**과목명: 시스템프로그래밍**

**1 분반**

**<<Project #2>>**

**서강대학교 유럽문화학과**

**20170301**

**황선애**

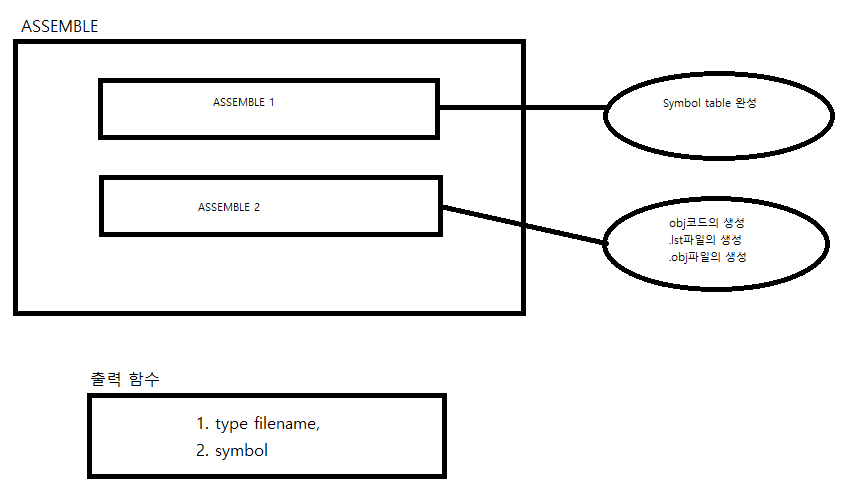
목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. 프로그램 흐름도
3. **모듈 정의**
   1. 정의한 모듈에 대해 간략적인 설명
4. **전역 변수 정의**
5. **코드 설명**
6. **프로그램 개요**

프로젝트1에서 구현한 셀(shell)에 assemble 기능을 추가하는 프로그램. SIC/XE의assembly program source 파일을 입력받아서 object파일을 생성하고, 어셈블리과정 중 생성된 symbol table과 결과물인 object 파일을 볼 수 있는 기능을 제공해야 함. 교재의2.2까지 설명된 SIC/XE 어셈블러의 기능을 구현함을 원칙으로 한다.

1. **프로그램 설명**

**2.1 프로그램 흐름도**



1. **모듈 정의**

int typeFilename(char \*filename); // 입력한 filename의 내용을 보여준다.

int assemble(char \*asmfile, int pass); // pass1, pass2 알고리즘을 이용해 loc와 obj코드를 생성하고, 작성한다.

int insert\_symbol(int hashBal, symbol\* newNode); // 심볼테이블에 심볼을 삽입하는 함수이다.

int findSymbol(char\* symbol); // 심볼 테이블에서 해당 해시값이 주인이 있는지

int h\_s(char\* mnemonic); // 심볼테이블 작성을 위한 해시함수로서, 심볼에서 아스키문자'A'를 뺀다.

int showSt(void); // 심볼테이블을 보여준다.

int makeObFile(char \*filename); // 오브젝트 파일을 만든다.

**4 전역 변수 정의**

char\* fm=NULL; // findOpcode\_2 함수에서 포맷을 저장한다

int ob; // findOpcode\_2 함수에서 인자로 들어온 opcode의 16진수 값을 저장한다.

int start; // Assembly 파일의 시작 주소

int end; // Assembly 파일의 끝 주소

int error\_line; // 에러가 났던 줄 넘버를 저장

symbol\* ht\_s[26]; // 알파벳 자모의 개수 26개, 링크드리스트로 구현한 해시테이블 (SYMTAB)

int LOCCTR; // 심볼의 Loc값

int base; // 베이스 레지스터

int loc; // opr1의 LOC값

**5 코드 설명**

**20170301.c – Proj2에서 작성한 함수와 명령어**

1. **Main 함수**

else if (!strcmp(processed[0],"assemble")&&processed[1]&&!processed[2]){

int i=0;

// 심볼테이블의 초기화. 가장 최신에 어셈블한 것으로 테이블을 채우기 위해

symbolPtr worker,next;

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

ht\_s[i]=NULL;

