과목 명: 시스템프로그래밍

담당 교수 명: 박 운 상

<<Project 3>>

**서강대학교 컴퓨터학과**

**[학번] 20161647**

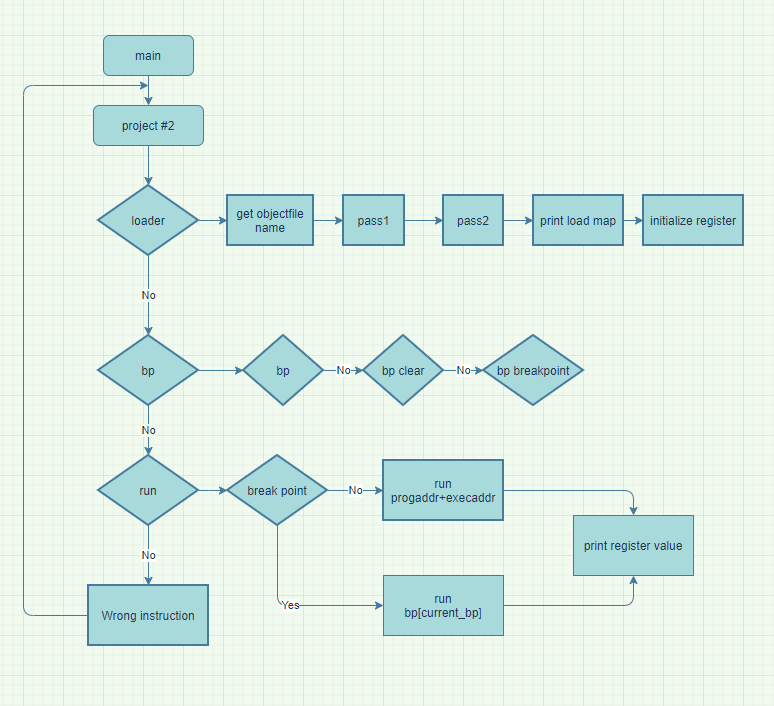
**[이름] 조이준**

목 차

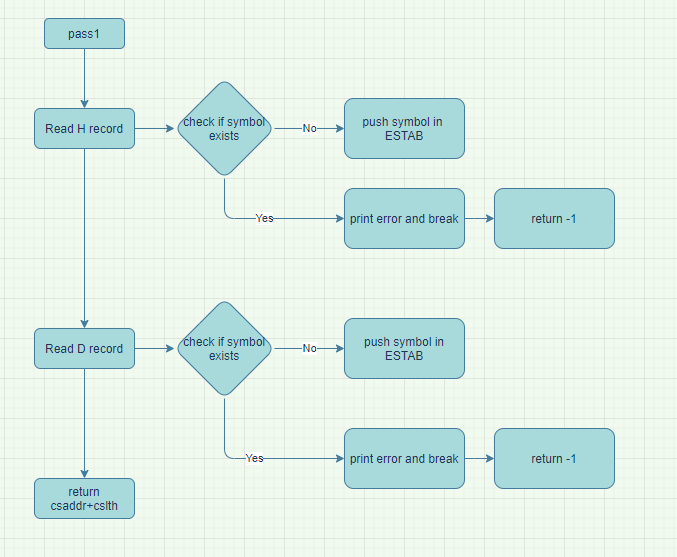
1. 프로그램 개요
2. 프로그램 설명
   1. 프로그램 흐름도
3. 모듈 정의
   1. 모듈 이름 main()
      1. 기능
      2. 사용 변수
   2. 모듈 이름 func\_ESTABPUSH()
      1. 기능
      2. 사용 변수
   3. 모듈 이름 func\_ESTABSEARCH()
      1. 기능
      2. 사용 변수
   4. 모듈 이름 func\_ESTABFREE()
      1. 기능
      2. 사용 변수
   5. 모듈 이름 func\_PRINTLOADMAP()
      1. 기능
      2. 사용 변수
   6. 모듈 이름 func\_CSECNUM()
      1. 기능
      2. 사용 변수
   7. 모듈 이름 func\_LOADpass1()
      1. 기능
      2. 사용 변수
   8. 모듈 이름 func\_LOADpass2()
      1. 기능
      2. 사용 변수
   9. 모듈 이름 func\_RegisterNum()
      1. 기능
      2. 사용 변수
   10. 모듈 이름 func\_breakpoint()
       1. 기능
       2. 사용 변수
   11. 모듈 이름 func\_initializeregister()
       1. 기능
       2. 사용 변수
   12. 모듈 이름 func\_run()
       1. 기능
       2. 사용 변수
   13. 모듈 이름 func\_getdataformat3()
       1. 기능
       2. 사용 변수
   14. 모듈 이름 func\_getdataformat4()
       1. 기능
       2. 사용 변수
   15. 모듈 이름 func\_regvalue()
       1. 기능
       2. 사용 변수
4. 전역 변수 정의
   1. progaddr
   2. execaddr
   3. program\_len
   4. ESTAB est[3]
   5. csec
   6. bp[20]
   7. bp\_len
   8. current\_bp
   9. is\_bp
   10. reg[10]
5. 코드
6. **프로그램 개요**

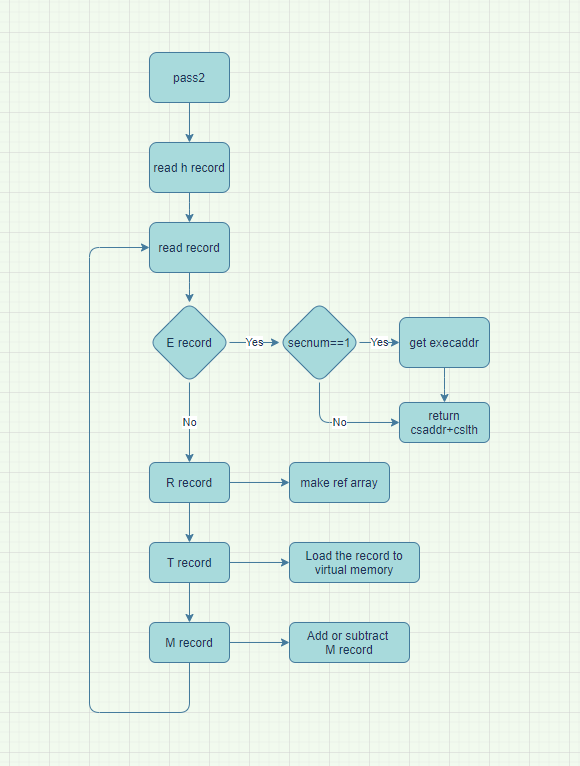
오브젝트 파일을 가상 메모리에 올리고 각 오브젝트 파일을 링크하는 링킹 로더를 구현하고 주어진 copy.obj 파일을 실행하는 run 명령어를 구현한다. 프로그램 내에 중단점을 설정하여 run 명령어 수행 시에 중단점에서 멈추는 기능 또한 구현한다.

1. **프로그램 설명**
   1. 프로그램 흐름도
      1. Main

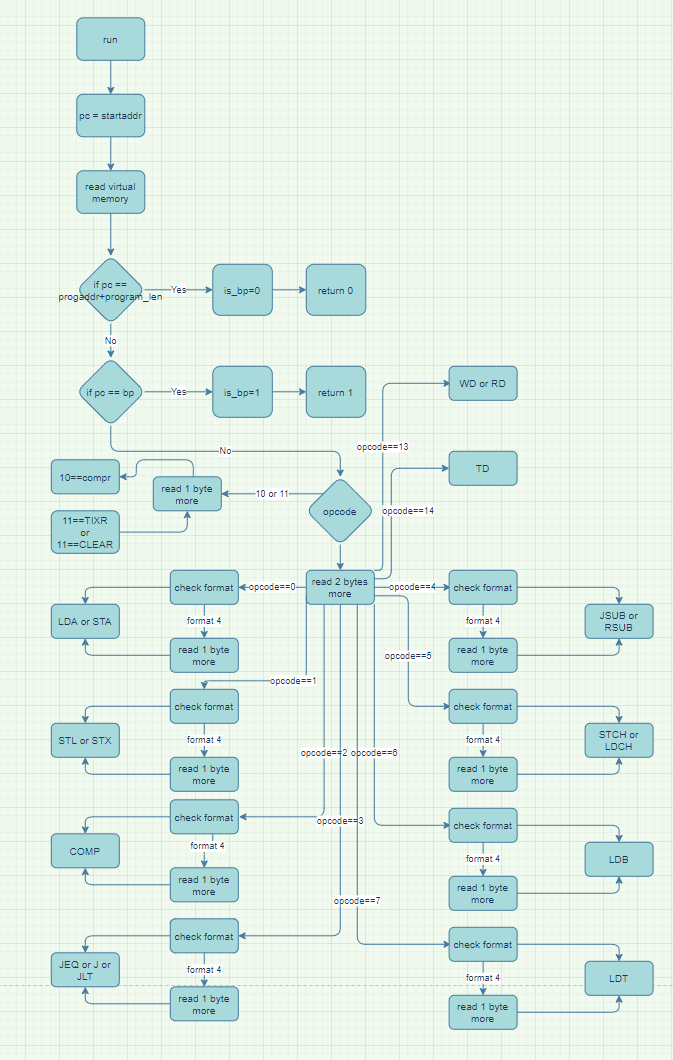


* + 1. Linking Loader





* + 1. Run



1. **모듈 정의**
   1. 모듈 이름 main()
      1. *기능*

Loader objectfile, run, bp, bp clear, bp breakpoint 명령어를 새로 추가하여 해당 명령어를 수행하는 함수를 호출한다

