STX();

//printf("STX\n");

}

else if((to10(now\_m\_value)/4)\*4==104)//LDB

{

LDB();

//printf("LDB\n");

}

else if((to10(now\_m\_value)/4)\*4==116)//LDT

{

LDT();

//printf("LDT\n");

}

else if((to10(now\_m\_value)/4)\*4==160)//COMPR

{

flag=COMPR();

PC+=1;

//printf("COMPR\n");

}

else if((to10(now\_m\_value)/4)\*4==184)//TIXR

{

flag=TIXR();

//printf("RESULT is %d\n",flag);

PC+=1;

//printf("TIXR\n");

}

else if((to10(now\_m\_value)/4)\*4==224)//TD

{

//TD();

flag=-1;

if(to10(m[PC+1])/16==1)//format 4

PC+=3;

else//format3

PC+=2;

//printf("TD\n");

}

else if((to10(now\_m\_value)/4)\*4==220)//WD

{

//WD();

if(to10(m[PC+1])/16==1)

PC+=3;

else

PC+=2;

}

else if((to10(now\_m\_value)/4)\*4==216)//RD

{

//RD();

if(to10(m[PC+1])/16==1)

PC+=3;

else

PC+=2;

}

else

{

PC++;

plus=1;

}

if(plus==0)

PC++;

if(PC>=tempend)//breakpoint 만날경우

{

break;

}

}

}

if(realend==1)

{

printf("\t\tA : %06X X : %06X\n",A,X);

printf("\t\tL : %06X PC: %06X\n",L,PC);

printf("\t\tB : %06X S : %06X\n",B,S);

printf("\t\tT : %06X\n",T);

//프로그램 끝일 경우

printf("\tEnd Program\n");

}

else//breakpoint가 남아있을경우

{

printf("\t\tA : %06X X : %06X\n",A,X);

printf("\t\tL : %06X PC: %06X\n",L,beforePC);

printf("\t\tB : %06X S : %06X\n",B,S);

printf("\t\tT : %06X\n",T);

printf("\tStop at checkpoint[%X]\n",tempend);

}

}

int findxbpe(int\* x,int\* b,int\* p,int\* e)

{

int result;

result=to10(m[PC+1])/16;

if(result==1)

\*e=1;

else if(result==2)

\*p=1;

else if(result==4)

\*b=1;

else if(result==8)

\*x=1;

else if(result==9)

{

\*x=1;

\*e=1;

}

else if(result==10)

{

\*p=1;

\*x=1;

}

else if(result==12)

{

\*b=1;

\*x=1;

}

return result;

}

void CLEAR(void)

{

if(m[PC+1][0]=='0')

A=0;

else if(m[PC+1][0]=='1')

X=0;

else if(m[PC+1][0]=='2')

L=0;

else if(m[PC+1][0]=='3')

B=0;

else if(m[PC+1][0]=='4')

S=0;

else if(m[PC+1][0]=='5')

T=0;

else if(m[PC+1][0]=='8')

PC=0;

return;

}

void LDA(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC or simple addressing

{

A=to10(temp\_t);

}

else if(ni==1)//immediate addressing

{

A=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp\_2[7];

strcpy(temp\_2,m[ta]);

strcat(temp\_2,m[ta+1]);

strcat(temp\_2,m[ta+2]);

A=to10(temp\_2);

}

if(format==4)

PC+=3;

else

PC+=2;

}

void JSUB(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

//int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

L=PC+4;

//format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

L=PC+3;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

L=PC+3;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

L=PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

L=PC+3;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

L=PC+3;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

L=PC+3;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

L=PC+3;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC or simple addressing

{

PC=ta;

}

else if(ni==1)//immediate addressing

{

PC=ta;

}

else if(ni==2)//indirect addressing

{

PC=to10(temp\_t);

}

}

void J(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

//int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

//format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC or simple addressing

{

PC=to10(temp\_t);

}

else if(ni==1)//immediate addressing

{

PC=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

/\*char temp\_2[7];

printf("ta is %d\n",ta);

strcpy(temp\_2,m[ta]);

strcat(temp\_2,m[ta+1]);

strcat(temp\_2,m[ta+2]);

PC=to10(temp\_2);\*/

PC=ta;

}

}

void JLT(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(flag==-1)

{

if(ni==0||ni==3)//SIC or simple addressing

{

PC=ta;

}

else if(ni==1)//immediate addressing

{

PC=ta;

}

else if(ni==2)//indirect addressing

{

PC=to10(temp\_t);

}

}

else

{

if(format==4)

PC+=4;

else

PC+=3;

}

}

void JEQ(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(flag==0)

{

if(ni==0||ni==3)//SIC or simple addressing

{

PC=ta;

}

else if(ni==1)//immediate addressing

{

PC=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

PC=ta;

}

}

else

{

if(format==4)

PC+=4;

else

PC+=3;

}

}

int COMP(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

int data;

if(ni==0||ni==3)//SIC or simple addressing

{

data=to10(temp\_t);

}

else if(ni==1)//immediate addressing

{

data=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp\_2[7];

strcpy(temp\_2,m[ta]);

strcat(temp\_2,m[ta+1]);

strcat(temp\_2,m[ta+2]);

data=to10(temp\_2);

}

if(format==4)

PC+=3;

else

PC+=2;

if(A==data)

return 0;

else if(A<data)

return -1;

else

return 1;

}

int data\_reg(int num)

{

if(num==0)

return A;

else if(num==1)

return X;

else if(num==2)

return L;

else if(num==8)

return PC;

else if(num==3)

return B;

else if(num==4)

return S;

else

return T;

}

int COMPR(void)

