|  |
| --- |
|  |

|  |
| --- |
| **시스템 프로그래밍 플젝 #1** |
| **Control Section방식 어셈블러** |
|  |
|  |
|  |
| **교수 : 최재영 교수님** |
| **가반 114번**  **컴퓨터학부 20132361 박진수** |
|  |

**목 차**

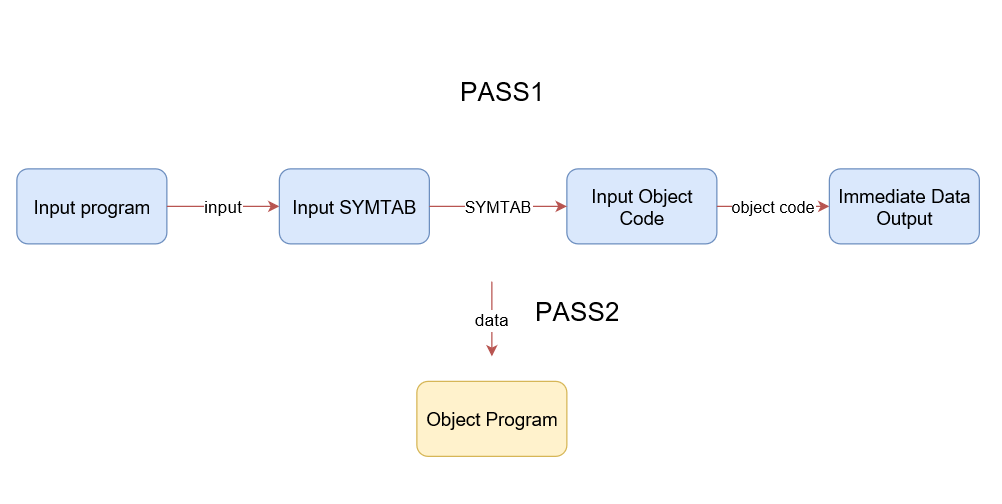
1. **동기/목적**
2. **설계/구현 아이디어**
3. **수행결과**
4. **결론 및 보충할 점**
5. **소스코드**
6. **동기/목적**

**과제4의 파싱 프로그램으로 SIC/XE 머신을 이해한 후 Control Section방식의 어셈블러를 만들어 어셈블러가 아닌 메모리 접근 방식에 좀더 유연한 방식으로 만들어 좀 더 효율적인 어셈블러를 구현하고 어셈블러에서 편리하게 상수를 정의하는 리터럴(Literal)과 기호 정의문 중 EQU(equate), 수식들을 사용한 어셈블러를 구현함, 뿐만 아니라 SIC/XE머신의 추가적인 레지스터와 4형식의 사용, 간접주소지정 방식과 직접주소지정 박식, 4형식의 사용 등의 특징을 이해하고 구현하였음.**

**주어진 C 소스코드 외 헤더파일을 이용하여 SIC/XE 소스를 Object Program으로 변환하는 과정을 이해하고 이 후 확장되는 과제 내용에 맞추어 프로그램의 확장성을 효과적으로 증징시키기 위한 기본 지식을 학습**

**.**

1. **설계/구현 아이디어**



**SYMTAB 저장(sym\_table 구조체)**

**토큰테이블을 한 개씩 읽어 들어 symbol이 있을 경우 SYMTAB에 저장하고 location counter값도 계산하여 LOCCTR 배열에 저장함. EQU에는 절대값을 저장하고 연산이 있는 경우에는 search\_cal함수로 연산이 있는지 검사하여 loc값을 계산하여 저장함. 명령어의 경우 형식에 따라 loc값을 더해서 LOCCTR배열에 저장하였음. LTORG을 만나거나 프로그램 끝에서 리터럴 폴을 비울 경우에는 literals배열에 저장하였던 값을 추가하고 리터럴 테이블에 추가하였음.**

**Object Code 저장(addr 구조체)**

**토큰테이블을 한 개씩 읽어 들어 명령어일 경우에는 opcode와 어드레싱 모드와 형식, x레지스터에 따라 nixbpe 값과 어드레싱 모드에 따른 주소값을 addr 구조체에 저장함. 참조값을 사용하는 경우에는 주소값에 0을 저장. 명령어가 아니고 WORD, BYTE나 리터럴 값일 경우에는 연산이 있는 경우 없는 경우, 참조값을 사용하는 경우를 고려하여 주소값을 저장함.**

**Immediate Data Ouput**

**addr구조체에 저장한 값을 명령어 코드에 맞게 출력함.**

**Object Program Output(PASS 2)**

**H레코드에는 addr구조체의 처음 주소값과 pass1에서 section으로 나눈 각 루틴의 마지막 주소값을 저장하였던 값을 출력, T레코드일 경우 30바이트의 수 제한이나 LTORG, section에 의한 조건으로 개행하여 길이와 첫 주소값과 함께 object code를 출력하였음. M레코드의 경우 PASS1에서 참조값을 사용한 경우 addr구조체에 확인 값을 저장하여 명령어 코드의 주소와 수정할 길이, 수정할 operand를 출력, section의 루틴이 끝날 경우에는 E레코드를 출력**

**3. 수행결과**

**Input**

COPY START 0 COPY FILE FROM IN TO OUTPUT

EXTDEF BUFFER,BUFEND,LENGTH

EXTREF RDREC,WRREC

FIRST STL RETADR SAVE RETURN ADDRESS

CLOOP +JSUB RDREC READ INPUT RECORD

LDA LENGTH TEST FOR EOF (LENGTH = 0)

COMP #0

JEQ ENDFIL EXIT IF EOF FOUND

+JSUB WRREC WRITE OUTPUT RECORD

J CLOOP LOOP

ENDFIL LDA =C'EOF' INSERT END OF FILE MARKER

STA BUFFER

LDA #3 SET LENGTH = 3

STA LENGTH

+JSUB WRREC WRITE EOF

J @RETADR RETURN TO CALLER

RETADR RESW 1

LENGTH RESW 1 LENGTH OF RECORD

LTORG

BUFFER RESB 4096 4096-BYTE BUFFER AREA

BUFEND EQU \*

MAXLEN EQU BUFEND-BUFFER MAXIMUM RECORD LENGTH

RDREC CSECT

EXTREF BUFFER,LENGTH,BUFEND

CLEAR X CLEAR LOOP COUNTER

CLEAR A CLEAR A TO ZERO

CLEAR S CLEAR S TO ZERO

LDT MAXLEN

RLOOP TD INPUT TEST INPUT DEVICE

JEQ RLOOP LOOP UNTIL READY

RD INPUT READ CHARACTER INTO REGISTER A

COMPR A,S TEST FOR END OF RECORD (X'00')

JEQ EXIT EXIT LOOP IF EOR

+STCH BUFFER,X STORE CHARACTER IN BUFFER

TIXR T LOOP UNLESS MAX LENGTH

JLT RLOOP HAS BEEN REACHED

EXIT +STX LENGTH SAVE RECORD LENGTH

RSUB RETURN TO CALLER

INPUT BYTE X'F1' CODE FOR INPUT DEVICE

MAXLEN WORD BUFEND-BUFFER

WRREC CSECT

EXTREF LENGTH,BUFFER

CLEAR X CLEAR LOOP COUNTER

+LDT LENGTH

WLOOP TD =X'05' TEST OUTPUT DEVICE

JEQ WLOOP LOOP UNTIL READY

+LDCH BUFFER,X GET CHARACTER FROM BUFFER

WD =X'05' WRITE CHARACTER

TIXR T LOOP UNTIL ALL CHARACTERS

JLT WLOOP HAVE BEEN WRITTEN

RSUB RETURN TO CALLER

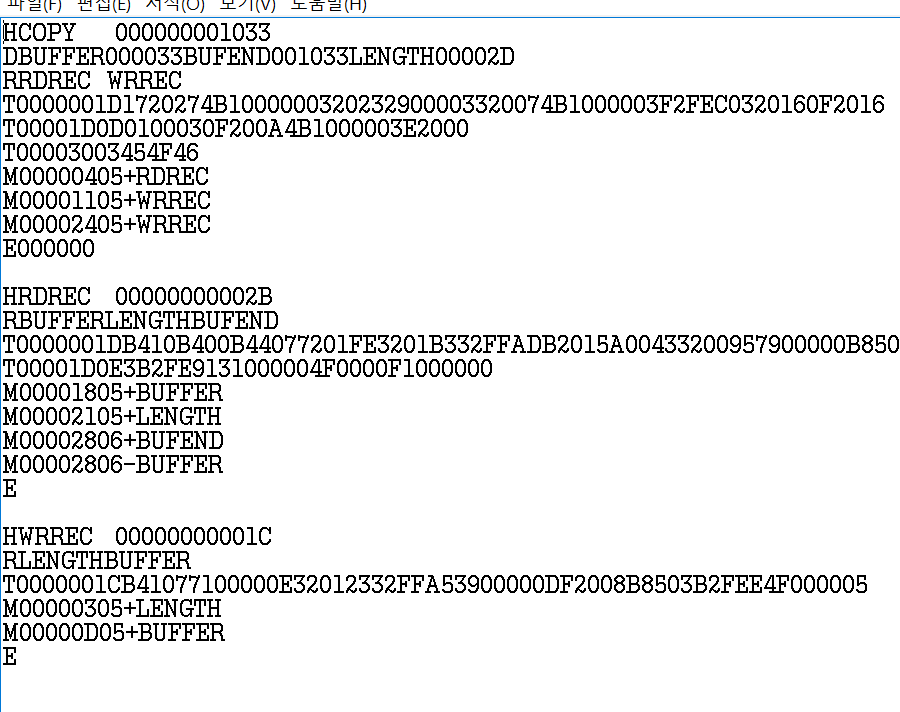
END FIRST

**Immediate Data(화면출력)**





**Output(파일 출력)**



**4. 결론 및 보충할 점**

**파싱을 통한 토큰 테이블을 통하여 SYMTAB을 만들고 어셈블러 지시어를 처라하고 LOCCTR값과 주소값을 구한 PASS 1을 구현하고 PASS 1을 통하여 기계어 코드로 번역한 Object Program을 완성하였음.**