* + 1. *사용 변수*

섹션의 길이를 나타내는 cslth, 섹션의 번호를 나타내는 secnum을 추가했다.

* 1. 모듈 이름 func\_ESTABPUSH()
     1. *기능*

Linking Lodaer pass1 과정에서 읽은 EXTDEF와 Section name의 심볼과 location 등을 저장하는 ESTAB에 원소를 추가하는 기능이다.

* + 1. *사용 변수*

ESTAB\*tmp, ESTAB \* newnode를 지역변수로 선언했다. Tmp는 ESTAB에 접근하는 용도, newnode는 ESTAB에 추가할 새로운 노드이다.

인자로 추가할 심볼 명, 위치, 섹션 번호, 프로그램 길이를 받는다.

* 1. 모듈 이름 func\_ESTABSEARCH()
     1. *기능*

심볼을 인자로 받아 해당 심볼이 ESTAB에 존재하는 탐색한다. 존재하면 해당 심볼의 location값을 반환하고 존재하지 않으면 -1을 반환한다.

* + 1. *사용 변수*

ESTAB\*tmp ; 테이블의 원소에 접근하기 위한 포인터이다.

* 1. 모듈 이름 func\_ESTABFREE()
     1. *기능*

동적 할당한ESTAB의 내용을 해제한다.

* + 1. *사용 변수*

ESTAB\*tmp, \*free\_tmp ; 테이블의 노드를 가리키고 메모리를 해제하는 데 사용되는 ESTAB 포인터이다.

* 1. 모듈 이름 func\_PRINTLOADMAP()
     1. *기능*

링킹 로더 과정이 종료되면 해당 과정에서 생성한 ESTAB을 출력한다.

* + 1. *사용 변수*

ESTAB\*tmp ; ESTAB노드 에 접근하기 위한 ESTAB 포인터

* 1. 모듈 이름 func\_CSECNUM()
     1. *기능*

Loader 명령어의 입력으로 loader obj1 obj2 obj3가 들어오면 각 오브젝트 파일 명을 얻어서 문자형 배열에 저장한다.

* + 1. *사용 변수*

char\*tmp, \*copy ; 인자로 받은 cmd가 strtok로 변하는 것을 방지하기 위해 복사하고 이를 strtok 명령어로 분리하는데 사용

int len ; 인자로 받은 명령어의 길이

int num ; 오브젝트 파일의 개수를 저장하여 반환

* 1. 모듈 이름 func\_LOADpass1()
     1. *기능*

링킹 로더의 첫번쨰 pass로 H record와 D record를 읽어서 ESTAB을 구성한다.

* + 1. *사용 변수*

FILE\*fp ; 오브젝트 파일의 파일 포인터

char \* tmp ; strtok에 사용되는 임시 문자열

char symbol[7], data[7], text[50]; record를 읽을 때 사용할 문자배열이다. symbol에는 심볼을 data에는 해당 심볼의 정보, text는 오브젝트 파일의 한 줄을 저장한다.

int count, cslth, value ; count에는 EXTDEF의 개수, cslth는 csec의 길이, value에는 data 정보를 십진수로 변환한 값을 저장한다.

* 1. 모듈 이름 func\_LOADpass2()
     1. *기능*

링킹 로더의 두 번째 pass로 D record를 제외한 모든 record를 읽는다. 실질적인 로드 과정으로 오브젝트 파일을 읽을 때, 한 바이트 즉 두 개의 숫자를 읽고 이에 대한 정보를 가상 메모리 mem에 저장한다.

Reference record를 읽을 때는 편의를 위해서 reference 배열을 생성하고 이에 각 reference 값들을 인덱스로 지정하여 저장한다.

T record를 읽을 때는 첫 시작에 나온 주소를 이용하여 오브젝트 파일의 정보를 해당 주소의mem 저장한다. 주소는 1씩 증가하지만 오브젝트 파일은 두 개의 숫자를 읽기 때문에 오브젝트 파일에 접근하는 인덱스는 2 증가시킨다.

Modification record를 읽을 때는 만든 reference 배열을 통해서 더하거나 빼야 하는 값을 얻어와 이를 해당 주소에 더하거나 뺀다.

End record를 읽을 때는 첫 번째 섹션인 경우에만 주소를 execaddr 전역변수에 저장한다. 이는 프로그램을 실행시킬 때 시작할 주소다. (progaddr이 0이라면)

이 함수가 종료되면 func\_PRINTLOADMAP()이 실행되어 ESTAB이 출력된다.

* + 1. *사용 변수*

FILE\*fp ; 오브젝트 파일의 파일 포인터

char text[100], value[7], refidx[3] ; text는 오브젝트 파일의 한 줄을 읽어서 저장, value에는 오브젝트 파일에서 필요한 정보를 읽을 때 text로부터 복사하여 저장한다. Refidx는 modification record를 읽을 때 주어진 인덱스를 저장한다.

char \*tmp ; strtok에 사용되는 임시 저장소

int cslth, startadr; csec의 길이와 csec의 시작 주소를 저장한다.

int \*refnum, refcount ; Reference값을 저장하는 동적 배열 refnum과 reference의 개수를 저장하는 refcount

int datanum, dataidx, idx, len ; datanum에는 오브젝트 파일에서 현재 읽는 줄의 데이터 개수, dataidx는 현재 읽고 있는 위치, idx는 refnum 배열에 접근하기 위한 인덱스, len은 현재 읽은 오브젝트 파일의 한 라인의 길이를 저장한다.

unsigned char data; 가상 메모리에 저장할 정보를 저장한다.

* 1. 모듈 이름 func\_initializeregister()
     1. *기능*

Register의 값들을 전부 0으로 초기화한다.

* + 1. *사용 변수*
  1. 모듈 이름 func\_breakpoint()
     1. *기능*

Break point와 관련된 명령어를 처리한다. bp만 들어온 경우 현재까지 저장된 중단점을 전부 출력하고, bp clear이면 모든 중단점을 삭제한다. Bp breakpoint이면 해당 break point가 프로그램의 범위에 맞는지 파악하고 맞으면 중단점을 추가하고 아니면 에러를 출력한다.

* + 1. *사용 변수*

int len ; 명령어의 길이

int addr ; break point를 십진수 변환한 값 저장

char \* copy; 문자열을 복사하여 명령어를 수행한다.

* 1. 모듈 이름 func\_run()
     1. *기능*

로드한 copy.obj 프로그램을 실행하는 함수이다. 중단점이 있으면 중단점까지 실행하고 중단하고 다시 run 명령어를 실행하면 중단점부터 수행한다. 중단점이 없이 처음 시작하면 progaddr과 execaddr을 참고하여 실행을 시작할 첫 주소를 정한다. 해당 주소부터 저장된 정보를 읽기 시작하여 해당 정보가 어떤 명령어 인지 구분하는데 이는 첫 바이트를 읽어서 해당 바이트의 앞 4비트가 어떤 값인지 파악하여 구분한다. Register 연산은 하나의 바이트만 더 읽어서 처리하고 그 이외의 명령어 인 경우에 3형식인지 4형식인지 두 번째 바이트에서 e비트가 1인지 파악하여 구분한다. 3형식이면 두 개의 바이트를 더 읽고 getdatformat3 함수를 통해서 target address를 구한다. 4형식이면 세 개의 바이트를 더 읽고 getdataforamt4 함수를 통해서 target address를 구한다. 이 때, immediate addressing 인 경우에는 target address를 통해서 메모리 접근을 하지 않고 target address 자체로 원하는 값이기 때문에 이를 구분한다. 첫 번째 바이트에서 n=0, i=1이라는 점을 이용하여 immediate addressing 인지 구분할 수 있다.