{

int reg\_f1;

int reg\_f2;

reg\_f1=(to10(m[PC+1]))/16;

reg\_f2=(to10(m[PC+1]))%16;

if(data\_reg(reg\_f1)==data\_reg(reg\_f2))

return 0;

else if(data\_reg(reg\_f1)>data\_reg(reg\_f2))

return 1;

else

return -1;

}

int TIXR(void)

{

X++;

int reg\_f1;

//int reg\_f2;

reg\_f1=(to10(m[PC+1]))/16;

//reg\_f2=(to10(m[PC+1]))%16;

if(reg\_f1==0)//compare with A

{

if(X==A)

return 0;

else if(X<A)

return -1;

else

return 1;

}

else if(reg\_f1==1)//compare with X

{

if(X==X)

return 0;

else if(X<X)

return -1;

else

return 1;

}

else if(reg\_f1==2)//compare with L

{

if(X==L)

return 0;

else if(X<L)

return -1;

else

return 1;

}

else if(reg\_f1==8)//compare with PC

{

if(X==PC)

return 0;

else if(X<PC)

return -1;

else

return 1;

}

else if(reg\_f1==3)//compare with B

{

if(X==B)

return 0;

else if(X<B)

return -1;

else

return 1;

}

else if(reg\_f1==4)//compare with S

{

if(X==S)

return 0;

else if(X<S)

return -1;

else

return 1;

}

else if(reg\_f1==5)//Compare with T

{

if(X==T)

return 0;

else if(X<T)

return -1;

else

return 1;

}

return 10;

}

void STCH(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC or simple addressing

{

char temp[3];

int t=A%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

}

else if(ni==1)//immediate addressing

{

//A=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp[3];

int t=A%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

}

if(format==4)

PC+=3;

else

PC+=2;

}

void LDCH(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC or simple addressing

{

A=(A/(16\*16))\*16\*16+to10(m[ta]);

}

else if(ni==1)//immediate addressing

{

A=(A/(16\*16))\*16\*16+ta/(16\*16\*16\*16);

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

A=(A/(16\*16))\*16\*16+to10(m[ta]);

}

if(format==4)

PC+=3;

else

PC+=2;

}

void LDT(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==3)//simple addressing

{

T=to10(temp\_t);

}

else if(ni==1)//immediate addressing

{

T=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp\_2[7];

strcpy(temp\_2,m[ta]);

strcat(temp\_2,m[ta+1]);

strcat(temp\_2,m[ta+2]);

T=to10(temp\_2);

}

if(format==4)

PC+=3;

else

PC+=2;

}

void LDB(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==3)//simple addressing

{

B=to10(temp\_t);

}

else if(ni==1)//immediate addressing

{

B=ta;

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp\_2[7];

strcpy(temp\_2,m[ta]);

strcat(temp\_2,m[ta+1]);

strcat(temp\_2,m[ta+2]);

B=to10(temp\_2);

}

if(format==4)

PC+=3;

else

PC+=2;

}

void STX(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC and simple addressing

{

char temp[3];

int t=X/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(X%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=X%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp[3];

int t=X/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(X%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=X%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

if(format==4)

PC+=3;

else

PC+=2;

}

void STA(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC and simple addressing

{

char temp[3];

int t=A/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(A%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=A%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp[3];

int t=A/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(A%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=A%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

if(format==4)

PC+=3;

else

PC+=2;

}

void STL(void)

{

int ni=to10(m[PC])-(to10(m[PC])/4)\*4;//n,i의 값이다.ex) n=1,i=1면 3

int format=3;

int x,b,p,e;

x=0;

b=0;

p=0;

e=0;

int ta;//target address이다.

int disp;

char temp3[7];

char temp4[7];

temp4[0]=m[PC+1][1];

temp4[1]='\0';

strcat(temp4,m[PC+2]);

strcat(temp4,m[PC+3]);//4형식일때 disp

temp3[0]=m[PC+1][1];

temp3[1]='\0';

strcat(temp3,m[PC+2]);//3형식일때 disp

int k;

k=findxbpe(&x,&b,&p,&e);

if(e==1)//FORMAT4

{

ta=to10(temp4);

format=4;

}

else if(ni==0)//SIC mnemonic

{

ta=to10(temp3);

if(x==1)

ta+=X;

}

else if(k==0)//FORMAT3 and not pc,b

{

disp=to10(temp3);

ta=disp;

}

else if(k==2)//PC relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3;

}

else if(k==4)//B relative

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B;

}

else if(k==8)//X indexed

{

disp=to10(temp3);

ta=disp+X;

}

else if(k==10)//p, x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+PC+3+X;

}

else if(k==12)//b,x indexed

{

disp=to10(temp3);

if(disp>=0x800)

disp=disp-0xfff-0x1;

ta=disp+B+X;

}

char temp\_t[7];

strcpy(temp\_t,m[ta]);

strcat(temp\_t,m[ta+1]);

strcat(temp\_t,m[ta+2]);

if(ni==0||ni==3)//SIC and simple addressing

{

char temp[3];

int t=L/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(L%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=L%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

else if(ni==2)//indirect addressing

{

ta=to10(temp\_t);

char temp[3];

int t=L/(16\*16\*16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta],temp);

}

t=(L%(16\*16\*16\*16))/(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+1],temp);

}

t=L%(16\*16);

if(strlen(to16(t))==1)

{

temp[0]='0';

temp[1]=to16(t)[0];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

else

{

temp[0]=to16(t)[0];

temp[1]=to16(t)[1];

temp[2]='\0';

strcpy(m[ta+2],temp);

}

}

if(format==4)

PC+=3;

else

PC+=2;

}