**SIC/XE 어셈블러에서 Control Section을 방식으로 구현하여 각각의 루틴을 독립적인 부분으로 나누어 그에 따른 메모리 저장과 접근의 이점을 이해하고 외부 참조값과 정의값에 의한 모르는 주소값을 수정 레코드로 수정하여 로더 할 때의 문제점을 해결하고 편리함을 위한 Literal과 EQU를 사용함에 따라 편리함과 신뢰도를 향상 시킬 수 있음.**

**고려해야 할 상황이 많아 그에 따른 프로그램의 복잡도가 늘어남. 좀 더 효율적인 알고리즘과 자료구조의 활용이 필요하고 불필요한 변수나 구조체의 사용을 줄임으로써 프로그램의 복잡도와 실행속도를 줄여가면서 보충하겠음.**

**5. 소스코드**

**#include "my\_assembler\_114.h"**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**FILE \* rp;**

**int main(void)**

**{**

**if (init\_my\_assembler()< 0)**

**{**

**printf("init\_my\_assembler: error\n");**

**return -1;**

**}**

**make\_opcode\_output("output(hw4).txt");**

**assem\_pass1();**

**assem\_pass2();**

**//동적할당 해제**

**for (int i = 0; i < line\_num; i++)**

**{**

**free(table[i]->operand[3]);**

**free(table[i]->operand[2]);**

**free(table[i]->operand[1]);**

**free(table[i]->operand[0]);**

**free(table[i]->comment);**

**free(table[i]->operate);**

**free(table[i]->label);**

**free(table[i]);**

**}**

**return 0;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 프로그램 초기화를 위한 자료구조 생성 및 파일을 읽는 함수이다.**

**\* 매계 : 없음**

**\* 반환 : 정상종료 = 0 , 에러 발생 = -1**

**\* 주의 : 각각의 명령어 테이블을 내부에 선언하지 않고 관리를 용이하게 하기**

**\* 위해서 파일 단위로 관리하여 프로그램 초기화를 통해 정보를 읽어 올 수 있도록**

**\* 구현하였다.**

**\* ----------------------------------------------------------------------------------**

**\*/**

**int init\_my\_assembler(void)**

**{**

**int result = 1;**

**if ((result = init\_inst\_file("inst.data")) < 0)**

**{**

**return -1;**

**}**

**if ((result = init\_input\_file("input.txt")) < 0)**

**{**

**return -1;**

**}**

**return result;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 머신을 위한 기계 코드목록 파일을 읽어 기계어 목록 테이블(inst\_table)을**

**\* 생성하는 함수이다.**

**\* 매계 : 기계어 목록 파일**

**\* 반환 : 정상종료 = 0 , 에러 < 0**

**\* 주의 : 기계어 목록파일 형식**

**\* || 이름 || opcode || 형식 || operand 개수 ||**

**\* ----------------------------------------------------------------------------------**

**\*/**

**int init\_inst\_file(char \*inst\_file)**

**{**

**FILE \* insp = fopen(inst\_file, "rt"); //기계어 코드목록 파일**

**int errno;**

**int ccnt = 0;**

**int str\_cnt = 0;**

**char op\_str[4];**

**char format\_str[2];**

**char op\_cnt\_str[2];**

**op\_str[0] = '\0';**

**format\_str[0] = '\0';**

**op\_cnt\_str[0] = '\0';**

**errno = 0;**

**if (insp == NULL) //파일이 없을 경우 -1 리턴**

**{**

**errno = -1;**

**return errno;**

**}**

**for (int i = 0; i < MAX\_INST; i++)**

**{**

**str\_cnt = 0;**

**char \*stop;**

**char str[20];**

**if (fgets(str, 15, insp) != NULL) //파일을 1줄씩 읽음**

**{**

**if (!feof(insp))**

**{**

**str[strlen(str) - 1] = '\0'; //파일 끝이 아닐 경우에는 개행 삭제**

**}**

**//명령어 이름을 공백이나 탭을 제거하고 inst테이블에 저장**

**for (int j = 0; j < 6; j++)**

**{**

**if (str[j] == '\t' || str[j] == ' ')**

**{**

**continue;**

**}**

**else**

**{**

**instruction[i].name[ccnt++] = str[j];**

**if (str[j + 1] == '\t' || str[j + 1] == ' ')**

**{**

**str\_cnt = j + 1;**

**ccnt = 0;**

**break;**

**}**

**}**

**}**

**//명령어의 opcode를 공백이나 탭을 제거하고 inst테이블에 저장**

**for (int j = str\_cnt; j < 10; j++)**

**{**

**if (str[j] == '\t' || str[j] == ' ')**

**{**

**continue;**

**}**

**else**

**{**

**instruction[i].op[ccnt++] = str[j];**

**if (str[j + 1] == '\t' || str[j + 1] == ' ')**

**{**

**str\_cnt = j + 1;**

**ccnt = 0;**

**op\_str[0] = '\0';**

**break;**

**}**

**}**

**}**

**//명령어의 형식을 공백이나 탭을 제거하고 inst테이블에 저장**

**for (int j = str\_cnt; j < 12; j++)**

**{**

**if (str[j] == '\t' || str[j] == ' ')**

**{**

**continue;**

**}**

**else**

**{**

**format\_str[ccnt++] = str[j];**

**if (str[j + 1] == '\t' || str[j + 1] == ' ' || str[j + 1] == '\0')**

**{**

**instruction[i].format = atoi(format\_str);**

**str\_cnt = j + 1;**

**ccnt = 0;**

**format\_str[0] = '\0';**

**break;**

**}**

**}**

**}**

**//명령어의 operand 갯수를 공백이나 탭을 제거하고 inst테이블에 저장**

**for (int j = str\_cnt; j < 15; j++)**

**{**

**if (str[j] == '\t' || str[j] == ' ')**

**{**

**continue;**

**}**

**else**

**{**

**op\_cnt\_str[ccnt++] = str[j];**

**if (str[j + 1] == '\t' || str[j + 1] == ' ' || str[j + 1] == '\0')**

**{**

**instruction[i].op\_cnt = atoi(op\_cnt\_str);**

**str\_cnt = j + 1;**

**ccnt = 0;**

**op\_cnt\_str[0] = '\0';**

**break;**

**}**

**}**

**}**

**}**

**else**

**{**

**break;**

**}**

**}**

**return errno;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 어셈블리 할 소스코드를 읽어오는 함수이다.**

**\* 매계 : 어셈블리할 소스파일명**

**\* 반환 : 정상종료 = 0 , 에러 < 0**

**\* 주의 :**

**\* ----------------------------------------------------------------------------------**

**\*/**

**int init\_input\_file(char \*input\_file)**

**{**

**rp = fopen(input\_file, "rt");**

**int errno;**

**errno = 0;**

**//파일이 없을경우 -1 리턴**

**if (rp == NULL)**

**{**

**errno = -1;**

**return errno;**

**}**

**token\_parsing(0); //읽을 파일에서 첫번째 줄부터 파싱시작**

**fclose(rp);**

**return errno;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 소스 코드를 읽어와 토큰단위로 분석하고 토큰 테이블을 작성하는 함수이다.**

**\* 패스 1로 부터 호출된다.**

**\* 매계 : 소스코드의 라인번호**

**\* 반환 : 정상종료 = 0 , 에러 < 0**

**\* 주의 : assembler 프로그램에서는 라인단위로 토큰 및 오브젝트 관리를 하고 있다.