* + 1. *사용 변수*

unsigned char object\_code[4] ; 가상 메모리로부터 받아온 데이터를 저장한다.

int opcode, data ; object\_code의 첫 바이트로부터 해당 명령어의 opcode 값을 유추하여 opcode에 저장한다. Data는 각 명령어를 형식 별로 구분했을 때, 3형식과 4형식은 object\_code를 참조하여 target address를 data에 저장한다.

* 1. 모듈 이름 func\_getdataformat3()
     1. *기능*

3형식 명령어의 addressing mode를 파악하여 target address를 구하여 반환한다.

* + 1. *사용 변수*

int value ; 가상 메모리로부터 얻어온 정보를 십진수로 변환하여 저장한다. 이후에 주소 처리방식에 따라 값이 변한다.

int tmp1, tmp2 ; indirect addressing에 사용하는 변수

* 1. 모듈 이름 func\_getdataformat4()
     1. *기능*

4형식 명령어의 addressing mode를 파악하여 target address를 구하여 반환한다.

* + 1. *사용 변수*

int value ; 가상 메모리로부터 얻어온 정보를 십진수로 변환하여 저장한다. 이후에 주소 처리 방식에 따라 값이 변한다.

* 1. 모듈 이름 func\_regvalue()
     1. *기능*

Sic xe machine의 레지스터들의 값을 출력한다. A X L B S T PC SW의 값을 출력한다.

* + 1. *사용 변수*

1. **전역 변수 정의**
   1. **int progaddr**

프로그램의 시작 주소

* 1. **int execaddr**

실행을 시작할 주소

* 1. **int program\_len**

프로그램의 길이

* 1. **ESTAB est[3]**

로딩 과정에서 생성한 ESTAB으로 EXTDEF을 저장할 테이블

* 1. **int csec**

컨트롤 섹션의 수를 저장할 변수

* 1. **int bp[20]**

중단점 배열

* 1. **int bp\_len**

중단점의 수

* 1. **int current\_bp**

현재 bp의 인덱스

* 1. **int is\_bp**

중단점의 존재 여부를 나타내는 변수

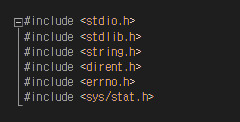
* 1. **int reg[10]**

레지스터 배열

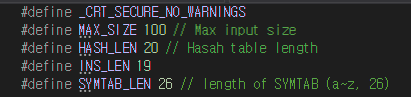
1. **코드**

/\*포함되는 파일\*/

#include “20161647.h”

