과목 명: 시스템프로그래밍

담당 교수 명: 김 지 환

<<Project 1>>

**서강대학교 컴퓨터공학과**

**[20141556]**

**[이성문]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 3

3.1 모듈 이름 : main() 3

3.1.1 기능 3

3.1.2 사용 변수 3

3.2 모듈 이름: display\_instructions() 3

3.2.1 기능 3

3.2.2 사용 변수 3

3.3 모듈이름: get\_data() 4

3.3.1 기능 4

3.3.2 사용변수 4

3.4 모듈이름: display\_report() 4

3.4.1 기능 4

3.4.2 사용변수 4

3.5 모듈이름: continue\_function() 4

3.5.1 기능 4

3.5.2 사용변수 4

4. 전역 변수 정의 4

4.1 long income[100] 4

4.2 int month[100], day[100], year[100] 4

4.3 int ctr 4

4.4 int cont 5

# 프로그램 개요

이 프로그램은 SIC/XE 머신을 구현하기 위한 전 단계로서 어셈블러, 링크, 로더들을 실행하게 될 셸과 컴파일을 통해서 만들어진 object 코드가 적재되고 실행될 메모리 공간과 mnemonic 을 opcode 값으로 변환하는 opcode 테이블과 관련 명령어들을 구현하는 프로그램이다.

# 프로그램 설명

## 프로그램 흐름도

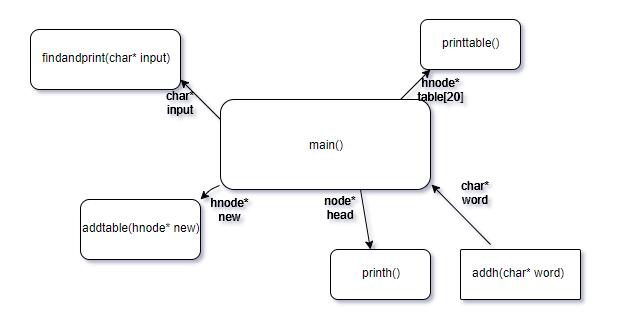


그림 1> 프로그램 흐름도.

# 모듈 정의

## 모듈 이름 : main()

### 기능

Input String을 입력받아 옳은 문법일 경우, “help”,”dir”,”quit”,”history”, “dump”, “edit”,”fill”,”reset”,”opcode” 에 해당될 경우 각 기능을 main에서 거의 모두 수행한다. 예외와 오류를 모두 main에서 처리했으며 모든 기능 또한 main에서 기능된다.

### 사용 변수

Char\* input2 – 명령어 입력시 받는 전체 스트링

Char\* input – input2 스트링에서 제일 앞에 있는 단어

Char\* inputn – input2에서 빈칸과 탭을 지워서 줄인 스트링 / 오류검사할 때 사용한다

Char enter;//junk 문자

Char d[4] – 파일 입력때 필요

Char\* strat,end – dump와 fill 실행시 주소의 처음과 끝

DIR\* d; - dirent 헤더 파일에서 파일을 불러오기 위한 변수

Struct dirent\* f – 파일 정보가 담겨있는 구조체

Struct stat info – 파일 모드가 담겨있는 구조체

Int error1, error2, error3 – 입력 스트링들의 에러를 확인해주는 변수 0일시 오류가 없고 1일시 오류 있음을 표현

Char comma – ‘,’를 저장하는 변수

Int lineone,linetwo – dump시 주소 출력을 시작하는 줄과 끝내는 줄

Int two – 출력의 마지막 주소값 (인덱스)

Char\* add – edit 실행시 바꿀 주소

Char\* value – edit 실행시 바꿔질 메모리 내용값

Char\* m[8] – opcode 실행시 입력받을 명령어

## 모듈 이름: addh(char\* word)

### 기능

올바른 명령어가 들어왔을 경우 word(명령어)를 head 노드에 linked list로 구현하여 연결시켜준다..