**

**\* ----------------------------------------------------------------------------------**

**\*/**

**int token\_parsing(int index)**

**{**

**int str\_cnt = 0;**

**int no\_operand = 0;**

**int no\_comment = 0;**

**int operand\_cnt = 0;**

**char space[1] = "";**

**char m\_space[1] = "";**

**for (int i = 0; i < MAX\_LINES; i++)**

**{**

**str\_cnt = 0;**

**operand\_cnt = 0;**

**char str[150];**

**//구조체 포인터 배열 table을 동적할당**

**table[i] = (token \*)malloc(sizeof(token));**

**table[i]->label = (char \*)malloc(sizeof(char) \* 10);**

**table[i]->operate = (char \*)malloc(sizeof(char) \* 10);**

**table[i]->comment = (char \*)malloc(sizeof(char) \* 100);**

**table[i]->operand[0] = (char \*)malloc(sizeof(char) \* 20);**

**table[i]->operand[1] = (char \*)malloc(sizeof(char) \* 10);**

**table[i]->operand[2] = (char \*)malloc(sizeof(char) \* 10);**

**table[i]->operand[3] = (char \*)malloc(sizeof(char) \* 30);**

**//널값으로 초기화**

**strcpy(table[i]->label, m\_space);**

**strcpy(table[i]->operate, m\_space);**

**strcpy(table[i]->comment, m\_space);**

**strcpy(table[i]->operand[0], m\_space);**

**strcpy(table[i]->operand[1], m\_space);**

**strcpy(table[i]->operand[2], m\_space);**

**strcpy(table[i]->operand[3], m\_space);**

**if (fgets(str, 150, rp) != NULL) //파일에서 1줄씩 읽어옴**

**{**

**if (!feof(rp))**

**{**

**str[strlen(str) - 1] = '\0'; //파일 끝이 아닐 경우 개행제거**

**}**

**if (str[0] == '\t' || str[0] == '\0')**

**{**

**strcpy(table[i]->label, "\t"); //label이 없을경우 탭으로 저장**

**}**

**else**

**{**

**//공백이 아닐경우 label에 입력된 값 저장**

**for (int j = 0; j < 150; j++)**

**{**

**if (str[j] == '\t' || str[j] == '\0')**

**{**

**continue;**

**}**

**else**

**{**

**//값이 없는 주소의 값에 널 값을 넣음**

**\*space = \*(str + j);**

**strncat(table[i]->label, space, 1);**

**if (str[j + 1] == '\t' || str[j + 1] == '\0')**

**{**

**str\_cnt = j + 1;**

**break;**

**}**

**}**

**}**

**}**

**//공백을 제거하고 label과 같이 operate(명령어) 저장**

**for (int j = str\_cnt; j < 150; j++)**

**{**

**if (str[j] == '\t' || str[j] == '\0')**

**{**

**continue;**

**}**

**else**

**{**

**\*space = \*(str + j);**

**strncat(table[i]->operate, space, 1);**

**if (str[j + 1] == '\t' || str[j + 1] == '\0')**

**{**

**str\_cnt = j + 1;**

**break;**

**}**

**}**

**}**

**//공백을 제거하고 위와 같이 operand값 저장(아직 구별을 하지 않으므로 operand[0]에 모두 저장)**

**int k = 0;**

**for (int j = str\_cnt; j < 150; j++)**

**{**

**if (str[j] == '\t')**

**{**

**k++;**

**//operand가 없을경우 널값으로 저장**

**if (k == 2)**

**{**

**strcpy(table[i]->operand[3], "\0");**

**strcpy(table[i]->operand[0], "\0");**

**str\_cnt = j + 3;**

**break;**

**}**

**else**

**{**

**continue;**

**}**

**}**

**//operand와 comment모두 없을 경우 널값으로 모두 저장**

**else if (str[j] == '\0')**

**{**

**no\_operand = 1;**

**strcpy(table[i]->operand[0], "\0");**

**strcpy(table[i]->operand[3], "\0");**

**strcpy(table[i]->comment, "\0");**

**break;**

**}**

**else**

**{**

**\*space = \*(str + j);**

**strncat(table[i]->operand[3], space, 1);**

**switch (operand\_cnt)**

**{**

**case 0:**

**{**

**if(str[j]!=',' && str[j] != '-' && str[j] != '+')**

**strncat(table[i]->operand[0], space, 1);**

**break;**

**}**

**case 1:**

**{**

**if (str[j] != ',' && str[j] != '-' && str[j] != '+')**

**strncat(table[i]->operand[1], space, 1);**

**break;**

**}**

**case 2:**

**{**

**if (str[j] != ',' && str[j] != '-' && str[j] != '+')**

**strncat(table[i]->operand[2], space, 1);**

**break;**

**}**

**default:**

**break;**

**}**

**if (str[j] == ',' || str[j] == '-' || str[j] == '+')**

**{**

**operand\_cnt++;**

**}**

**//strncat(table[i]->operand[0], space, 1);**

**if (str[j + 1] == '\t' || str[j + 1] == '\0')**

**{**

**str\_cnt = j + 1;**

**break;**

**}**

**}**

**}**

**k = 0;**

**//위와 같이 comment도 저장**

**for (int j = str\_cnt; j < 150; j++)**

**{**

**if (no\_operand) //operand와 comment 없을 경우 넘어감**

**{**

**break;**

**}**

**if (str[j] == '\t' || str[j] == '\0')**

**{**

**strcpy(table[i]->comment, "\0");**

**break;**

**}**

**else**

**{**

**\*space = \*(str + j);**

**strncat(table[i]->comment, space, 1);**

**if (str[j + 1] == '\t' || str[j + 1] == '\0')**

**{**

**break;**

**}**

**}**

**}**

**}**

**else**

**{**

**line\_num = i;**

**break;**

**}**

**}**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 입력 문자열이 기계어 코드인지를 검사하는 함수이다.**

**\* 매계 : 토큰 단위로 구분된 문자열**

**\* 반환 : 정상종료 = 기계어 테이블 인덱스, 에러 or 기계어가 아님 = -1**

**\* ----------------------------------------------------------------------------------**

**\*/**

**int search\_opcode(char \*str)**

**{**

**int cnt = 0;**

**char a[20] = "+";**

**if (str == NULL)**

**{**

**return -1;**

**}**

**//명령어 테이블에 있는지 검사**

**for (int j = 0; j < MAX\_INST; j++)**

**{**

**strcat(a, instruction[j].name); //4형식일 경우**

**if (!strcmp(str, instruction[j].name))**

**{**

**if (instruction[j].format == 4)**

**{**

**instruction[j].format = 3;**

**}**

**cnt = 1;**

**return j;**

**}**

**else if (!strcmp(str, a))**

**{**

**cnt = 1;**

**instruction[j].format = 4;**

**return j;**

**}**

**a[0] = '\0';**

**strcat(a, "+");**

**}**

**if (cnt == 0)**

**{**

**return -1;**

**}**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 입력된 문자열의 이름을 가진 파일에 프로그램의 결과를 저장하는 함수이다.**

**\* 여기서 출력되는 내용은 명령어 옆에 OPCODE가 기록된 표(과제 4번) 이다.**

**\* 매계 : 생성할 오브젝트 파일명**

**\* 반환 : 없음**

**\* 주의 : 만약 인자로 NULL값이 들어온다면 프로그램의 결과를 표준출력으로 보내어**

**\* 화면에 출력해준다.**

**\* -----------------------------------------------------------------------------------**

**\*/**

**void make\_opcode\_output(char \*file\_name)**

**{**

**char \*sps = "";**

**FILE \* wp = fopen(file\_name, "wt");**

**int cnt = 0;**

**int k;**

**//파일이 NULL로 들어올 경우 표준출력**

**if (wp == NULL)**

**{**

**fprintf(wp, "%s", stdout);**

**}**

**char start[10] = "START";**

**//명령어 테이블에서 확인 후 opcode값을 token 테이블에 저장**

**for (int i = 0; i < MAX\_LINES; i++)**

**{**

**if (table[i]->operate[0] == NULL)**

**{**

**break;**

**}**

**if (!strcmp(table[i]->operate, start)) //start일 경우 op코드값에 널값 저장**

**{**

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