}

check=assemble(processed[1],1);

int check2=-1;

if (!check) {

check2=assemble(processed[1],2); // pass1 실행

if(!check2){ // pass1이 정상적으로 실행되면,

makeObFile(processed[1]); // pass2 실행

insert(original);

}

}

}

else if (!strcmp(processed[0],"symbol")&&!processed[1]){

check=showSt(); // SYMTAB 출력

if (!check) insert(original);

}

else if (!strcmp(processed[0],"type")&&processed[1]&&!processed[2]){

check=typeFilename(processed[1]);

if(!check) insert(original);

}

1. **그 외 함수**

int h\_s(char\* mnemonic){ // 심볼 테이블 작성을 위해 필요한 해시 함수

int retVal=0;

retVal=mnemonic[0]-'A';

return retVal;

}

int findOpcode\_2(char\* mnemonic){ // fincdOpcode와 기능은 똑같지만 전역 변수 2개에 값을 저장하고, 출력은 하지 않음

int flag=0;

bucketPtr worker=ht[h(mnemonic)];

while(worker){

if (!strcmp(worker->key, mnemonic)){

fm=malloc(sizeof(char)\*4);

fm[3]='\0';

strcpy(fm,worker->format);

ob=worker->opcode;

flag=1;

break;

}

worker=worker->link;

}

if (flag==1) return 0;

else return -1;

}

int showHt(void){

bucketPtr worker;

int i;

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

printf("%d : ",i);

if (!ht[i]){ printf("\n"); continue;}

for(worker=ht[i];worker->link!=NULL;worker=worker->link){

printf("[%s,%02X] -> ",worker->key,worker->opcode);

}

printf("[%s,%02X]",worker->key,worker->opcode);

printf("\n");

}

return 0;

}

int typeFilename(char\* filename){

// 인자로 들어온 lst 혹은 obj 파일을 line by line으로 출력

char buffer[100];

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

if (!fp) {

printf("error : Type correct file name.\n");

return -1;

}

while (fgets(buffer,100,fp)!=NULL) printf("%s",buffer);

return 0;

}

int findSymbol(char\* symbol){

// 심볼테이블에서 라벨이 있는지 검사, 이미 있으면 set error flag

// 전역변수 base, loc는 각각 베이스의 LOC값과 OPR1의 LOC값

int flag=0;

symbolPtr worker=ht\_s[h\_s(symbol)];

while (worker){

if (!strcmp(worker->opcode,symbol)){

if(!strcmp(worker->opcode,"LENGTH")){

base=worker->loc;

}

loc=worker->loc;

flag=1;

break;

}

worker=worker->link;

}

if (flag==1) return 0; // SYMTAB에 있다면 0을 리턴

else return -1;

}

int insert\_symbol(int hashVal,symbol\* newNode){

if (!ht\_s[hashVal]) ht\_s[hashVal]=newNode;

else {

newNode->link=ht\_s[hashVal];

ht\_s[hashVal]=newNode;

}

return 0;

}

int assemble(char\* asmfile,int pass){ // loc와 objcode 생성

char buffer[100];

char tmpBuf[100];

char \*label, \*opcode; // 최대 6글자

char \*opr1,\*opr2;

char \*arb; // opr1의 constant value(e.g. 'EOF', 'X1')

char \*filename=malloc(sizeof(char)\*20); // asmfile의 이름을 저장

char \*format;

int line=1; // 몇번째 줄인지

int hashVal;

int objcode;

int opVal; // #constant의 int 형변환 결과

int error\_flag=0;

FILE \*fp, \*lfp; // fp는 asmfile용, lfp는 list file용파일 포인터

fp=fopen(asmfile,"r");

if (pass==2){ // w모드로

filename=asmfile;

int nameLen=strlen(asmfile); // asm 파일 이름의 길이

filename[nameLen-3]='l';

filename[nameLen-2]='s';

filename[nameLen-1]='t'; // <2\_5.lst> 형태로 변경

lfp=fopen(filename,"w"); // intermediate file

}

if (!fp){

printf("error : read file open error\n");

return -1;

}

// read first input line & set LOCCTR.

fgets(buffer,100,fp);

strcpy(tmpBuf,buffer);