### 사용 변수

Char\* word - 입력 문자열

Node\* head – history를 저장하는 head 노드이다.

## 모듈이름: printh()

### 기능

Node\* head에 연결되있는 모든 노드들을 화면에 출력시켜준다. 즉, 이제껏 수행된 명령어들을 출력시켜준다.

### 사용변수

Node\* head – history를 저장하는 head 노드이다.

Node\* temp – 출력할때 필요한 임시 노드 변수이다.

## 모듈이름: addtable(hnode\* new)

### 기능

Opcode.txt 파일에서 파일 입력으로 받은 문자열들, 즉 opcode와 명령어들을 메인에서 구현한 Hash table에 Key 값을 참조해 linked list로 연결해주는 함수이다. 여기서 Key는 각 명령어의 첫번째 알파벳으로 하여, 같을 경우 linked list로 연결해주었다.

### 사용변수

Hnode\* new – 여기서 hnode는 해쉬 테이블 저장을 위한 노드이고 new는 새로운 입력받은 해쉬 노드이다.

Hnode\* table[20] – 사이즈가 20인Hash table이다.

## 모듈이름: findandprint(char\* input)

### 기능

사용자에게 입력받은 input 문자열을 Hash table에서 찾아서 있으면 출력해준다. 있을경우 1을 리턴하고 없을 경우 0을 리턴한다.

### 사용변수

Hnode\* temp – Input 을 찾을 때 사용되는 임시 노드 변수

Hnode\* table[20] – Hash table 데이터

## 모듈이름: printable()

### 기능

Hash table에 저장된 데이터들을 차례대로 Key값에 따라 정해진 형식에 맞게 출력해준다.

### 사용변수

Hnode\* temp – Hash table에 있는 데이터들을 출력할 때 필요한 임시 노드 변수

Hnode\* table[20] – Hash table 데이터

# 전역 변수 정의

## Node\* head

Typedef struct \_NODE{

Char\* word;

Struct \_NODE\* link;

}node;

History에 저장될 명령어들의 연결 리스트이다.

## Hnode\* table[20]

Typedef struct \_HNODE{

Char op[3];

Char ins[7];

Struct \_HNODE\* link;

}hnode;

Hash table에 저장될 opcode 데이터들의 구조체이다.

## Int savenum

Dump시 마지막에 출력되 다음 dump를 수행할 시 시작될 주소 값을 저장한 변수이다.

.

## Char m[1048576][3]

가상 메모리 배열이다. 총 크기는 16의 5제곱이고 메모리 내용값을 저장하기 위해 이중 배열로 [3]을 정적 할당하였다.

# 코드

/\* 포함되는 헤더 파일들\*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<dirent.h>

#include<sys/stat.h>

typedef struct \_NODE{

char\* word;

struct \_NODE\* link;

}node;

node\* head;

typedef struct \_HNODE{

char op[3];

char ins[7];

struct \_HNODE\* link;

}hnode;

hnode\* table[20];

int savenum;//저장되는 마지막 address

void addh(char\* word);//histroy에 node를 저장해주는함수

void printh();//history에 저장된 노드들을 출력해주는 함수

void deletem();//메모리 해제

int checkin(char\* input);//16진수인지 확인해주는 함수

int checkin2(char\* input);//입력받은 start, end가 문법상 오류가 없는지 확인해주는 함수

char\* rmspace(char\* input);//빈칸이나 탭을 지워주는 함수

char\* changestart(char\* input);//start 뒤 부분에 공백이 있을경우 삭제해주는 함수

char\* rmfordump(char\* input);//dump 내에서 새로 문법을 고쳐주는 함수

int powe(int one ,int two);//one의 two승을 계산해주는 함수

int to10(char\* input);//16진수를 10진수로 바꿔줌

char\* to16(int input);//10진수를 16진수로 바꿔줌

char m[1048576][3];//메모리크기이다.

void addtable(hnode\* new);//새로운 테이블 노드를 테이블안에 넣어주는 함수

int findandprint(char\* input);//명령어를 입력하면 해당하는 opcode를 출력해주는 함수

void printtable(void);//테이블 데이터들을 모두 출력해주는 함수.