**continue;**

**}**

**k = search\_opcode(table[i]->operate);**

**//명령어 테이블에 일치하는 명령어가 없을 경우 널값 저장**

**if (k < 0)**

**{**

**strcpy(table[i]->op, "\0");**

**}**

**else**

**{**

**strcpy(table[i]->op, instruction[k].op);**

**}**

**}**

**//operate가 NULL값일 때까지 ouput파일에 출력**

**for (int i = 0; i < MAX\_LINES; i++)**

**{**

**if (table[i]->operate[0] == NULL)**

**{**

**break;**

**}**

**if (!strcmp(table[i]->label, "\t"))**

**fprintf(wp, "%s%s\t%s\t%s", table[i]->label, table[i]->operate, table[i]->operand[3], table[i]->op);**

**else**

**fprintf(wp, "%s\t%s\t%s\t%s", table[i]->label, table[i]->operate, table[i]->operand[3], table[i]->op);**

**fputc('\n', wp);**

**}**

**fclose(wp);**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 어셈블리 코드를 위한 패스1과정을 수행하는 함수이다.**

**\* 패스1에서는..**

**\* 1. 프로그램 소스를 스캔하여 해당하는 토큰단위로 분리하여 프로그램 라인별 토큰**

**\* 테이블을 생성한다.**

**\***

**\* 매계 : 없음**

**\* 반환 : 정상 종료 = 0 , 에러 = < 0**

**\* 주의 : 현재 초기 버전에서는 에러에 대한 검사를 하지 않고 넘어간 상태이다.**

**\* 따라서 에러에 대한 검사 루틴을 추가해야 한다.**

**\***

**\* -----------------------------------------------------------------------------------**

**\*/**

**static int assem\_pass1(void)**

**{**

**input\_sym();**

**input\_address();**

**data\_output();**

**return 0;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 어셈블리 코드를 기계어 코드로 바꾸기 위한 패스2 과정을 수행하는 함수이다.**

**\* 패스 2에서는 프로그램을 기계어로 바꾸는 작업은 라인 단위로 수행된다.**

**\* 다음과 같은 작업이 수행되어 진다.**

**\* 1. 실제로 해당 어셈블리 명령어를 기계어로 바꾸는 작업을 수행한다.**

**\* 매계 : 없음**

**\* 반환 : 정상종료 = 0, 에러발생 = < 0**

**\* 주의 :**

**\* -----------------------------------------------------------------------------------**

**\*/**

**static int assem\_pass2(void)**

**{**

**make\_objectcode\_output("output\_114.txt");**

**return 0;**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : 입력된 문자열의 이름을 가진 파일에 프로그램의 결과를 저장하는 함수이다.**

**\* 여기서 출력되는 내용은 object code (프로젝트 1번) 이다.**

**\* 매계 : 생성할 오브젝트 파일명**

**\* 반환 : 없음**

**\* 주의 : 만약 인자로 NULL값이 들어온다면 프로그램의 결과를 표준출력으로 보내어**

**\* 화면에 출력해준다.**

**\***

**\* -----------------------------------------------------------------------------------**

**\*/**

**void make\_objectcode\_output(char \*file\_name)**

**{**

**FILE \* wp = fopen(file\_name, "wt");**

**int l\_cnt = 0;**

**int l\_cnt2 = 0;**

**int k = 0;**

**int rr\_cnt = 0;**

**int max = 30;**

**int real\_cnt = 0;**

**int count = 0;**

**int format = 0;**

**int enter = 0;**

**int start\_cnt = 0;**

**int m\_record[100] = { 0 };**

**int is\_word[100] = { 0 };**

**char m\_str[100][15];**

**int m\_cnt = 0;**

**int t\_cnt = 0;**

**if (wp == NULL)**

**{**

**fprintf(wp, "%s", stdout);**

**}**

**for (int i = 0; i < loc\_cnt; i++, l\_cnt++)**

**{**

**if (!strcmp(table[i]->operate, "START"))**

**{**

**int j = 0;**

**rr\_cnt++;**

**start\_cnt = 1;**

**fprintf(wp, "H%s\t%06X%06X\n", table[i]->label, 0, end[rr\_cnt]); //start의 헤더레코드**

**continue;**

**}**

**else if (!strcmp(table[i]->operate, "CSECT"))**

**{**

**if (m\_cnt != 0)**

**{**

**fputc('\n', wp);**

**for (int m = 0; m < m\_cnt; m++) //수정 레코드를 넣어줌**

**{**

**if (is\_word[m] == 1)**

**{**

**fprintf(wp, "M%06X06+%s\n", m\_record[m], m\_str[m]);**

**}**

**else if (is\_word[m] == 2)**

**{**

**fprintf(wp, "M%06X06%s\n", m\_record[m], m\_str[m]);**

**}**

**else if (is\_word[m] == 3)**

**{**

**fprintf(wp, "M%06X06%s\n", m\_record[m], m\_str[m]);**

**}**

**else**

**{**

**fprintf(wp, "M%06X05+%s\n", m\_record[m], m\_str[m]);**

**}**

**}**

**}**

**if (rr\_cnt == 1)**

**{**

**fprintf(wp, "E%06d\n\n", LOCCTR[0]);**

**rr\_cnt++;**

**}**

**else**

**{**

**fprintf(wp, "E\n\n");**

**rr\_cnt++;**

**}**

**count = 0;**

**fprintf(wp, "H%s\t%06X%06X\n", table[i]->label, 0, end[rr\_cnt]); //나눠진 섹션의 헤더레코드**

**l\_cnt2 = LOCCTR[l\_cnt];**

**start\_cnt = 1;**

**for (int j = 0; j < m\_cnt; j++)**

**{**

**is\_word[j] = 0;**

**}**

**m\_cnt = 0;**

**continue;**

**}**

**else if (!strcmp(table[i]->operate, "EXTDEF")) //D 레코드**

**{**

**int loc1 = -1, loc2 = -1, loc3 = -1;**

**for (int p = 0; p < sym\_cnt; p++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[p].symbol))**

**{**

**loc1 = sym\_table[p].addr;**

**}**

**if (!strcmp(table[i]->operand[1], sym\_table[p].symbol))**

**{**

**loc2 = sym\_table[p].addr;**

**}**

**if (!strcmp(table[i]->operand[2], sym\_table[p].symbol))**

**{**

**loc3 = sym\_table[p].addr;**

**}**

**}**

**if (table[i]->operand[2] != NULL)**

**{**

**fprintf(wp, "D%s%06X%s%06X%s%06X\n", table[i]->operand[0], loc1, table[i]->operand[1], loc2, table[i]->operand[2], loc3);**

**}**

**else if (table[i]->operand[1] != NULL)**

**{**

**fprintf(wp, "D%s%06X%s%06X\n", table[i]->operand[0], loc1, table[i]->operand[1], loc2);**

**}**

**else**

**{**

**fprintf(wp, "D%s%06X\n", table[i]->operand[0], loc1);**

**}**

**continue;**

**}**

**else if (!strcmp(table[i]->operate, "EXTREF")) //R 레코드**

**{**

**if (rr\_cnt == 1)**

**{**

**fputc('R', wp);**

**for (int r = 0; r < refer\_table[rr\_cnt].cnt; r++)**

**{**

**fprintf(wp, "%s ", refer\_table[rr\_cnt].refer[r]);**

**}**

**fputc('\n', wp);**

**}**

**else**

**{**

**fputc('R', wp);**

**for (int r = 0; r < refer\_table[rr\_cnt].cnt; r++)**

**{**

**fprintf(wp, "%s", refer\_table[rr\_cnt].refer[r]);**

**}**

**fputc('\n', wp);**

**}**

**continue;**

**}**

**//object code가 없을 경우 넘어감**

**else if (!strcmp(table[i]->operate, "RESB") || !strcmp(table[i]->operate, "RESW") || !strcmp(table[i]->operate, "EQU"))**