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

label=strtok(tmpBuf," \n"); // copy 저장(LABEL)

opcode=strtok(NULL," \n"); // START 저장(OPCODE)

opr1=strtok(NULL," \n"); // start번지인 0을 저장(OPR1)

if (!strcmp(opcode,"START")){ // if opcode == 'START', then

LOCCTR=strtol(opr1,NULL,16); // save opr1 as starting address

start=LOCCTR; // start address is set

if(pass==2) fprintf(lfp,"%-3d \t%04X \t%s",line\*5,LOCCTR,buffer); // write line to inter~

line++;

}

else {

printf("start address doesn't exist.\n");

printf("start address is set to 0.\n");

LOCCTR=0;

start=LOCCTR;

if(pass==2) fprintf(lfp,"%-3d \t%04X \t%s",line\*5,LOCCTR,buffer);

line++;

}

// While (OPCODE!=END) DO..

while (fgets(buffer,100,fp)!=NULL){

if (buffer[0]=='.') { // 주석인 경우

line++;

continue;

}

else if (strlen(buffer)<7){

strcpy(tmpBuf,buffer);

char\* check=strtok(tmpBuf," \n");

if(check==NULL) {line++; continue;} // 53-54번째 줄

else{ // 그 외 문자

error\_flag=-1;

error\_line=line;

break;

}

}

else { // 주석도 공백도 아닌 경우

strcpy(tmpBuf,buffer);

tmpBuf[strlen(tmpBuf)-1]='\0'; // 개행문자 빼고

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

strncpy(check,buffer+7,6);

if (pass==2){

if (!strncmp(check,"BASE",4)||!strncmp(check,"END",3)) fprintf(lfp,"%-3d \t \t%s",line\*5,buffer);

else {

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

fprintf(lfp,"%-3d \t%04X \t%s",line\*5,LOCCTR,buffer);

}

}

if (strncmp(buffer," ",6)){ // 라벨 0

label=strtok(tmpBuf," \n");

if(pass==1){

// 심볼테이블에(ht\_s) 라벨이 이미 있는지 검사

if(findSymbol(label)) { // not found -> insert\_symbol

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

newNode->opcode=malloc(sizeof(char)\*6);

strcpy(newNode->opcode,label);

newNode->loc=LOCCTR;

newNode->link=NULL;

hashVal=h\_s(label);

insert\_symbol(hashVal,newNode); // 심볼 삽입

}

else { // found -> set error flag

printf("error : duplicate symbol\n");

error\_flag=-1;

error\_line=line;

break;

}

}

opcode=strtok(NULL," \n"); // 마저 자르기

opr1=strtok(NULL," ,\n");

opr2=strtok(NULL," \n");

} // SYMTAB만들기

else { // 라벨X

opcode=strtok(tmpBuf," \n");

opr1=strtok(NULL," ,\n");

opr2=strtok(NULL," \n");

}

}

// loc

if (!strcmp(opcode,"END")){ // directive - end

end=LOCCTR;

break;

}

else{

if (!strcmp(opcode,"BASE")) { // directive - base

line++;

continue;

}

else{ //상수, 변수

if (opcode[0]=='+'){ // format 4

LOCCTR+=4;

}

else if (!strcmp(opcode,"WORD")){

LOCCTR+=3;

}

else if (!strcmp(opcode,"RESW")){

LOCCTR+=3\*strtol(opr1,NULL,10);

}

else if (!strcmp(opcode,"RESB")){

LOCCTR+=strtol(opr1,NULL,10);

}

else if (!strcmp(opcode,"BYTE")){

if (opr1[0]=='X') LOCCTR+=1;

else if (opr1[0]=='C') LOCCTR+=strlen(buffer)-17;

}

else { // opcode가 들어온 경우(format 4 제외)

if (findOpcode\_2(opcode)){

printf("error : can't find that opcode.\n");

error\_line=line;

error\_flag=-1;

break;

}

else if (!strcmp(fm,"1")) LOCCTR+=1;

else if (!strcmp(fm,"2")) LOCCTR+=2;

else LOCCTR+=3;

}

}

line++;

} // [끝] opcode가 END가 아닌 경우

// obj

if (pass==2){

if (!strcmp(opcode,"BASE")){

if(findSymbol(opr1)) { // SYMTAB에 없다면 -1리턴

printf("error : can't find the symbol in table.\n");

error\_line=line;

error\_flag=-1;

break;