//여기까지가 20141556.h 헤더파일 내용

#include"20141556.h"

int main()

{

int i,j;

savenum=0;

for(i=0;i<=1048575;i++)//메모리 내용 초기화

{

m[i][0]='0';

m[i][1]='0';

m[i][2]='\0';

}

head=NULL;//linked list 구현을 위한 head 포인터 초기화

char\* input2;//입력되는 명령어

char\* inputn;

char\* input;

char enter;

char d[4];

//여기서부터 hash table에 opcode 파일입력

FILE\* p=fopen("opcode.txt","rt");//해쉬 테이블을 위한 파일 입력

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

{

table[i]=(hnode\*)malloc(sizeof(hnode));//메모리 할당

strcpy(table[i]->op,"--");

}

for(i=0;;i++)

{

hnode\* new;//새로운 노드

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

new->link=NULL;

if(fscanf(p,"%s %s %s%c",new->op,new->ins,d,&enter)==EOF)

break;

addtable(new);//해쉬 테이블에 추가

}

fclose(p);//종료

while(1){//반복한다

printf("sicsim> ");//명령어를 입력받는다.

input2=(char\*)malloc(sizeof(char)\*256);

input=(char\*)malloc(sizeof(char)\*256);

for(i=0;;i++)//history에 저장을 위해 공백도 다 입력받는다.

{

scanf("%c",input2+i);

if(input2[i]=='\n')

break;

}

input2[i]='\0';

inputn=rmspace(input2);//공백을 지워준다.

char\* start;//dump에서 첫번째 주소를 위한 변수

char\* end;//마찬가지로 두번째 주소를 위한 변수.

start=(char\*)malloc(sizeof(char)\*20);//메모리 할당

end=(char\*)malloc(sizeof(char)\*20);

start[0]='\*';//이 작업은 start만 받았는지 아니면 둘다 받았는지 아니면 아무것도 안받았는지 확인하기 위한 것임

end[0]='\*';

sscanf(input2,"%s",input);//앞에 한단어만 추려낸다.

//sscanf(input2,"%s %2[^,] %\*c %s",input,start,end);

if(strcmp(input,"help")==0||strcmp(input,"h")==0)//help 명령어 수행

{

if(strcmp(input,"help")==0)//예외 작업

{

if(inputn[4]!='\0')

{

printf("Error\n");

continue;

}

}

else if(strcmp(input,"h")==0)//예외 작업

if(inputn[1]!='\0')

{

printf("Error\n");

continue;

}

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");

addh(input);//linked list에 추가

}

else if(strcmp(input,"dir")==0||strcmp(input,"d")==0)//dir 명령어 수행

{

if(strcmp(input,"dir")==0){//예외 작업

if(inputn[3]!='\0')

{

printf("Error\n");

continue;

}

}

else if(strcmp(input,"d")==0)//예외 작업

{

if(inputn[1]!='\0')

{

printf("Error\n");

continue;

}

}

DIR\* d;

struct dirent \*f;//파일 정보를 얻기위한 변수

struct stat info;

d=opendir(".");//현재 위치 디렉토리를 열어준다.

if(!d)continue;//오류일경우 명령어를 다시 받는다.

while(1)

{

f=readdir(d);//현재 디렉토리 환경을 f 변수에 저장

if(!f)break;

stat(f->d\_name,&info);//파일 정보를 입력받는다.