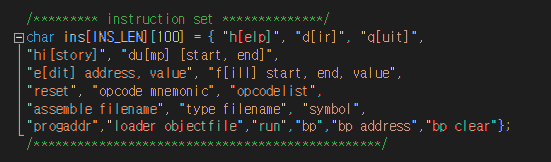


/\*정의되는 상수\*/

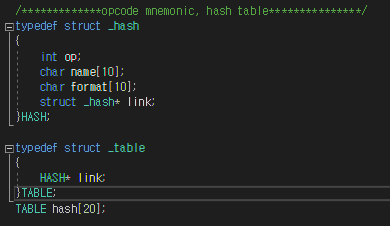


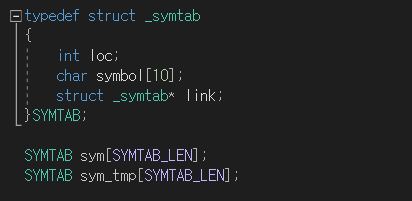
/\*명령어가 추가되어 INS\_LEN이 증가했다. \*/

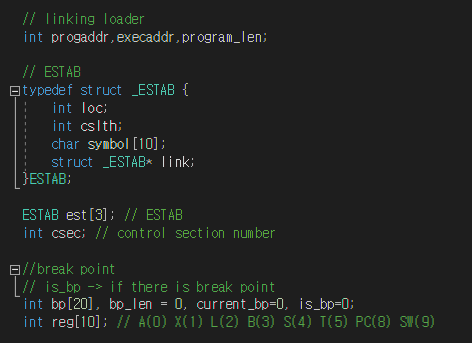
/\*전역 변수\*/



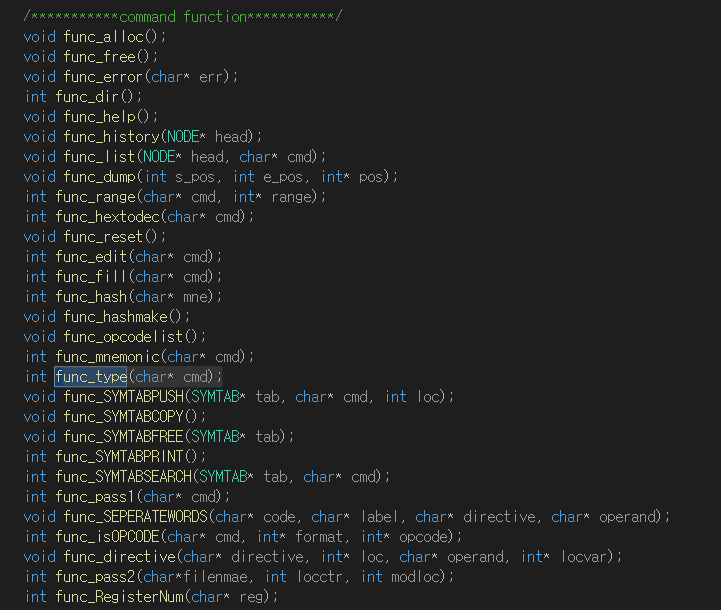


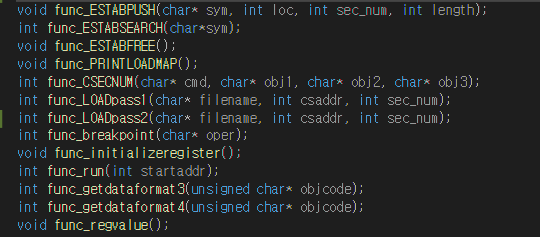






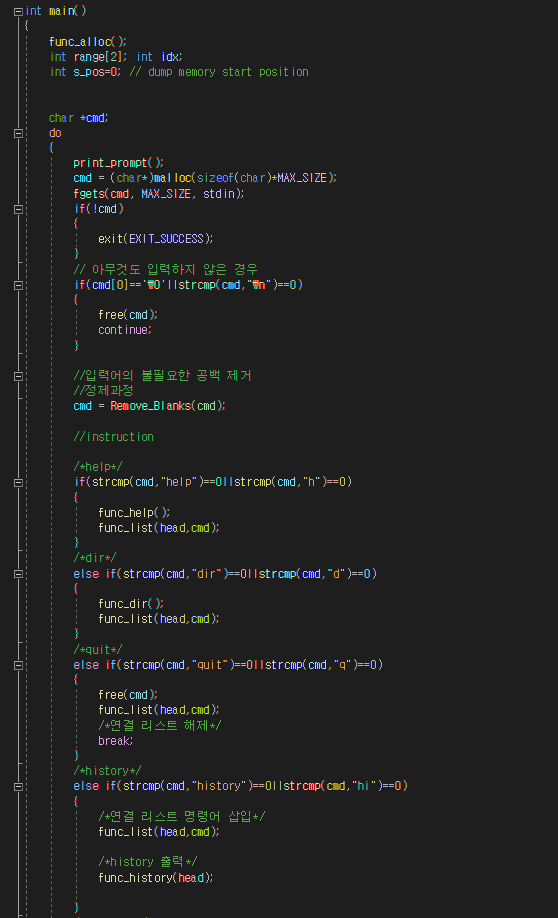
/\*함수 원형\*/

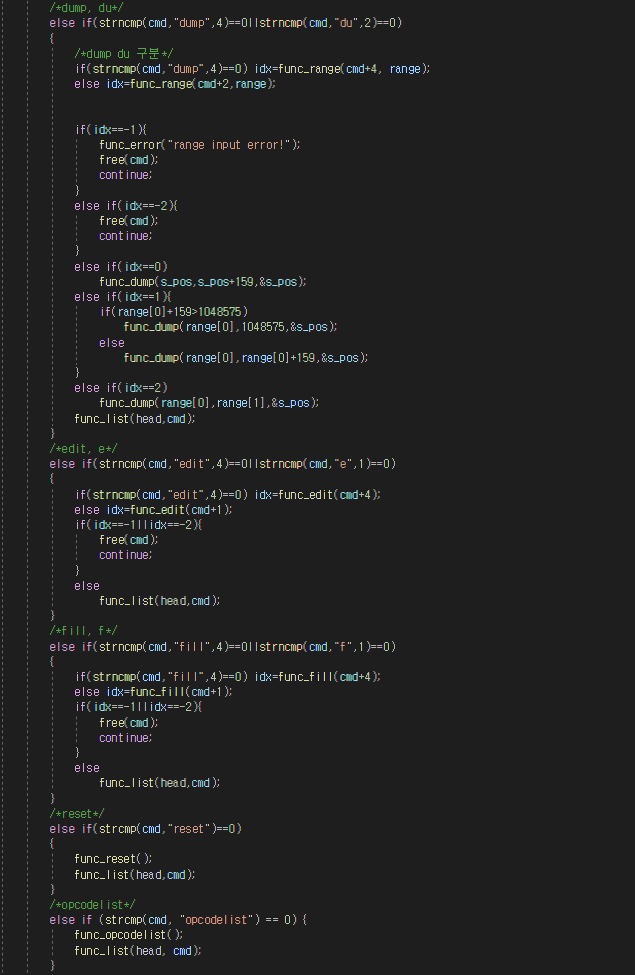


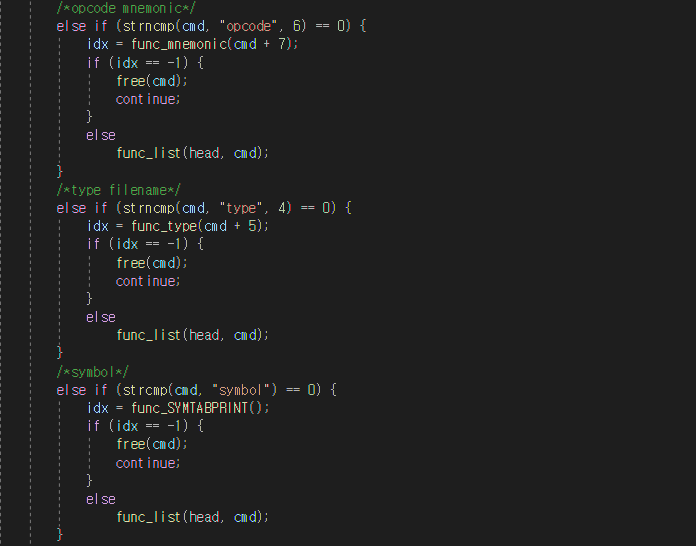


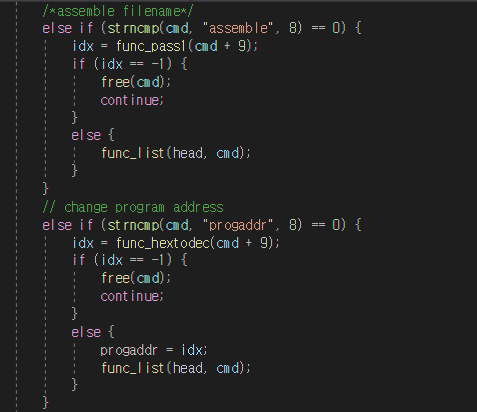
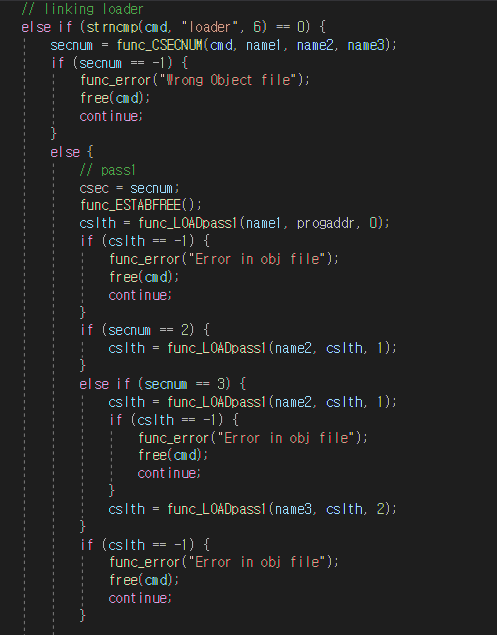
/\*프로그램 시작\*/

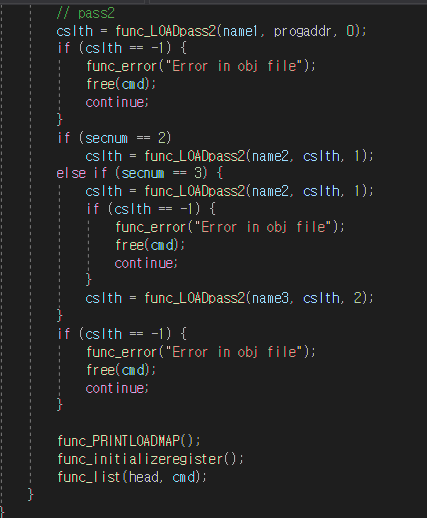
/\* loader, run, bp 명령어 추가 \*/

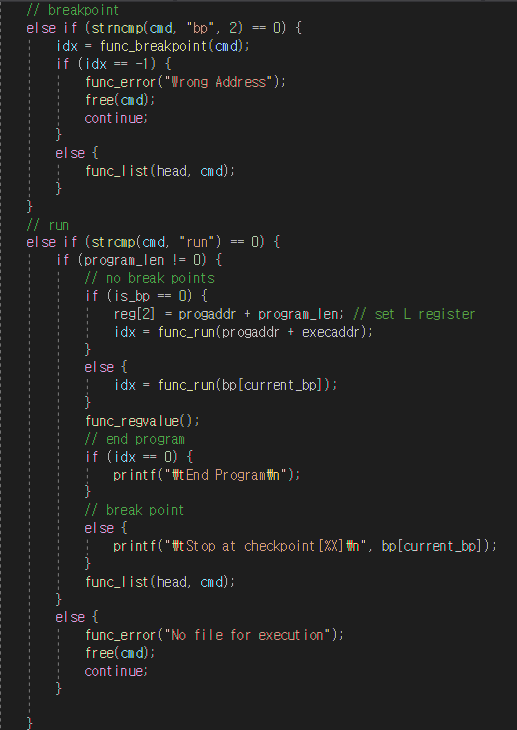


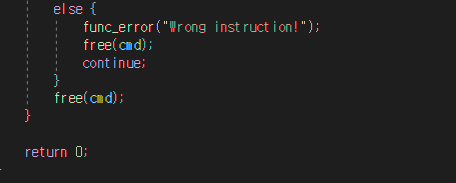










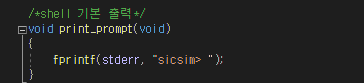
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: print\_prompt\*/

/\*목적: 셸 출력 형식으로 sicsim> 을 출력한다\*/

/\*리턴 값: 없음\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

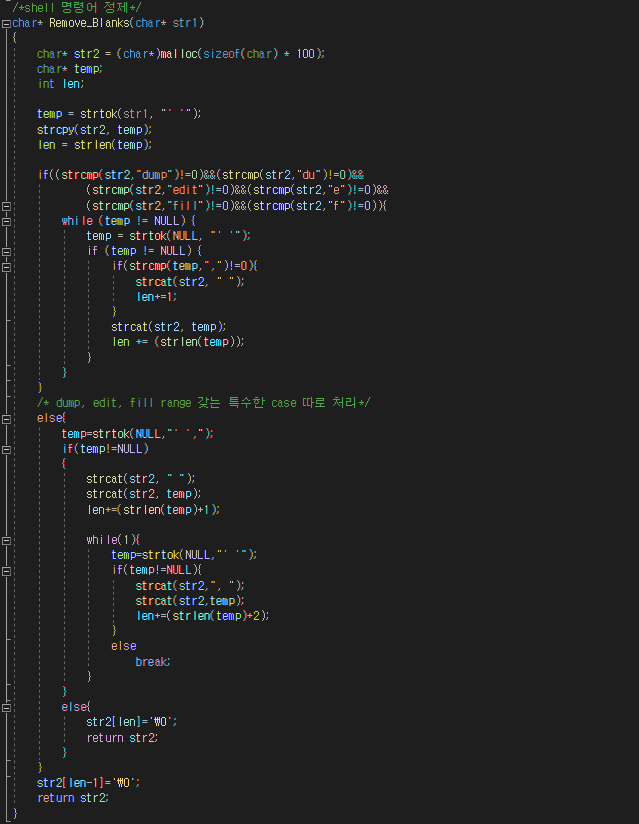


/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: Remove\_Blanks\*/

/\*목적: 사용자가 입력한 명령어를 정제한다.\*/

/\*리턴 값: 정제된 문자열을 반환한다.\*/

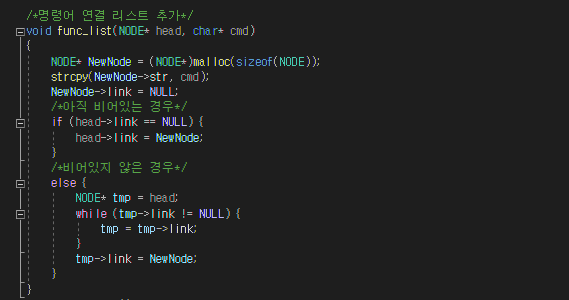


/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_list\*/

/\*목적: 명령어 연결 리스트에 새 노드를 삽입한다.\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