**{**

**continue;**

**}**

**else**

**{**

**t\_cnt++;**

**if (addr[l\_cnt].is\_r == 1) //수정할 object code일 경우**

**{**

**//수정할 object code의 명령어가 word일 경우**

**if (!strcmp(table[i]->operate, "WORD"))**

**{**

**if (!strcmp(table[i]->operand[1], "\0"))**

**{**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 1;**

**strcpy(m\_str[m\_cnt], table[i]->operand[0]);**

**m\_cnt++;**

**}**

**else if (!strcmp(table[i]->operand[2], "\0"))**

**{**

**search\_cal(table[i]->operand[3]);**

**char a[20] = "";**

**strcpy(a, cal[0]);**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 1;**

**strcpy(m\_str[m\_cnt], table[i]->operand[0]);**

**m\_cnt++;**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 2;**

**strcat(a, table[i]->operand[1]);**

**strcpy(m\_str[m\_cnt], a);**

**m\_cnt++;**

**}**

**else**

**{**

**search\_cal(table[i]->operand[3]);**

**char a[20] = "";**

**strcpy(a, cal[0]);**

**char b[20] = "";**

**strcpy(b, cal[0]);**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 1;**

**strcpy(m\_str[m\_cnt], table[i]->operand[0]);**

**m\_cnt++;**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 2;**

**strcat(a, table[i]->operand[1]);**

**strcpy(m\_str[m\_cnt], a);**

**m\_cnt++;**

**m\_record[m\_cnt] = LOCCTR[l\_cnt];**

**is\_word[m\_cnt] = 3;**

**strcat(b, table[i]->operand[2]);**

**strcpy(m\_str[m\_cnt], b);**

**m\_cnt++;**

**}**

**}**

**//수정할 object code 추가**

**else**

**{**

**m\_record[m\_cnt] = LOCCTR[l\_cnt] + 1;**

**strcpy(m\_str[m\_cnt], table[i]->operand[0]);**

**m\_cnt++;**

**}**

**}**

**//T 레코드의 줄이 개행될 때**

**if (count == 0)**

**{**

**for (int c = l\_cnt; c < l\_cnt + 12; c++)**

**{**

**if (real\_cnt + 3 > max || addr[c].is\_r == 2 || addr[c].r\_cnt != rr\_cnt)**

**{**

**break;**

**}**

**else**

**{**

**real\_cnt += addr[c].format;**

**}**

**}**

**if (!strcmp(table[i]->operate, "END"))**

**{**

**return;**

**}**

**fputc('T', wp);**

**if (start\_cnt == 1)**

**{**

**fprintf(wp, "%06X%02X", LOCCTR[l\_cnt], real\_cnt);**

**start\_cnt = 0;**

**}**

**else**

**{**

**fprintf(wp, "%06X%02X", LOCCTR[l\_cnt], real\_cnt);**

**}**

**real\_cnt = 0;**

**}**

**//형식에 따라 object code출력**

**k = search\_opcode(table[i]->operate);**

**format = instruction[k].format;**

**switch (format)**

**{**

**case 2:**

**{**

**fprintf(wp, "%02X%02d", addr[l\_cnt].op, addr[l\_cnt].loc);**

**count += 2;**

**break;**

**}**

**case 3:**

**{**

**fprintf(wp, "%02X%1X%03X", addr[l\_cnt].op, addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**count += 3;**

**break;**

**}**

**case 4:**

**{**

**fprintf(wp, "%02X%1X%05X", addr[l\_cnt].op, addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**count += 4;**

**break;**

**}**

**default:**

**{**

**//OPERATOR가 byte일 경우**

**if (!strcmp(table[i]->operate, "BYTE"))**

**{**

**if (table[i]->operand[0][0] == 'C')**

**{**

**fprintf(wp, "%02X%02X%02X", addr[l\_cnt].byte[0], addr[l\_cnt].byte[1], addr[l\_cnt].byte[2]);**

**count += 3;**

**}**

**else if (table[i]->operand[0][0] == 'X')**

**{**

**fprintf(wp, "%s", addr[l\_cnt].byte);**

**count += 1;**

**}**

**}**

**//OPERATOR가 word일 경우**

**else if (!strcmp(table[i]->operate, "WORD"))**

**{**

**fprintf(wp, "%06X", addr[l\_cnt].loc);**

**count += 3;**

**}**

**//LTORG이 나올경우**

**else if (!strcmp(table[i]->operate, "LTORG"))**

**{**

**enter = 1;**

**int lk = 0;**

**for (int j = 0; j < r\_lit\_cnt; j++)**

**{**

**if (rr\_cnt == lit\_table[lk].sub\_r\_cnt)**

**{**

**l\_cnt++;**

**t\_cnt++;**

**l\_cnt2 = l\_cnt;**

**if (lit\_table[lk].symbol[1] == 'C')**

**{**

**//fprintf(wp, "%02X%02X%02X", addr[l\_cnt].byte[0], addr[l\_cnt].byte[1], addr[l\_cnt].byte[2]);**

**count += 3;**

**lk++;**

**}**

**else if (lit\_table[lk].symbol[1] == 'X')**

**{**

**//fprintf(wp, "%02X", addr[l\_cnt].byte);**

**count += 1;**

**lk++;**

**}**

**}**

**else**

**{**

**lk++;**

**continue;**

**}**

**}**

**}**

**break;**

**}**

**}**

**//LTORG가 나올경우 T 레코드를 개행해줌**

**if (enter == 1)**

**{**

**enter = 0;**

**count = 0;**

**fputc('\n', wp);**

**for (int c = l\_cnt2; c < l\_cnt2 + 12; c++)**

**{**

**if (real\_cnt + 3 > max || addr[c].r\_cnt != rr\_cnt)**

**{**

**break;**

**}**

**else**

**{**

**real\_cnt += addr[c].format;**

**}**

**}**

**fputc('T', wp);**

**fprintf(wp, "%06X%02X", LOCCTR[l\_cnt2], real\_cnt);**

**real\_cnt = 0;**

**int lk = 0;**

**for (int j = 0; j < r\_lit\_cnt; j++)**

**{**

**if (rr\_cnt == lit\_table[lk].sub\_r\_cnt)**

**{**

**if (lit\_table[lk].symbol[1] == 'C')**

**{**

**fprintf(wp, "%02X%02X%02X", addr[l\_cnt2].byte[0], addr[l\_cnt2].byte[1], addr[l\_cnt2].byte[2]);**

**count += 3;**

**lk++;**

**}**

**else if (lit\_table[lk].symbol[1] == 'X')**

**{**

**fprintf(wp, "%s", addr[l\_cnt2].byte);**

**count += 1;**

**lk++;**

**}**

**}**

**else**

**{**

**break;**

**}**

**}**

**}**

**//T레코드의 바이트 수가 30을 넘을경우**

**else**

**{**

**if (count + 3 > max)**

**{**

**l\_cnt2 = l\_cnt;**

**count = 0;**

**fputc('\n', wp);**

**}**

**}**

**//프로그램의 마지막**

**if (!strcmp(table[i]->operate, "END"))**

**{**

**int lk = 0;**

**for (int j = 0; j <= r\_lit\_cnt; j++)**

**{**

**if (rr\_cnt == lit\_table[j].sub\_r\_cnt)**

**{**

**l\_cnt++;**

**if (lit\_table[j].symbol[1] == 'C')**

**{**

**fprintf(wp, "%02X%02X%02X", addr[l\_cnt].byte[0], addr[l\_cnt].byte[1], addr[l\_cnt].byte[2]);**

**count += 3;**

**lk++;**

**}**

**else if (lit\_table[j].symbol[1] == 'X')**

**{**

**fprintf(wp, "%s", addr[l\_cnt].byte);**

**count += 1;**

**lk++;**

**}**

**}**

**else**

**{**

**continue;**

**}**

**}**

**if (m\_cnt != 0)**

**{**

**fputc('\n', wp);**

**for (int m = 0; m < m\_cnt; m++)**

**{**

**if (is\_word[m] == 1)**

**{**

**fprintf(wp, "M%06X06+%s\n", m\_record[m], m\_str[m]);**

**}**

**else**

**{**

**fprintf(wp, "M%06X05+%s\n", m\_record[m], m\_str[m]);**

**}**

**}**

**}**

**fprintf(wp, "E");**

**return;**

**}**

**}**

**}**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : Object Code를 저장하는 함수**

**\* 매계 : 없음**

**\* 반환 : 없음**

**\* -----------------------------------------------------------------------------------**

**\*/**

**void input\_address()**

**{**

**int l\_cnt = 0;**

**int sub\_cnt = 0;**

**int bbreak = 0;**

**for (int i = 0; i < line\_num; i++)**

**{**

**int k = 0;**

**search\_opcode(table[i]->operate);**

**addr[l\_cnt].r\_cnt = sub\_cnt;**

**//프로그램이 끝나기 전 리터럴 폴이 있을 경우 object code를 저장**

**if (!strcmp(table[i]->operate, "END"))**

**{**

**l\_cnt++;**

**addr[l\_cnt].r\_cnt = sub\_cnt;**

**for (int t = 0; t <= r\_lit\_cnt; t++)**

**{**

**if (sub\_cnt == lit\_table[t].sub\_r\_cnt)**

**{**

**if (lit\_table[t].symbol[1] == 'C')**

**{**

**char space[1] = " ";**

**for (int j = 1; j < strlen(lit\_table[t].symbol); j++)**

**{**

**\*space = lit\_table[t].symbol[j];**

**if (lit\_table[t].symbol[j] != 39 && lit\_table[t].symbol[j] != 'C'&& lit\_table[t].symbol[j] != '=')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 3;**

**}**

**}**

**}**

**else if (lit\_table[t].symbol[1] == 'X')**

**{**

**char space[1] = " ";**

**for (int j = 1; j < strlen(lit\_table[t].symbol); j++)**

**{**

**\*space = lit\_table[t].symbol[j];**

**if (lit\_table[t].symbol[j] != 39 && lit\_table[t].symbol[j] != 'X'&& lit\_table[t].symbol[j] != '=')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 1;**

**}**

**}**

**}**

**l\_cnt++;**

**}**

**}**

**return;**

**}**

**if(!strcmp(table[i]->operate, "START") || !strcmp(table[i]->operate, "CSECT"))**

**{**

**sub\_cnt++;**

**l\_cnt++;**

**continue;**

**}**

**if (!strcmp(table[i]->operate, "RESW") || !strcmp(table[i]->operate, "RESB") || !strcmp(table[i]->operate, "EQU") ||**

**!strcmp(table[i]->operate, "EXTDEF") || !strcmp(table[i]->operate, "EXTREF") )**

**{**

**l\_cnt++;**

**continue;**

**}**

**k = search\_opcode(table[i]->operate); //inst 목록에서 operator을 검사**

**if (k < 0) //inst 목록에 없는경우**

**{**

**if (!strcmp(table[i]->operate, "WORD")) //WORD인 경우**

**{**

**addr[l\_cnt].format = 3;**

**int calt = search\_cal(table[i]->operand[3]); //operand에 연산이 있는지 검사**

**switch (calt)**

**{**

**//연산이 없는 경우**

**case 0:**

**{**

**for (int t = 0; t < 3; t++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[sub\_cnt].refer[t])) //참조값인지 확인**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**}**

**if (bbreak == 1)**

**{**

**bbreak = 0;**

**break;**

**}**

**addr[l\_cnt].loc = atoi(table[i]->operand[0]);**

**for (int t = 0; t < sym\_cnt; t++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[t].addr;**

**}**

**}**

**}**

**l\_cnt++;**

**break;**

**}**

**//연산이 1개인 경우**

**case 1:**

**{**

**int op1 = 0, op2 = 0, sum = 0;**

**for (int t = 0; t < 3; t++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[sub\_cnt].refer[t]))**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**if (!strcmp(table[i]->operand[1], refer\_table[sub\_cnt].refer[t]))**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**}**

**if (bbreak == 1)**

**{**

**bbreak = 0;**

**break;**

**}**

**for (int t = 0; t < sym\_cnt; t++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**op1 = sym\_table[t].addr;**

**}**

**}**

**if (!strcmp(table[i]->operand[1], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**op2 = sym\_table[t].addr;**

**}**

**}**

**}**

**if (op1 == 0)**

**{**

**op1 = atoi(table[i]->operand[0]);**

**}**

**if (op2 == 0)**

**{**

**op2 = atoi(table[i]->operand[1]);**

**}**

**if (cal[0][0] == '+')**

**{**

**sum = op1 + op2;**

**}**

**else if (cal[0][0] == '-')**

**{**

**sum = op1 - op2;**

**}**

**addr[l\_cnt++].loc = sum;**

**break;**

**}**

**//연산이 2개인 경우**

**case 2:**

**{**

**int op3=0, op1 = 0, op2 = 0, sum = 0;**

**//참조값인지 검사**

**for (int t = 0; t < 3; t++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[sub\_cnt].refer[t]))**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**if (!strcmp(table[i]->operand[1], refer\_table[sub\_cnt].refer[t]))**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**if (!strcmp(table[i]->operand[2], refer\_table[sub\_cnt].refer[t]))**