}

}

else if (!strcmp(opcode,"RESB")||!strcmp(opcode,"RESW")){

fprintf(lfp,"%c",0x0A);

continue;

}

else if (!strcmp(opcode,"BYTE")){

if (opr1[0]=='X'){

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

arb[2]='\0';

strncpy(arb,buffer+16,2);

objcode=strtol(arb,NULL,16);

fprintf(lfp,"\t%02X",objcode);

}

else if (opr1[0]=='C'){

arb=malloc(sizeof(char)\*(strlen(buffer)-16));

arb[strlen(buffer)-17]='\0';

opr1++;

opr1++;

strncpy(arb,opr1,strlen(buffer)-17);

int i=0;

fprintf(lfp,"\t");

for (i=0;i<strlen(arb);i++) fprintf(lfp,"%02X",arb[i]);

}

}

else if (!strcmp(opcode,"WORD")){

arb=malloc(sizeof(char)\*6); // max Loc : FFFF(=65535)

arb[strlen(opr1)]='\0';

strncpy(arb,opr1,strlen(opr1));

objcode=strtol(arb,NULL,16);

fprintf(lfp,"\t%06X",objcode);

fprintf(lfp,"%c",0x0A);

}

else { // opcode가 들어온 경우

if (buffer[7]=='+') opcode++;

if (buffer[14]=='#'||buffer[14]=='@') opr1++;

findOpcode\_2(opcode); // 전역변수 fm,ob에 포맷과 opcode 값 저장

if(!strcmp(fm,"1")){

fprintf(lfp,"\t%02X",ob);

}

else if (!strcmp(fm,"2")) {

ob<<=8; // 레지스터 자리

if(!strcmp(opr1,"A")) ob+=0x00;

else if (!strcmp(opr1,"X")) ob+=0x10;

else if (!strcmp(opr1,"L")) ob+=0x20;

else if (!strcmp(opr1,"PC")) ob+=0x80;

else if (!strcmp(opr1,"SW")) ob+=0x90;

else if (!strcmp(opr1,"B")) ob+=0x30;

else if (!strcmp(opr1,"S")) ob+=0x40;

else if (!strcmp(opr1,"T")) ob+=0x50;

else if (!strcmp(opr1,"F")) ob+=0x60;

if(opr2) { // 레지스터2도 있는 경우

if (!strcmp(opr2,"A")) ob+=0x00;

else if (!strcmp(opr2,"X")) ob+=0x01;

else if (!strcmp(opr2,"L")) ob+=0x02;

else if (!strcmp(opr2,"PC")) ob+=0x08;

else if (!strcmp(opr2,"SW")) ob+=0x09;

else if (!strcmp(opr2,"B")) ob+=0x03;

else if (!strcmp(opr2,"S")) ob+=0x04;

else if (!strcmp(opr2,"T")) ob+=0x05;

else if (!strcmp(opr2,"F")) ob+=0x06;

}

if(!opr2) fprintf(lfp,"\t\t%04X",ob);

else fprintf(lfp,"\t%04X",ob);

}

else { // format 3 or 4

if(buffer[14]=='@') ob|=0x02; // ni=10

else if(buffer[14]=='#') ob|=0x01; // ni=01

else { ob|=0x03;} // ni=11

ob<<=4; // xbpe가 될 4비트 자리(fm4면 +20, fm3이면 +12씩)

if (!opr1) { // mnemonic만 존재

ob<<=12;

fprintf(lfp,"\t\t%06X",ob);

}

else{ // mnemonic, operand(s)

if (opr2){

if(!strcmp(opr2,"X")){

ob|=0x008; // xbpe=1000

}

}

if((buffer[14]=='#')&&('0'<=buffer[15]&&buffer[15]<='9')) {

loc=strtol(opr1,NULL,10);

}

else{

if(findSymbol(opr1)){

printf("error : symbol doesn't exist in SYMTAB.\n");

error\_line=line;

error\_flag=-1;

break;

}

}

if (buffer[7]=='+') {

ob|=0x001; // e만 1

ob<<=20; // 주소 20빗

ob+=loc; // findSymbol할 때 저장한 opr1의 loc값

fprintf(lfp,"\t%08X",ob-1);