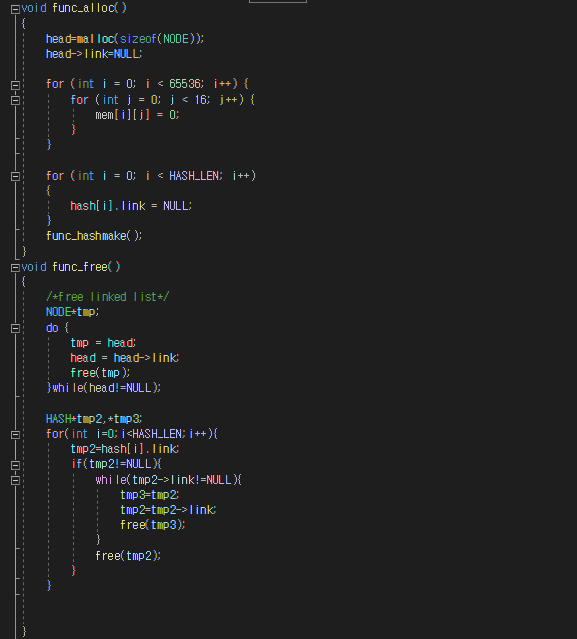
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_alloc\*/

/\*목적: 프로그램 시작 때 필요한 초기화 작업을

한다.\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

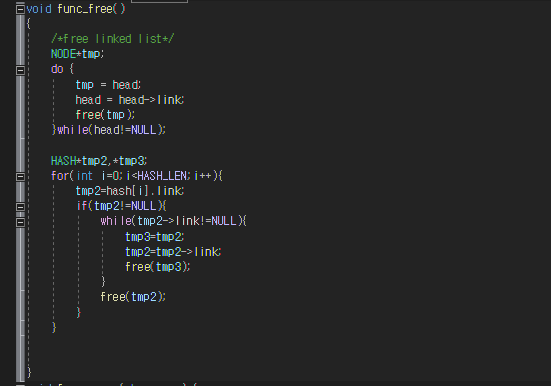
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_free\*/

/\*목적: 프로그램 종료 시에 필요한 해제 작업을

한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_error\*/

/\*목적: 명령어 처리 과정에서 발생한 오류를 출력

한다\*/

/\*리턴 값: 없음\*/



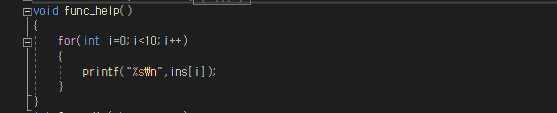
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_help\*/

/\*목적: Help 명령어의 기능을 수행한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

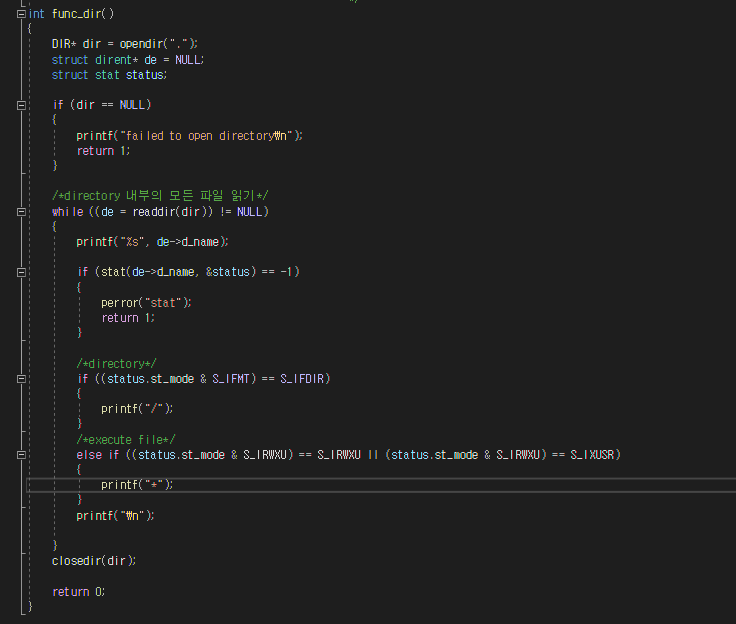
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_dir\*/

/\*목적: Dir명령어의 기능을 수행한다\*/

/\*리턴 값: 1; 디렉토리 열기에 실패한 경우

0; 성공적으로 명령어를 수행한 경우\*/



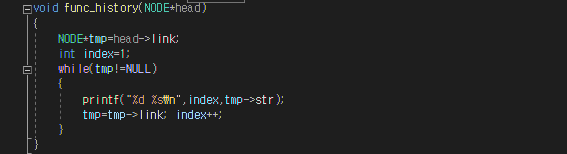
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_history\*/

/\*목적: History 명령어의 기능을 수행한다\*/

/\*리턴 값: 없음\*/



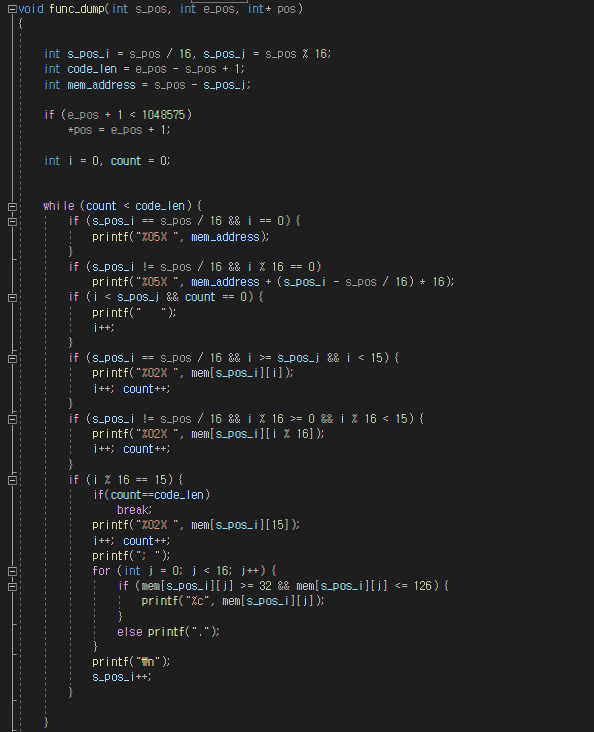
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_dump\*/

/\*목적: Dump 명령어의 기능을 수행한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_range\*/