**{**

**addr[l\_cnt].is\_r = 1;**

**addr[l\_cnt++].loc = 0;**

**bbreak = 1;**

**break;**

**}**

**}**

**if (bbreak == 1)**

**{**

**bbreak = 0;**

**break;**

**}**

**for (int t = 0; t < sym\_cnt; t++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**op1 = sym\_table[t].addr;**

**}**

**}**

**if (!strcmp(table[i]->operand[1], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**op2 = sym\_table[t].addr;**

**}**

**}**

**if (!strcmp(table[i]->operand[2], sym\_table[t].symbol))**

**{**

**if (sub\_cnt == sym\_table[t].r\_cnt)**

**{**

**op1 = sym\_table[t].addr;**

**}**

**}**

**}**

**if (op1 == 0)**

**{**

**op1 = atoi(table[i]->operand[0]);**

**}**

**if (op2 == 0)**

**{**

**op2 = atoi(table[i]->operand[1]);**

**}**

**if (op3 == 0)**

**{**

**op3 = atoi(table[i]->operand[1]);**

**}**

**if (cal[0][0] == '+')**

**{**

**sum = op1 + op2;**

**}**

**else if (cal[0][0] == '-')**

**{**

**sum = op1 - op2;**

**}**

**if (cal[1][0] == '+')**

**{**

**sum += op3;**

**}**

**else if (cal[1][0] == '-')**

**{**

**sum -= op3;**

**}**

**addr[l\_cnt++].loc = sum;**

**break;**

**}**

**default :**

**break;**

**}**

**continue;**

**}**

**//BYTE인 경우**

**if (!strcmp(table[i]->operate, "BYTE"))**

**{**

**if (table[i]->operand[0][0] == 'C')**

**{**

**char space[1] = " ";**

**for (int j = 0; j < strlen(table[i]->operand[0]); j++)**

**{**

**\*space = \*(table[i]->operand[0] + j);**

**if (table[i]->operand[0][j] != 39 && table[i]->operand[0][j] != 'C')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 3;**

**}**

**}**

**}**

**else if (table[i]->operand[0][0] == 'X')**

**{**

**char space[1] = " ";**

**for (int j = 0; j < strlen(table[i]->operand[0]); j++)**

**{**

**\*space = \*(table[i]->operand[0] + j);**

**if (table[i]->operand[0][j] != 39 && table[i]->operand[0][j] != 'X')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 1;**

**}**

**}**

**}**

**l\_cnt++;**

**continue;**

**}**

**//LTORG가 나올 경우 리터럴 폴 비우기**

**if (!strcmp(table[i]->operate, "LTORG"))**

**{**

**l\_cnt++;**

**addr[l\_cnt].r\_cnt = sub\_cnt;**

**addr[l\_cnt].is\_r = 2;**

**for (int t = 0; t <= r\_lit\_cnt; t++)**

**{**

**if (sub\_cnt==lit\_table[t].sub\_r\_cnt)**

**{**

**if (lit\_table[t].symbol[1] == 'C')**

**{**

**char space[1] = " ";**

**for (int j = 1; j < strlen(lit\_table[t].symbol); j++)**

**{**

**\*space = lit\_table[t].symbol[j];**

**if (lit\_table[t].symbol[j] != 39 && lit\_table[t].symbol[j] != 'C'&& lit\_table[t].symbol[j] != '=')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 3;**

**}**

**}**

**}**

**else if (lit\_table[t].symbol[1] == 'X')**

**{**

**char space[1] = " ";**

**for (int j = 1; j < strlen(lit\_table[t].symbol); j++)**

**{**

**\*space = lit\_table[t].symbol[j];**

**if (lit\_table[t].symbol[j] != 39 && lit\_table[t].symbol[j] != 'X'&& lit\_table[t].symbol[j] != '=')**

**{**

**strncat(addr[l\_cnt].byte, space, 1);**

**addr[l\_cnt].format = 1;**

**}**

**}**

**}**

**l\_cnt++;**

**}**

**}**

**continue;**

**}**

**}**

**//inst에 있는경우**

**else**

**{**

**switch (instruction[k].format)**

**{**

**case 2: //2형식일 경우**

**{**

**if (!strcmp(table[i]->operand[1], "\0")) //operand가 1개일 경우**

**{**

**int r = search\_register(table[i]->operand[0]); //무슨 레지스터인지 검사**

**addr[l\_cnt].loc = r \* 10;**

**addr[l\_cnt].format = 2;**

**sscanf(table[i]->op, "%x", &addr[l\_cnt].op);**

**l\_cnt++;**

**break;**

**}**

**else**

**{**

**int r1= search\_register(table[i]->operand[0]); //무슨 레지스터인지 검사**

**int r2 = search\_register(table[i]->operand[1]); //무슨 레지스터인지 검사**

**addr[l\_cnt].loc = r1 \* 10;**

**addr[l\_cnt].loc += r2;**

**addr[l\_cnt].format = 2;**

**sscanf(table[i]->op, "%x", &addr[l\_cnt].op);**

**l\_cnt++;**

**break;**

**}**

**}**

**case 3: //3형식일 경우**

**{**

**int integer = 0;**

**//RSUB인 경우**

**if (!strcmp(table[i]->operate, "RSUB"))**

**{**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].xbpe = 0;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**//리터럴 값을 사용한 경우**

**if (table[i]->operand[0][0] == '=')**

**{**

**for (int t = 0; t <= r\_lit\_cnt; t++)**

**{**

**if (!strcmp(table[i]->operand[0], lit\_table[t].symbol))**

**{**

**addr[l\_cnt].loc = lit\_table[t].addr - LOCCTR[l\_cnt + 1];**

**if (addr[l\_cnt].loc < 0)**

**{**

**addr[l\_cnt].loc = addr[l\_cnt].loc & 0xFFF;**

**}**

**break;**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 2;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**//Immediate인 경우**

**if (table[i]->operand[0][0] == '#')**

**{**

**char \*im;**

**im = strtok(table[i]->operand[0], "#");**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].loc = atoi(im);**

**addr[l\_cnt].op += integer + 1;**

**addr[l\_cnt].xbpe = 0;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**//indirect인 경우**

**if (table[i]->operand[0][0] == '@')**

**{**

**char \*ch;**

**ch = strtok(table[i]->operand[0], "@");**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr - LOCCTR[l\_cnt + 1];**

**if (addr[l\_cnt].loc < 0)**

**{**

**addr[l\_cnt].loc = addr[l\_cnt].loc & 0xFFF;**

**}**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**//break;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 2;**

**addr[l\_cnt].xbpe = 2;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**if (!strcmp(table[i]->operand[1], "\0")) //operand가 1개일 경우**

**{**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr - LOCCTR[l\_cnt + 1];**

**if (addr[l\_cnt].loc < 0)**

**{**

**addr[l\_cnt].loc = addr[l\_cnt].loc & 0xFFF;**

**}**

**addr[l\_cnt].is\_r = 0;**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 2;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**else**

**{**

**//X 레지스터를 사용한 경우**

**if (table[i]->operand[1][0] == 'X')**

**{**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr - LOCCTR[l\_cnt + 1];**

**if (addr[l\_cnt].loc < 0)**

**{**

**addr[l\_cnt].loc = addr[l\_cnt].loc & 0xFFF;**

**}**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**//break;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 10;**

**addr[l\_cnt].format = 3;**

**l\_cnt++;**

**break;**

**}**

**}**

**}**

**case 4: //4형식인 경우**

**{**

**int integer = 0;**

**//리터럴 값을 사용한 경우**

**if (table[i]->operand[0][0] == '=')**

**{**

**for (int t = 0; t <= r\_lit\_cnt; t++)**

**{**

**if (!strcmp(table[i]->operand[0], lit\_table[t].symbol))**

**{**

**addr[l\_cnt].loc = lit\_table[t].addr;**

**break;**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 1;**

**addr[l\_cnt].format = 4;**

**l\_cnt++;**

**break;**

**}**

**//Immediate인 경우**

**if (table[i]->operand[0][0] == '#')**

**{**

**char im[10];**

**strcpy(im, table[i]->operand[0]);**

**im[0] = " ";**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].loc = atoi(im);**

**addr[l\_cnt].op += integer + 1;**

**addr[l\_cnt].xbpe = 1;**

**addr[l\_cnt].format = 4;**

**l\_cnt++;**

**break;**

**}**

**//indirect인 경우**

**if (table[i]->operand[0][0] == '@')**

**{**

**char \*ch;**

**ch = strtok(table[i]->operand[0], "@");**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr;**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**//break;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 2;**

**addr[l\_cnt].xbpe = 1;**

**addr[l\_cnt].format = 4;**

**l\_cnt++;**

**break;**

**}**

**if (!strcmp(table[i]->operand[1], "\0")) //operand가 1개일 경우**

**{**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr;**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**//break;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 1;**