if(info.st\_mode==16877)

printf("%s\t/",f->d\_name);//디렉토리 파일일 경우 /출력

else if((S\_IEXEC & info.st\_mode)==S\_IEXEC)//실행 파일일 경우 \*출력

printf("%s\*\t",f->d\_name);

else

printf("%s\t",f->d\_name);//그 무엇도 아닐때 그냥 출력

}

closedir(d);//함수종류

printf("\n");

addh(input2);//linked list에 명령어 추가

}

else if(strcmp(input,"quit")==0||strcmp(input,"q")==0)//quit 명령어 수행

{

if(strcmp(input,"quit")==0)//예외 작업

{

if(inputn[4]!='\0')

{

printf("Error\n");

continue;

}

}

else if(strcmp(input,"q")==0)//예외작업

{

if(inputn[1]!='\0')

{

printf("Error\n");

continue;

}

}

break;

}

else if(strcmp(input,"history")==0||strcmp(input,"hi")==0)//history 명령어 수행

{

if(strcmp(input,"history")==0){//예외 작업

if(inputn[7]!='\0')

{

printf("Error\n");

continue;

}

}

else if(strcmp(input,"hi")==0){//예외 작업

if(inputn[2]!='\0')

{

printf("Error\n");

continue;

}

}

addh(input2);//history 또한 명령어므로 사용 내역에 추가

printh();//history에 저장된 노드들 출력

}

else if(strcmp(input,"dump")==0||strcmp(input,"du")==0)//dump 명령어를 입력받을 경우

{

int error1=0;

int error2=0;

char comma='\0';

char junk[2]={'\0','\0'};

sscanf(input2,"%s %[^,] %c %s %s",input,start,&comma,end,junk);

if(comma==',')//콤마가 있을경우 start end 둘다 있는지 확인

{

if(junk[0]!='\0'){//쭈우욱 예외 작업들이다.

printf("Error\n");

continue;

}

start=changestart(start);//오류 검사를 위한 변형

error1=checkin2(start);//첫번째 오류 검사

error2=checkin2(end);

if(error1+error2!=0)

{

printf("Error\n");

continue;

}

error1=0;

error2=0;

error1=checkin(start);//두번째 오류검사

error2=checkin(end);

if(error1+error2!=0)

{

printf("Error\n");

continue;

}

if(to10(start)>to10(end))

{

printf("Error\n");

continue;

}

int one,two;

int lineone,linetwo;//첫 줄과 마지막 줄

one=to10(start);

two=to10(end);

if(two>=1048576)//메모리 경계를 넘어갈 경우

two=1048575;

lineone=(one/16);//첫번째로 출력하는 공간

linetwo=(two/16);//마지막으로 출력하는 공간

for(i=lineone;i<=linetwo;i++)//줄수만큼 돌려준다.

{

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

for(j=i\*16;j<=i\*16+15;j++)

{

if(j>=one&&j<=two)//출력 범위일 경우

printf("%02X ",to10(m[j]));//16진수로 출력해준다.

else

printf(" ");

}

printf("; ");

for(j=i\*16;j<=i\*16+15;j++)//메모리 내용에 해당하는 문자를 출력한다.

{

if(j>=one&&j<=two)

{

if(to10(m[j])>=32&&to10(m[j])<=126)//출력 범위일때

printf("%c",to10(m[j]));

else

printf(".");

}

else

printf(".");

}

printf("\n");

}

if(two==1048575)//Boundary를 넘어갈 경우이다.

savenum=0;

else

savenum=to10(end)+1;

}

else if(start[0]=='\*')// dump만 입력받을 경우

{

if(inputn[4]==','||inputn[2]==',')//예외 작업

{

printf("Error\n");

continue;

}//savenum이 출발 지점

int lineone,linetwo;

lineone=savenum/16;

int two=savenum+159;

if(two>=1048576)//Boundary check

two=1048575;

linetwo=(two)/16;

for(i=lineone;i<=linetwo;i++)

{

printf("%05X ",i\*16);//16진수 출력

for(j=i\*16;j<=i\*16+15;j++)

{

if(j>=savenum&&j<=two)//범위 내일 경우 출력

printf("%02X ",to10(m[j]));

else

printf(" ");