/\*목적: Dump 명령어의 범위를 검사한다\*/

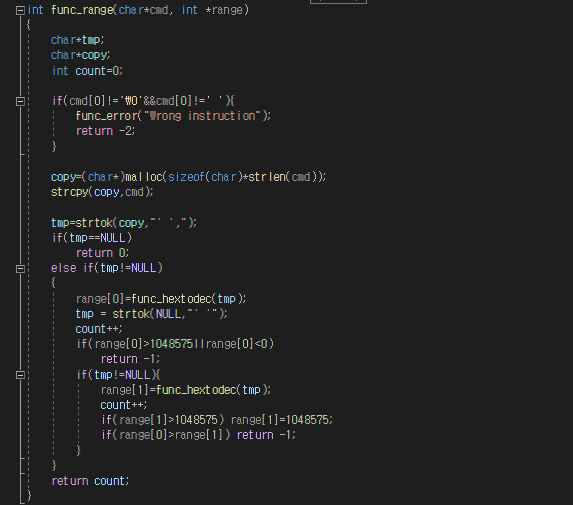
/\*리턴 값: -2 dump 명령으로 구분되었는데 올바르지 않은 경우

-1 입력된 범위가 올바르지 않은 경우

0 범위가 존재하지 않는 경우

1 시작 주소만 입력된 경우

2 시작 주소와 끝 주소가 입력된 경우\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

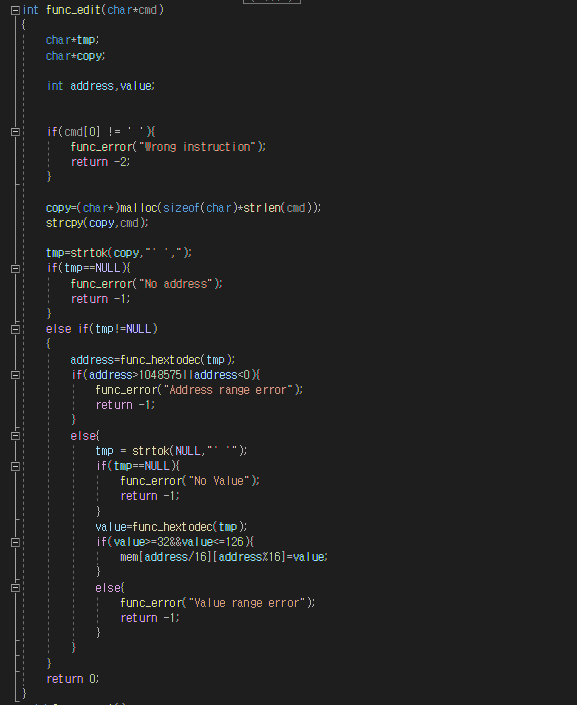
/\*함수: func\_edit\*/

/\*목적: Edit 명령어의 기능을 수행한다\*/

/\*리턴 값: -2 올바르지 않은 명령어 인 경우

-1 주소가 입력되지 않았거나, 저장한 값이 입력되지 않은 경우

0 올바른 명령어의 수행이 이루어진 경우\*/



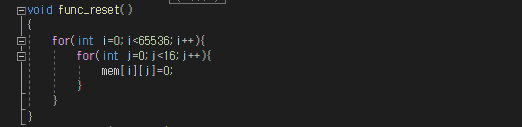
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_reset\*/

/\*목적: Reset명령어의 기능을 수행한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

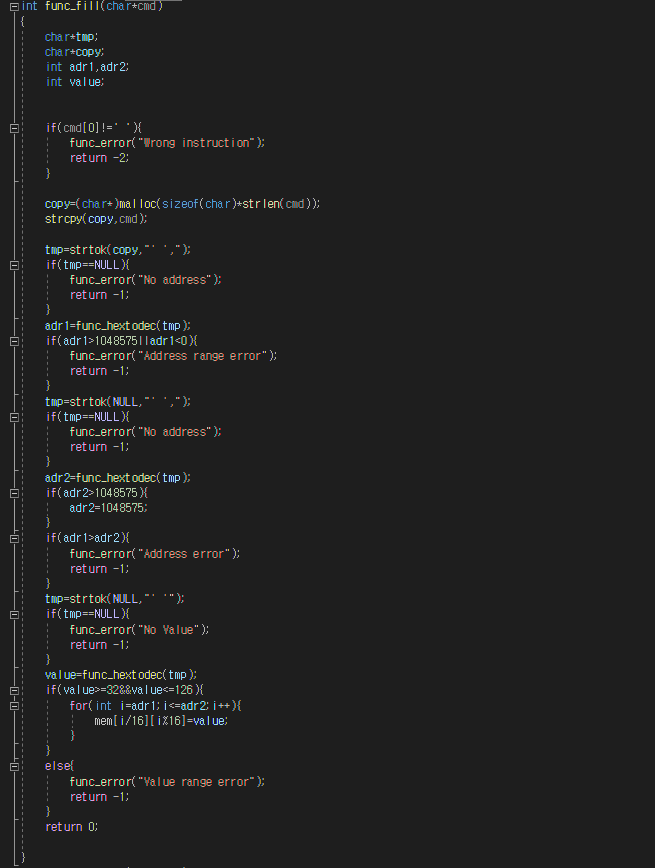
/\*함수: func\_fill\*/

/\*목적: Fill명령어의 기능을 수행한다\*/

/\*리턴 값: -2 명령어가 올바르지 않은 경우

-1 주소가 입력되지 않은 경우 또는 값이 입력되지 않은 경우

0 올바른 명령의 수행이 이루어진 경우\*/



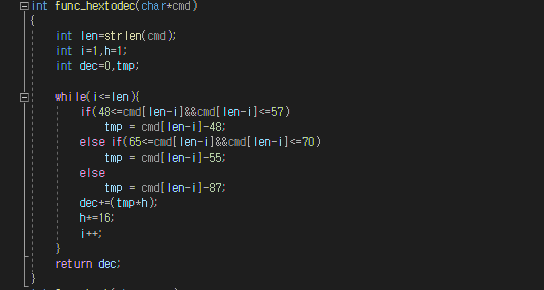
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_hextodec\*/

/\*목적: 문자열로 입력된 16진수의 값을 십진수로 변경한다\*/

/\*리턴 값: 10진수로 변경된 정수 값을 반환한다\*/



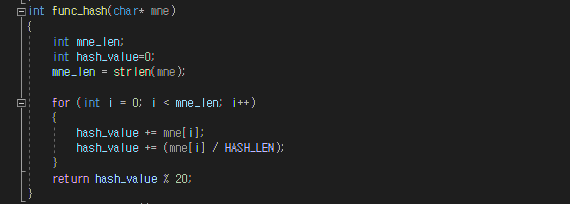
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_hash\*/

/\*목적: hash 함수를 통해 입력된 명령어의 hash 값을 구한다\*/

/\*리턴 값: 작성한 hash 함수를 통해 얻은hash value를 반환한다\*/



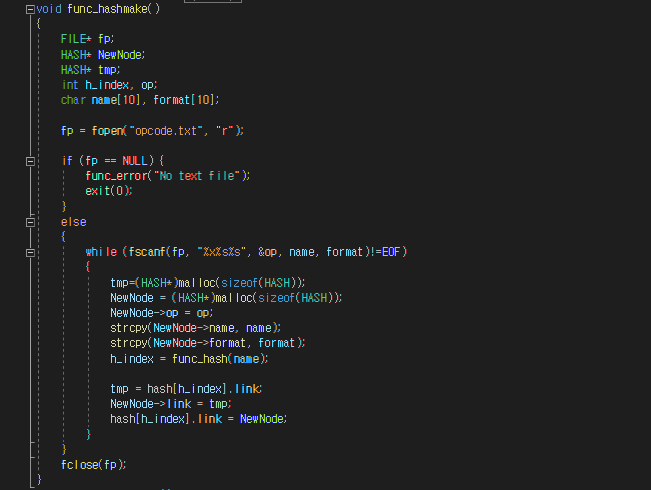
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_hashmake\*/

/\*목적: opcode.txt를 이용해 hash table을 작성한다\*/

/\*리턴 값: 없음\*/



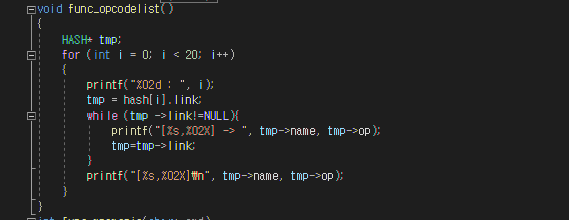
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_opcodelist \*/

/\*목적: hashmake 함수로 작성한 hash table을 출력한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

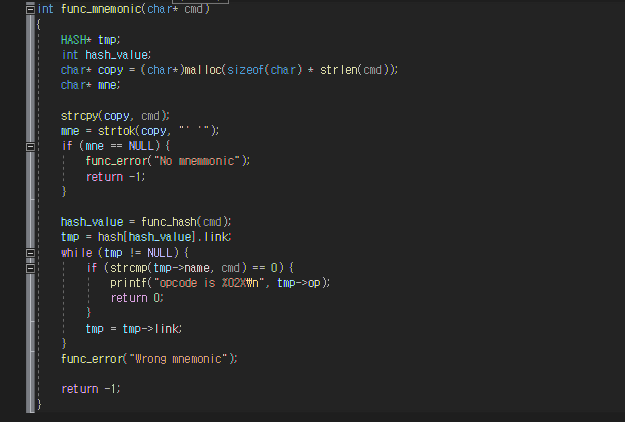
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_mnemonic \*/