**addr[l\_cnt].format = 4;**

**l\_cnt++;**

**break;**

**}**

**else**

**{**

**//X 레지스터를 사용한 경우**

**if (table[i]->operand[1][0] == 'X')**

**{**

**for (int j = 0; j< sym\_cnt; j++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[j].symbol))**

**{**

**if (sym\_table[j].r\_cnt == sub\_cnt)**

**{**

**addr[l\_cnt].loc = sym\_table[j].addr;**

**break;**

**}**

**else**

**{**

**addr[l\_cnt].loc = 0;**

**addr[l\_cnt].is\_r = 1;**

**//break;**

**}**

**}**

**}**

**sscanf(table[i]->op, "%x", &integer);**

**addr[l\_cnt].op += integer + 3;**

**addr[l\_cnt].xbpe = 9;**

**addr[l\_cnt].format = 4;**

**l\_cnt++;**

**break;**

**}**

**}**

**}**

**default:**

**break;**

**}**

**continue;**

**}**

**}**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : SYMTAB을 저장하는 함수**

**\* 매계 : 없음**

**\* 반환 : 없음**

**\* -----------------------------------------------------------------------------------**

**\*/**

**void input\_sym()**

**{**

**sym\_cnt = 0;**

**int lit\_cnt = 0;**

**r\_lit\_cnt = 0;**

**refer\_cnt = 0;**

**r\_cnt = 0;**

**int k = 0;**

**int loc = 0;**

**int num = 0;**

**int iscal = 0;**

**char literals[30][10];**

**char space[1] = "";**

**loc\_cnt = 0;**

**for (int i = 0; i < line\_num; i++)**

**{**

**if (!strcmp(table[i]->operate, "START") || !strcmp(table[i]->operate, "CSECT"))**

**{**

**if (!strcmp(table[i]->operate, "CSECT"))**

**{**

**int j = i - 1;**

**for (; j > 0; j--)**

**{**

**if (!strcmp(table[j]->operate, "EQU"))**

**{**

**continue;**

**}**

**else**

**{**

**break;**

**}**

**}**

**j += r\_lit\_cnt;**

**end[r\_cnt] = LOCCTR[j + 1]; //PASS2에서 쓰일 프로그램의 길이를 저장**

**for (int jk = 0; jk < 5; jk++)**

**{**

**int jjk = search\_opcode(table[i + jk + 1]->operate);**

**if (jjk >= 0)**

**{**

**strcpy(table[i + jk + 1]->label, table[i]->label);**

**break;**

**}**

**}**

**}**

**loc = 0;**

**LOCCTR[loc\_cnt] = loc;**

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

**for (int h = 0; h <= lit\_cnt; h++)**

**{**

**strcpy(literals[h], space);**

**}**

**lit\_cnt = 0;**

**r\_cnt++;**

**continue;**

**}**

**sym\_table[sym\_cnt].r\_cnt = r\_cnt;**

**//리터럴 값이 나왔을 경우 literals배열에 저장**

**if (table[i]->operand[0][0] == '=')**

**{**

**if (strcmp(table[i]->operand[0], literals[lit\_cnt - 1]))**

**{**

**strcpy(literals[lit\_cnt], table[i]->operand[0]);**

**lit\_cnt++;**

**}**

**}**

**//EXTDEF인 경우**

**if (!strcmp(table[i]->operate, "EXTDEF"))**

**{**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**continue;**

**}**

**//EXTREF인 경우**

**if (!strcmp(table[i]->operate, "EXTREF"))**

**{**

**int r = 0;**

**for (r = 0; r < 3; r++)**

**{**

**if (strcmp(table[i]->operand[r], ""))**

**{**

**strcpy(refer\_table[r\_cnt].refer[r], table[i]->operand[r]);**

**refer\_cnt++;**

**}**

**else**

**{**

**break;**

**}**

**}**

**refer\_table[r\_cnt].cnt = r + 1;**

**//refer\_table[r\_cnt].sub\_cnt = r\_cnt;**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**continue;**

**}**

**k = search\_opcode(table[i]->operate); //INST테이블의 명령어 인지 검사**

**if (k < 0) //명령어가 아닌 경우**

**{**

**//LTORG가 나왔을 경우 리터럴 폴 비우고 리터럴 테이블에 저장**

**if (!strcmp(table[i]->operate, "LTORG"))**

**{**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**for (int p = 0; p < lit\_cnt; p++)**

**{**

**strcpy(lit\_table[r\_lit\_cnt].symbol, literals[p]);**

**lit\_table[r\_lit\_cnt].addr = loc;**

**lit\_table[r\_lit\_cnt].sub\_r\_cnt = r\_cnt;**

**LOCCTR[loc\_cnt] = loc;**

**if (lit\_table[r\_lit\_cnt].symbol[1] == 'X')**

**{**

**loc += 1;**

**}**

**else**

**{**

**loc += 3;**

**}**

**loc\_cnt++;**

**LOCCTR[loc\_cnt] = loc;**

**r\_lit\_cnt++;**

**}**

**}**

**//RESW인 경우**

**if (!strcmp(table[i]->operate, "RESW"))**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**num = 3 \* atoi(table[i]->operand[3]);**

**loc += num;**

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

**sym\_cnt++;**

**continue;**

**}**

**//RESB인 경우**

**if (!strcmp(table[i]->operate, "RESB"))**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**num = atoi(table[i]->operand[3]);**