}

else {

if(buffer[14]=='#'&&'0'<=buffer[15]&&buffer[15]<='9') {ob|=0x000;}

else if(LOCCTR+3-2048<=loc&&loc<=LOCCTR+2047+3){

ob|=0x002; //xbpe=0010

loc-=(LOCCTR+1); // PC의 위치 : LOCCTR+3

}

else if (base<=loc&&loc<=base+4095){

ob|=0x004; //xbpe=0100

loc-=base;

}

else {

printf("error : can't use format 3..\n");

error\_line=line;

error\_flag=-1;

break;

}

ob<<=12;

loc&=0x000FFF;

ob+=loc;

fprintf(lfp,"\t%06X",ob);

}

}

}

}

fprintf(lfp,"%c",0x0A);

} // if pass 2

} //while

end=LOCCTR;

fclose(fp);

if(pass==2){

fclose(lfp);

}

if (error\_flag==-1) {

printf("error line is : %d\n",error\_line);

return -1;

}

return 0;

} // end of ASSEMBLE

int showSt(void){ // symbol table을 보여준다.

symbolPtr worker;

int i=0;

int indent;

for (worker=ht\_s[i];i<26;i++){

worker=ht\_s[i];

if (!worker) continue;

while (worker){

indent=8-strlen(worker->opcode);

printf(" %s",worker->opcode);

while(indent>0){

printf(" ");

indent--;

}

printf("%04X\n",worker->loc);

worker=worker->link;

}

}

return 0;

}

int makeObFile(char \*asmfile){ // .obj파일을 만든다.

FILE\* lfp, \*ofp; // .lst와 .obj 파일 포인터

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

lst=asmfile;

int nameLen=strlen(asmfile);

lst[nameLen-3]='l';

lst[nameLen-2]='s';

lst[nameLen-1]='t';

lfp=fopen(lst,"r"); // 읽기모드로 리스팅 파일 열기

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

strcpy(obj,lst);

obj[nameLen-3]='o';

obj[nameLen-2]='b';

obj[nameLen-1]='j';

ofp=fopen(obj,"w"); //쓰기모드로 .obj 파일 열기

//H

char name[7]={0,};

char buffer[100]={0,};

char tmpBuf[100]={0,};

fgets(buffer,100,lfp);

strncpy(name,buffer+11,6);

fprintf(ofp,"H%6s%06X%06X\n",name,start,end-start);

//T

char loc[4]={0,};

loc[4]='\0';

char label[7]={0,};

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

char compOpcode[3]={0,}; // RES 판별

compOpcode[3]='\0';

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

char \*objco=malloc(sizeof(char)\*20); // 오브젝트 코드

char tmpArr[80]={0,};// 오브젝트 코드를 이어붙임

int length=0; // 바이트 단위 길이

while(fgets(buffer,100,lfp)!=NULL){

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

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

strcpy(tmpBuf,buffer);

if(!length) {

strncpy(loc,buffer+5,4);

}

if (buffer[5]==' ') continue; // opcode가 base, end일 때

opcode=strtok(tmpBuf+18," \t\n");

operand=strtok(NULL," \t\n");

objco=strtok(NULL," \t\n");

if (objco==NULL) continue; // RSUB과 같이 opr가 없어서 objco에 NULL이 입력되는 경우

strncpy(compOpcode,opcode,3);

if (!strcmp(compOpcode,"END")) {

fprintf(ofp,"T00%4s%02X%s\n",loc,length,tmpArr);

break;

}

else{

if(!strcmp(compOpcode,"RES")){ //변수면 마저 출력하고 개행

if (!length) continue; // 아무것도 안써져있는 경우

else{

fprintf(ofp,"T00%4s%02X%s\n",loc,length,tmpArr);

length=0;

memset(tmpArr,'\0',80);

}

}

else if (length+(strlen(objco)/2)>0x1E) { // 글자수 초과하면 마저 출력하고 개행

fprintf(ofp,"T00%4s%02X%s\n",loc,length,tmpArr);

memset(tmpArr,'\0',80);

strcat(tmpArr,objco);

strncpy(loc,buffer+5,4);

length=(strlen(objco)/2);

}

else {

strcat(tmpArr,objco);

length+=(strlen(objco)/2);

}

}

}

//E

fprintf(ofp,"E%06X\n",start);

fclose(ofp);

fclose(lfp);

return 0;

}