/\*목적: opcode mnemonic 명령어의 기능을 수행한다\*/

/\*리턴 값: -1 mnemonic이 입력되지 않았거나 찾지 못한 경우

0 올바른 입력이 들어와 테이블에서 이를 찾은 경우\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

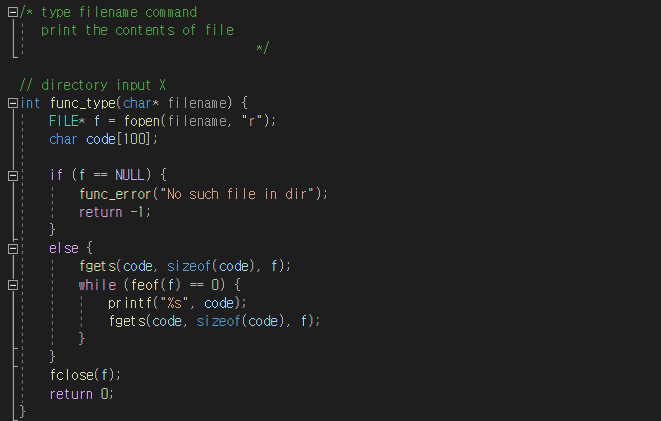
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_type \*/

/\*목적: 파일의 내용을 출력 한다.\*/

/\*리턴 값: -1 파일이 존재하지 않는 경우 \*/

0 파일이 존재하여 파일의 내용을 정상 출력한 경우\*/



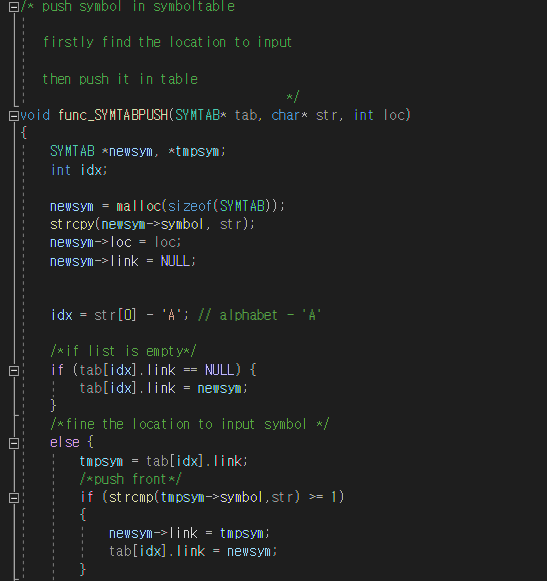
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

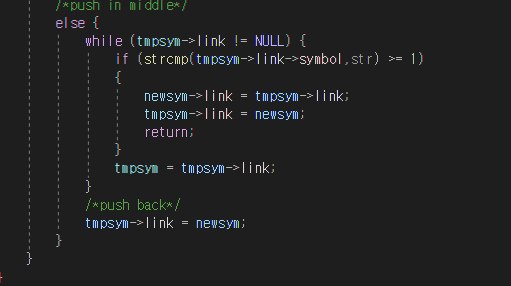
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SYMTABPUSH \*/

/\*목적: 심볼 테이블에 원소를 추가한다. \*/

/\*리턴 값: 없음 \*/





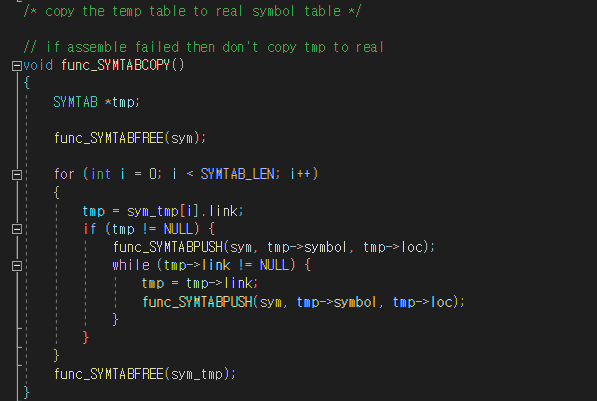
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SYMTABCOPY \*/

/\*목적: 임시 테이블의 내용을 실제 테이블에 복사 \*/

/\*리턴 값: 없음 \*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

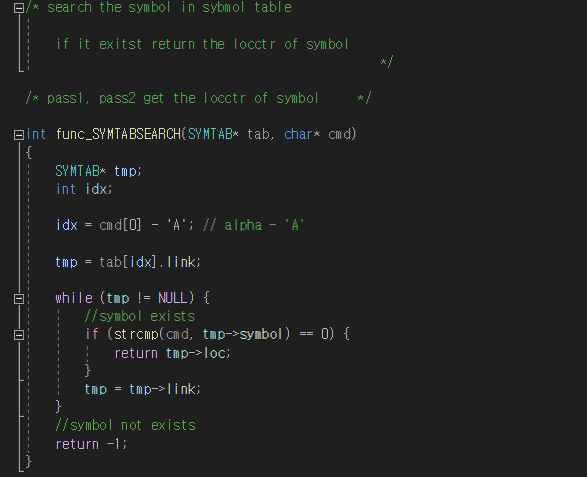
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SYMTABSEARCH \*/

/\*목적: 심볼 테이블에 해당 심볼이 존재하는지 탐색 \*/

/\*리턴 값: 심볼이 존재하면 해당 심볼의 location counter

심볼이 존재하지 않으면 -1을 반환한다. \*/



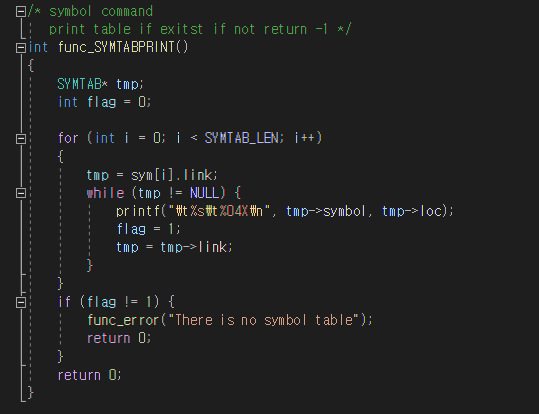
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SYMTABPRINT \*/

/\*목적: 심볼 테이블을 출력한다\*/

/\*리턴 값: 0 값을 리턴 한다.\*/



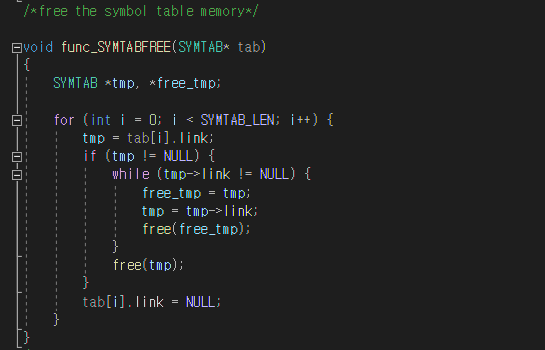
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SYMTABFREE \*/

/\*목적: 심볼 테이블에 동적할당 메모리를 해제한다.\*/

/\*리턴 값: 없음.\*/



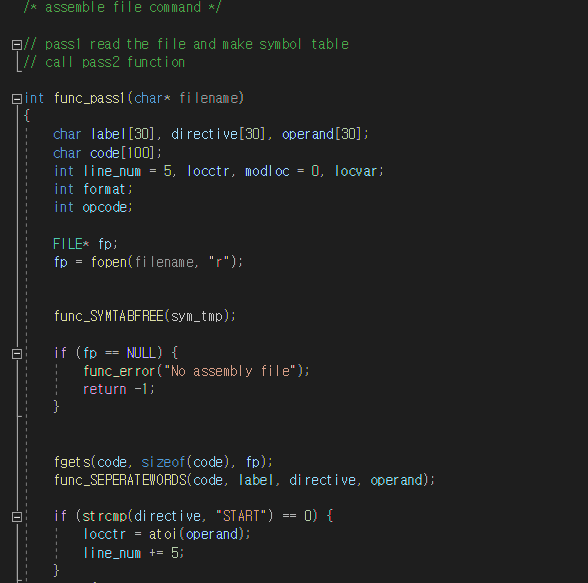
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