**loc += num;**

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

**sym\_cnt++;**

**continue;**

**}**

**//EQU인 경우**

**if (!strcmp(table[i]->operate, "EQU"))**

**{**

**//현재 주소값을 사용하는 경우**

**if (!strcmp(table[i]->operand[3], "\*"))**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

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

**continue;**

**}**

**else**

**{**

**int op1 = 0, op2 = 0, op3 = 0, sum = 0;**

**iscal = search\_cal(table[i]->operand[3]); //OPERAND가 연산이 있는지 검사**

**switch(iscal)**

**{**

**case 2: //연산이 2개인 경우**

**{**

**for (int s = 0; s < sym\_cnt; s++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[s].symbol))**

**{**

**op1 = sym\_table[s].addr;**

**for (int r = 0; r <= 3; r++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[r\_cnt].refer[r]))**

**{**

**op1 = -1;**

**}**

**}**

**}**

**if (!strcmp(table[i]->operand[1], sym\_table[s].symbol))**

**{**

**op2 = sym\_table[s].addr;**

**}**

**if (!strcmp(table[i]->operand[2], sym\_table[s].symbol))**

**{**

**op3 = sym\_table[s].addr;**

**}**

**}**

**if (op1 == 0)**

**{**

**op1 = atoi(table[i]->operand[0]);**

**}**

**else if (op1 == -1)**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

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

**break;**

**}**

**if (op2 == 0)**

**{**

**op2 = atoi(table[i]->operand[1]);**

**}**

**if (op3 == 0)**

**{**

**op3 = atoi(table[i]->operand[1]);**

**}**

**if (cal[0][0] == '+')**

**{**

**sum = op1 + op2;**

**}**

**else if (cal[0][0] == '-')**

**{**

**sum = op1 - op2;**

**}**

**if (cal[1][0] == '+')**

**{**

**sum += op3;**

**}**

**else if (cal[1][0] == '-')**

**{**

**sum -= op3;**

**}**

**sym\_table[sym\_cnt].addr = sum;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

**LOCCTR[loc\_cnt] = sum;**

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

**break;**

**}**

**case 1: //연산이 1개인 경우**

**{**

**op1 = 0, op2 = 0, op3 = 0, sum = 0;**

**for (int s = 0; s < sym\_cnt; s++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[s].symbol))**

**{**

**op1 = sym\_table[s].addr;**

**for (int r = 0; r <= 3; r++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[r\_cnt].refer[r]))**

**{**

**op1 = -1;**

**}**

**}**

**}**

**if (!strcmp(table[i]->operand[1], sym\_table[s].symbol))**

**{**

**op2 = sym\_table[s].addr;**

**}**

**}**

**if (op1 == 0)**

**{**

**op1 = atoi(table[i]->operand[0]);**

**}**

**else if (op1 == -1)**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

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

**break;**

**}**

**if (op2 == 0)**

**{**

**op2 = atoi(table[i]->operand[1]);**

**}**

**if (cal[0][0] == '+')**

**{**

**sum = op1 + op2;**

**}**

**else if (cal[0][0] == '-')**

**{**

**sum = op1 - op2;**

**}**

**sym\_table[sym\_cnt].addr = sum;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

**LOCCTR[loc\_cnt] = sum;**

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

**break;**

**}**

**case 0: //연산이 없는 경우**

**{**

**op1 = 0;**

**for (int s = 0; s < sym\_cnt; s++)**

**{**

**if (!strcmp(table[i]->operand[0], sym\_table[s].symbol))**

**{**

**op1 = sym\_table[s].addr;**

**for (int r = 0; r <= 3; r++)**

**{**

**if (!strcmp(table[i]->operand[0], refer\_table[r\_cnt].refer[r]))**

**{**

**op1 = -1;**

**}**

**}**

**}**

**}**

**if (op1 == 0)**

**{**

**op1 = atoi(table[i]->operand[0]);**

**sym\_table[sym\_cnt].addr = op1;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

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

**break;**

**}**

**else if (op1 == -1)**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

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

**break;**

**}**

**else**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**sym\_cnt++;**

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

**break;**

**}**

**}**

**default:**

**break;**

**}**

**}**

**}**

**//BYTE인 경우**

**if (!strcmp(table[i]->operate, "BYTE"))**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**loc += 1;**

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

**sym\_cnt++;**

**continue;**

**}**

**//WORD인 경우**

**if (!strcmp(table[i]->operate, "WORD"))**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**loc += 3;**

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

**sym\_cnt++;**

**continue;**

**}**

**}**

**else //명령어인 경우**

**{**

**switch (instruction[k].format)**

**{**

**case 2: //2형식인 경우**

**{**

**if (!strcmp(table[i]->label,"") || !strcmp(table[i]->label,"\t"))**

**{**

**loc += 2;**

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

**}**

**else**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**loc += 2;**

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

**sym\_cnt++;**

**}**

**break;**

**}**

**case 3: //3형식인 경우**

**{**

**if (!strcmp(table[i]->label, "") || !strcmp(table[i]->label, "\t"))**

**{**

**loc += 3;**

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

**}**

**else**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**loc += 3;**

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

**sym\_cnt++;**

**}**

**break;**

**}**

**case 4: //4형식인 경우**

**{**

**if (!strcmp(table[i]->label, "") || !strcmp(table[i]->label, "\t"))**

**{**

**loc += 4;**

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

**}**

**else**

**{**

**sym\_table[sym\_cnt].addr = loc;**

**strcpy(sym\_table[sym\_cnt].symbol, table[i]->label);**

**loc += 4;**

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

**sym\_cnt++;**

**}**

**break;**

**}**

**default :**

**break;**

**}**

**}**

**//프로그램의 마지막인 경우**

**if (!strcmp(table[i]->operate, "END"))**

**{**

**int j = i - 1;**

**if (lit\_cnt == 0)**

**{**

**for (; j > 0; j--)**

**{**

**if (!strcmp(table[j]->operate, "EQU"))**

**{**

**continue;**

**}**

**else**

**{**

**break;**

**}**

**}**

**end[r\_cnt] = LOCCTR[j + 1]; //마지막 프로그램의 길이 저장**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**return;**

**}**

**//리터럴 폴에 값이 있는경우 리터럴 폴 비우고 리터럴테이블에 저장**

**else**

**{**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**for (int p = 0; p < lit\_cnt; p++)**

**{**

**strcpy(lit\_table[r\_lit\_cnt].symbol, literals[p]);**

**lit\_table[r\_lit\_cnt].addr = loc;**

**lit\_table[r\_lit\_cnt].sub\_r\_cnt = r\_cnt;**

**LOCCTR[loc\_cnt] = loc;**

**if (lit\_table[r\_lit\_cnt].symbol[1] == 'X')**

**{**

**loc += 1;**

**}**

**else**

**{**

**loc += 3;**

**}**

**loc\_cnt++;**

**LOCCTR[loc\_cnt] = loc;**

**end[r\_cnt] = loc;**

**r\_lit\_cnt++;**

**}**

**}**

**r\_lit\_cnt--;**

**LOCCTR[loc\_cnt] = -1;**

**loc\_cnt++;**

**return;**

**}**

**}**

**}**

**/\* ----------------------------------------------------------------------------------**

**\* 설명 : Immediate data를 화면에 출력하는 함수**

**\* 매계 : 없음**

**\* 반환 : 없음**

**\* -----------------------------------------------------------------------------------**

**\*/**

**void data\_output()**

**{**

**int f = 0;**

**int k = 0;**

**int l\_cnt = 0;**

**int rr\_cnt = 0;**

**for (int i = 0; i < loc\_cnt; i++)**

**{**

**search\_opcode(table[i]->operate);**

**//섹션루틴이나 프로그램의 시작인 경우 loc값만 출력**

**if (!strcmp(table[i]->operate, "START") || !strcmp(table[i]->operate, "CSECT"))**

**{**

**printf("%04X\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate);**

**rr\_cnt++;**

**}**

**//LTORG가 나왔을 경우 리터럴 폴안의 값들의 object code와 함께 출력**

**else if (!strcmp(table[i]->operate, "LTORG"))**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\n", table[i]->label, table[i]->operate);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate);**

**}**

**for (int j = 0; j < r\_lit\_cnt; j++)**

**{**

**if (rr\_cnt == lit\_table[k].sub\_r\_cnt)**

**{**

**l\_cnt++;**

**if (lit\_table[k].symbol[1] == 'C')**

**{**

**printf("%04X\t%s\t\%s\t\t%02X%02X%02X\n", LOCCTR[l\_cnt], "\*", lit\_table[k].symbol,addr[l\_cnt].byte[0],**

**addr[l\_cnt].byte[1], addr[l\_cnt].byte[2] );**

**k++;**

**}**

**else if (lit\_table[k].symbol[1] == 'X')**

**{**

**printf("%04X\t%s\t\%s\t\t%s\n", LOCCTR[l\_cnt], "\*", lit\_table[k].symbol, addr[l\_cnt].byte);**

**k++;**

**}**

**}**

**else**

**{**

**break;**

**}**

**}**

**}**

**//프로그램의 끝과 리터럴 값 출력**

**else if (!strcmp(table[i]->operate, "END"))**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\t%s\n", table[i]->label, table[i]->operate,table[i]->operand[3]);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate);**

**}**

**for (int j = 0; j < r\_lit\_cnt; j++)**

**{**

**if (rr\_cnt == lit\_table[k].sub\_r\_cnt)**

**{**

**l\_cnt++;**

**if (lit\_table[k].symbol[1] == 'C')**

**{**

**printf("%04X\t%s\t\%s\t\t%02X%02X%02X\n", LOCCTR[l\_cnt], "\*", lit\_table[k].symbol, addr[l\_cnt].byte[0],**

**addr[l\_cnt].byte[1], addr[l\_cnt].byte[2]);**

**k++;**

**}**

**else if (lit\_table[k].symbol[1] == 'X')**

**{**

**printf("%04X\t%s\t\%s\t\t%s\n", LOCCTR[l\_cnt], "\*", lit\_table[k].symbol, addr[l\_cnt].byte);**

**k++;**

**}**

**}**

**else**

**{**

**break;**

**}**

**}**

**return;**

**}**

**//operator가 BYTE인 경우**

**else if (!strcmp(table[i]->operate, "BYTE"))**

**{**

**if (table[i]->operand[0][0] == 'C')**

**{**

**printf("%04X\t%s\t\%s\t%s\t%02X%02X%02X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[0],**

**addr[l\_cnt].byte[0],addr[l\_cnt].byte[1], addr[l\_cnt].byte[2]);**

**}**

**else if (table[i]->operand[0][0] == 'X')**

**{**

**printf("%04X\t%s\t\%s\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate,table[i]->operand[0],**

**addr[l\_cnt].byte);**

**}**

**}**

**//operator가 WORD인 경우**

**else if (!strcmp(table[i]->operate, "WORD"))**

**{**

**printf("%04X\t%s\t%s\t%s\t%06X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].loc);**

**}**

**else**

**{**

**int jk = search\_opcode(table[i]->operate); //명령어 인지 검사**

**f = instruction[jk].format; //명령어의 format값**

**switch (f)**

**{**

**case 2: //2형식일 경우**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\t%s\n", table[i]->label, table[i]->operate, table[i]->operand[3]);**

**}**

**else**

**{**

**printf("%04X%s\t%s\t%s\t%2X%02d\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op, addr[l\_cnt].loc);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\t%s\t%2X%02d\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op, addr[l\_cnt].loc);**

**}**

**break;**

**}**

**case 3: //3형식일 경우**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\t%s\n", table[i]->label, table[i]->operate, table[i]->operand[3]);**

**}**

**else**

**{**

**printf("%04X%s\t%s\t%s\t%02X%1X%03X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op,addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\t%s\t%02X%1X%03X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op,addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**}**

**break;**

**}**

**case 4: //4형식일 경우**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\t%s\n", table[i]->label, table[i]->operate, table[i]->operand[3]);**

**}**

**else**

**{**

**printf("%04X%s\t%s\t%s\t%02X%1X%05X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op, addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\t%s\t%02X%1X%05X\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3],**

**addr[l\_cnt].op, addr[l\_cnt].xbpe, addr[l\_cnt].loc);**

**}**

**break;**

**}**

**//명령어가 아닌 경우**

**default:**

**{**

**if (!strcmp(table[i]->label, "\t"))**

**{**

**if (LOCCTR[l\_cnt] == -1)**

**{**

**printf("%s\t%s\t%s\n", table[i]->label, table[i]->operate, table[i]->operand[3]);**

**}**

**else**

**{**

**printf("%04X%s\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3]**

**);**

**}**

**}**

**else**

**{**

**printf("%04X\t%s\t%s\t%s\n", LOCCTR[l\_cnt], table[i]->label, table[i]->operate, table[i]->operand[3]**

**);**

**}**

**break;**

**}**

**}**

**}**

**l\_cnt++;**

**}**

**}**

**//인자로 받은 char형 배열에 연산이 있는지 검색하는 함수 있을 경우 cal배열에 연산을 저장**

**//없을 경우 : 0 있을 경우 : 갯수**

**int search\_cal(char \*str)**

**{**

**int cal\_k = 0;**

**for (int i = 0; i < strlen(str); i++)**

**{**

**if (str[i] == '+')**

**{**

**cal[cal\_k++][0] = '+';**

**}**

**if (str[i] == '-')**

**{**

**cal[cal\_k++][0] = '-';**

**}**

**}**

**if (cal\_k == 0)**

**{**

**return 0;**

**}**

**else**

**{**

**return cal\_k;**

**}**

**}**

**//인자로 받은 char형 배열 레지스터가 있는 경우**

**//없을 경우 : 0 있을 경우 : 레지스터의 넘버**

**int search\_register(char \*str)**

**{**

**for (int i = 0; i < 9; i++)**

**{**

**if (!strcmp(reg[i].ch, str))**

**{**

**return reg[i].num;**

**}**

**}**

**return 0;**

**}**