}

printf("; ");

for(j=i\*16;j<=i\*16+15;j++)

{

if(j>=savenum&&j<=two)

{

if(to10(m[j])>=32&&to10(m[j])<=126)//마찬가지로 범위 내일 경우 출력

printf("%c",to10(m[j]));

else

printf(".");

}

else

printf(".");

}

printf("\n");

}

if(two==1048575)//Boundary를 넘길 경우 작업

savenum=0;

else

savenum=savenum+160;

}

else if(start[0]!='\*' && end[0]=='\*')//start 하나만 입력받을 경우

{

error1=0;

error1=checkin(start);

if(error1==1)

{

printf("Error\n");

continue;

}

int lineone,linetwo;savenum=to10(start);//위 내용들과 같은 작업들이다.

int two=to10(start)+159;

if(two>=1048576)

two=1048575;

lineone=to10(start)/16;

linetwo=(two)/16;

for(i=lineone;i<=linetwo;i++)

{

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

for(j=i\*16;j<=i\*16+15;j++)

{

if(j>=savenum&&j<=two)

printf("%02X ",to10(m[j]));

else

printf(" ");

}

printf("; ");

for(j=i\*16;j<=i\*16+15;j++)

{

if(j>=savenum&&j<=two)

{

if(to10(m[j])>=32&&to10(m[j])<=126)

printf("%c",to10(m[j]));

else

printf(".");

}

else

printf(".");

}

printf("\n");

}

if(two==1048575)

savenum=0;

else

savenum=savenum+160;

}

addh(input2);//history에 추가해준다.

}

else if(strcmp(input,"edit")==0||strcmp(input,"e")==0)//메모리 내용을 원하는 16진수로 바꿔줄때

{

int error1=0;

int error2=0;

char\* add=(char\*)malloc(sizeof(char)\*6);//바꿀 주소이다.

char\* value=(char\*)malloc(sizeof(char)\*3);//바꿀 메모리 내용이다.

char comma='\0';

char junk[2]={'\0','\0'};

sscanf(input2,"%s %[^,] %c %s %s",input,add,&comma,value,junk);//dump와 마찬가지 방법으로 인자들을 입력받는다.

if(comma==',')

{

if(junk[0]!='\0')//예외 처리이다.

{

printf("Error\n");

continue;

}

add=changestart(add);

error1=checkin2(add);

error2=checkin2(value);

if(error1+error2!=0)//오류 처리

{

printf("Error\n");

continue;

}

error1=0;error2=0;

error1=checkin(add);

error2=checkin(value);

if(error1+error2!=0)

{

printf("Error\n");

continue;

}

strcpy(m[to10(add)],value);//원하는 내용으로 바꿔준다.

}

else

{

printf("Error\n");

continue;

}

addh(input2);//history에 추가해준다.

}

else if(strcmp(input,"fill")==0||strcmp(input,"f")==0)//start번지부터 end 번지까지의 값을 value에 지정된 값으로 변경해주는 함수

{

int error1=0;

int error2=0;

int error3=0;

char a,b;a='\0';b='\0';

char junk[2]={'\0','\0'};

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

sscanf(input2,"%s %[^,] %c %[^,] %c %s %s",input,start,&a,end,&b,value,junk);

if(a==','&&b==',')//제대로 입력되었을 때

{

if(junk[0]!='\0')//오류 검사

{

printf("Error\n");

continue;

}

start=changestart(start);

end=changestart(end);

error1=checkin2(start);

error2=checkin2(end);

error3=checkin2(value);

if(error1+error2+error3!=0)//오류 검사

{

printf("Error\n");

continue;

}

error1=0;error2=0;error3=0;

error1=checkin(start);

error2=checkin(end);

error3=checkin(value);

if(error1+error2+error3!=0)//계속 되는오류 검사

{

printf("Error\n");

continue;

}

if(to10(start)>to10(end))//예외 처리

{

printf("Error\n");

continue;

}

if(strlen(value)>=3)

{

printf("Error\n");

continue;

}

//여기까지가 문법이나 이상한 오류 검사하는 과정

for(i=to10(start);i<=to10(end);i++)//바꿔준다.

{

strcpy(m[i],value);

}

}

else

{

printf("Error\n");

continue;

}

addh(input2);//history에 추가해준다.

}

else if(strcmp(input,"reset")==0)//모든 메모리를 0으로 만들고 싶을 때

{

if(inputn[5]!='\0')//오류 검사

{

printf("Error\n");

continue;

}

for(i=0;i<=1048575;i++)//모든 메모리를 바꿔준다.

{

strcpy(m[i],"00");

}

addh(input2);//history에 추가

}

else if(strcmp(input,"opcode")==0)//opcode 명령어일 때

{

char m[8];

char junk[2]={'\0','\0'};

sscanf(input2,"%s %s %s",input,m,junk);

if(junk[0]!='\0')//오류 검사

{

printf("Error\n");

continue;

}

if(findandprint(m)==0)//만약 출력할 명령어가 없을 경우이다. Invlalid

continue;

addh(input2);

}

else if(strcmp(input,"opcodelist")==0)//Hash table에 저장된 opcode들을 순서대로 출력해준다.

{

if(strcmp(inputn,"opcodelist")!=0)//오류 처리

{

printf("Error\n");

continue;

}

printtable();//출력해준다.

addh(input2);//history에 넣어준다.

}

else//올바르지않은 입력일 경우 에러

{

printf("Error\n");

}

}

deletem();//메모리 해제

return 0;

}

void addh(char\* word)//linked list에 추가하는 함수

{

node\* new;

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

new->link=NULL;

new->word=word;

if(head==NULL)

{

head=new;

return;

}

else

{

node\* temp;

temp=head;

while(temp->link!=NULL)

temp=temp->link;

temp->link=new;

}

}

void printh()//linked list에 저장된 데이터들을 출력해주는 함수.

{

node\* temp;

temp=head;

int num=1;

while(temp!=NULL)

{

printf("\t%-5d%s\n",num,temp->word);

temp=temp->link;

num++;

}

}

void deletem()// linked list에 할당된 메모리를 해제하는 함수이다.

{

node\* temp;

node\* temp2;

//할당이 안되있을 경우 함수 종료.

if(head!=NULL)

{

temp=head;

while(temp!=NULL)

{

temp2=temp;

free(temp2->word);

free(temp2);

temp=temp->link;

}

}

int i;

hnode\* temp3;

hnode\* temp4;

for(i=0;i<20;i++)//Hash table에 있는 메모리를 해제시켜준다.

{

if(strcmp(table[i]->op,"--")!=0)//링크드 리스트가 없을 경우이다.

{

temp3=table[i];

while(temp3!=NULL)

{

temp4=temp3;

free(temp4);

temp3=temp3->link;

}

}

else

{

temp3=table[i];

free(temp3);

}

}

}

int checkin(char\* input)//올바른 16진수가 들어왔는지 확인해주는 함수이다.

{

int i;

for(i=0;;i++)

{

if(input[i]=='\0')

break;

if((input[i]>='0'&&input[i]<='9')||(input[i]>='A'&&input[i]<='F')||(input[i]>='a'&&input[i]<='f'))//올바른 범위 안에 있을때

continue;

else

return 1;

}

return 0;

}

int checkin2(char\* input)//입력받은 start, end가 중간에 공백이 있거나 길이가 5자리를 넘어가는지 확인해준다.

{

int i;

int length=0;

for(i=0;;i++)

{

if(input[i]=='\0')

return 0;

if(length>=6)

return 1;

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

return 1;

length++;

}

}

char\* rmspace(char\* input)//빈칸을 지워서 앞으로 땡겨주는 함수이다.

{

int i;

int j=0;

char\* result;

result=(char\*)malloc(sizeof(char)\*strlen(input));

for(i=0;;i++)

{

if(input[i]=='\0')

break;

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

continue;

result[j]=input[i];

j++;

}

result[j]='\0';

return result;

}

char\* changestart(char\* input)//start 변수를 바꿔주는 함수이다.

{

char\* result;

int end=strlen(input)-1;

result=(char\*)malloc(sizeof(char)\*(end+1));

int i;

int nend;

for(i=end;;i--)//start의 뒤에 공백들을 없애주는 역할을 한다.

{

if(i==0)

{

nend=i;

break;

}

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

continue;

else{

nend=i;

break;

}

}

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

result[i]=input[i];

i++;

result[i]='\0';

return result;

}

int to10(char\* input)//10진수로 바꿔주는 함수이다.

{

int length=strlen(input);

int i;

int result=0;

for(i=length-1;i>=0;i--)

{

if(input[i]>='A'&&input[i]<='F')

result+=(input[i]-55)\*powe(16,length-i-1);

else if(input[i]>='a'&&input[i]<='f')

result+=(input[i]-87)\*powe(16,length-i-1);

else

result+=(input[i]-48)\*powe(16,length-i-1);

}

return result;

}

char\* to16(int input)//16진수 문자열로 바꿔주는 함수이다.

{

int\* num;

num=(int\*)malloc(sizeof(int)\*6);

char\* result;

result=(char\*)malloc(sizeof(char)\*6);

int i;

int temp=input;

for(i=0;i<5;i++)

{

if(temp<16)

{

num[i]=temp;

break;

}

num[i]=temp%16;

temp=temp/16;

}

int check=i;

num[i+1]=-1;

int j=0;

for(i=check;i>=0;i--)

{

if(num[i]>=0&&num[i]<=9)

result[j]=num[i]+'0';

else

result[j]=num[i]-10+'A';

j++;

}

result[j]='\0';

return result;

}

int powe(int one,int two)//one의 two승을 구해준다.

{

int result=one;

int i;

if(two==0)

return 1;

else if(two==1)

return one;

for(i=0;i<two-1;i++)

result\*=one;

return result;

}

void addtable(hnode\* new)//Hash table에 새로운 노드를 추가해준다.

{

int i;

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

{

if((table[i]->ins)[0]==(new->ins)[0])//같은 Key일경우이다.

{

hnode\* temp;

temp=table[i];

while(temp->link!=NULL)

temp=temp->link;

temp->link=new;

break;

}

else//키가 다르고

{

if(strcmp(table[i]->op,"--")==0)//채워져있지않은 table이면

{

table[i]=new;

break;

}

else//이미 채워져있으면

continue;

}

}

}

int findandprint(char\* input)//table 내에 input이 있는지 찾아주고 있으면 출력해주고 1을 리턴, 없으면 0을 리턴

{

int i;

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

{

if(input[0]==(table[i]->ins)[0])//Key 값이 같을 경웅이다.

{

hnode\* temp;

temp=table[i];

while(temp!=NULL&&(strcmp(temp->ins,input)!=0))

temp=temp->link;

if(temp==NULL)

{

printf("Invalid mnemonic\n");

return 0;

}

printf("opcode is %s\n",temp->op);

return 1;

}

}

printf("Invalid mnemonic\n");

return 0;

}

void printtable()//Hash table을 다 출력해준다.

{

int i=0;

hnode\* temp;

temp=table[0];

while(i<20)

{

printf("%d : ",i);

if(strcmp(table[i]->op,"--")==0)

{

printf("empty\n");

i++;

continue;

}

temp=table[i];

while(temp!=NULL)

{

printf("[%s,%s] ",temp->op,temp->ins);

temp=temp->link;

if(temp==NULL)

break;

else

{

printf("-> ");

}

}

printf("\n");

i++;

}

return;

}