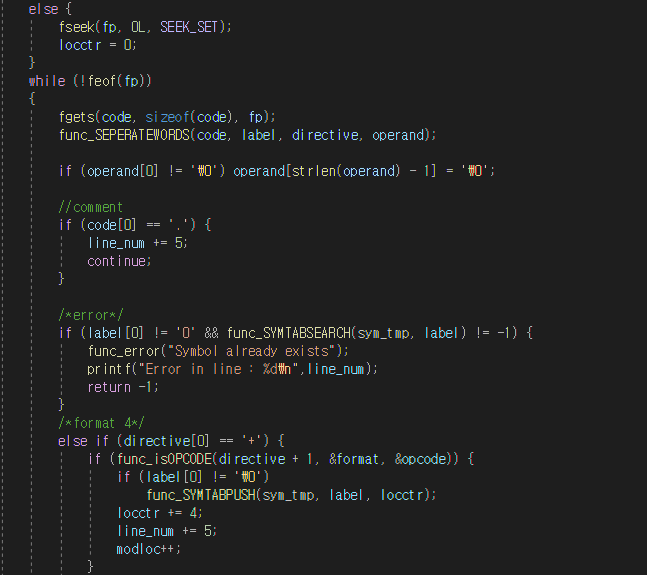
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

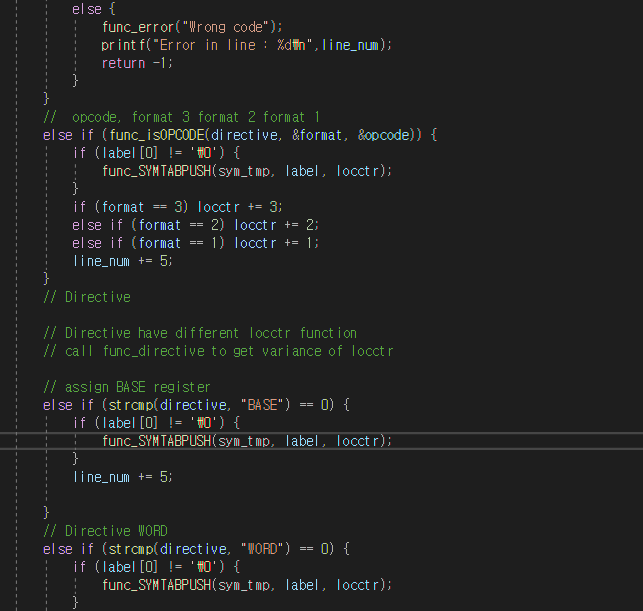
/\*함수: func\_pass1 \*/

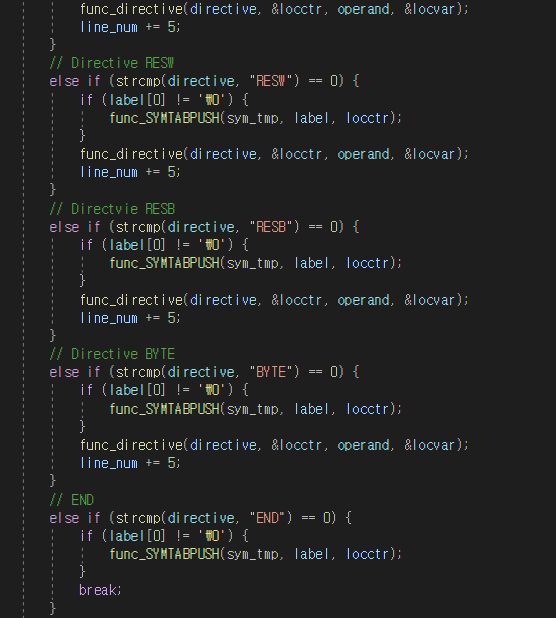
/\*목적: 어셈블 명령의 첫번째 어셈블리 파일 읽는 과정\*/

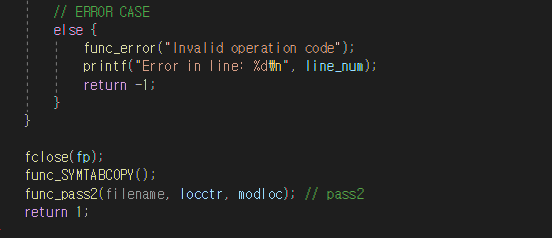
/\*리턴 값: 에러가 존재하면 -1을 존재하지 않으면 1을 리턴.\*/











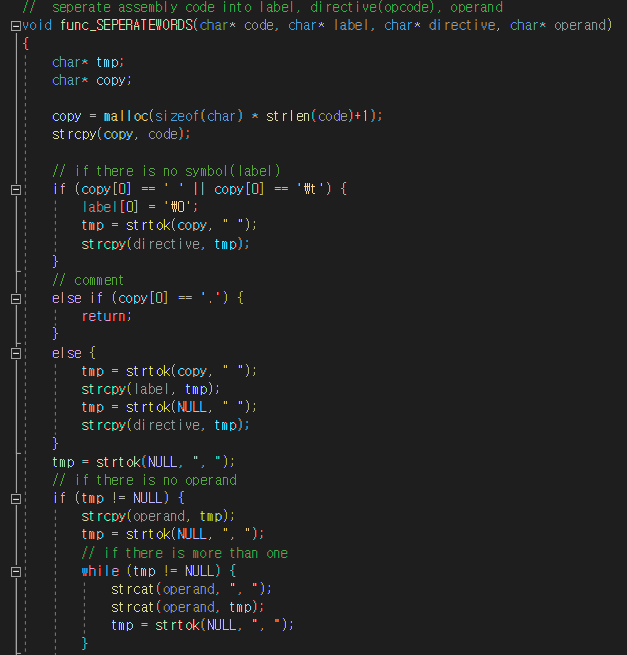
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

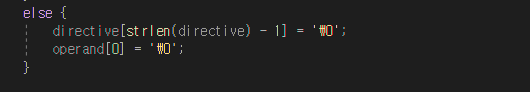
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_SEPERATEWORDS \*/

/\*목적: 문자열을 세 가지 파트로 분리한다.\*/

/\*리턴 값: 없음.\*/





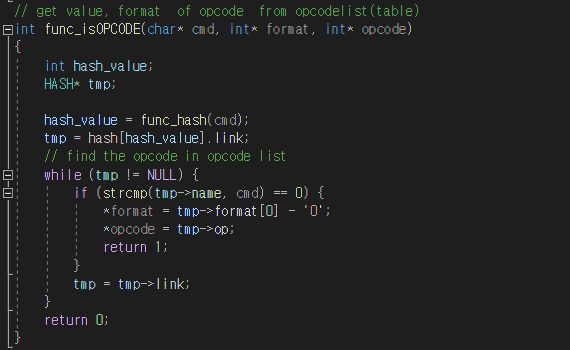
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_isOPCODE \*/

/\*목적: opcode인지 확인하고 format값과 opcode 값을 저장\*/

/\*리턴 값: opcode이면 1을 아니면 0을 반환한다.\*/



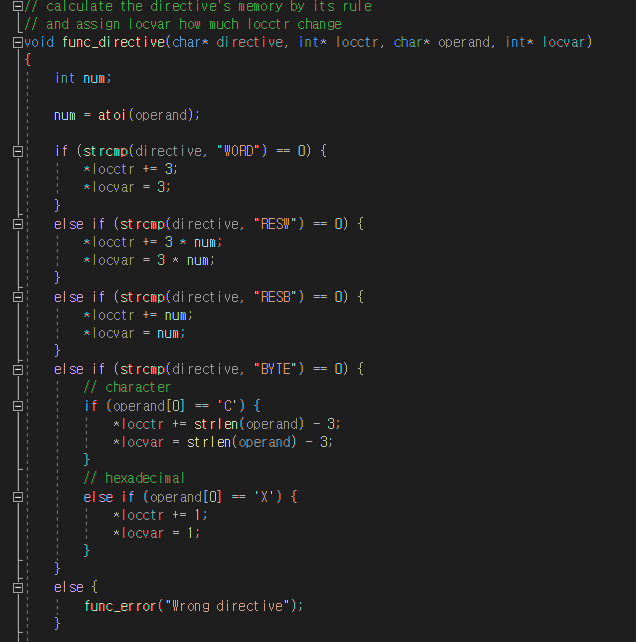
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_directive \*/

/\*목적: WORD, RESW, RESB, BYTE를 처리한다.\*/

/\*리턴 값: 없음 \*/



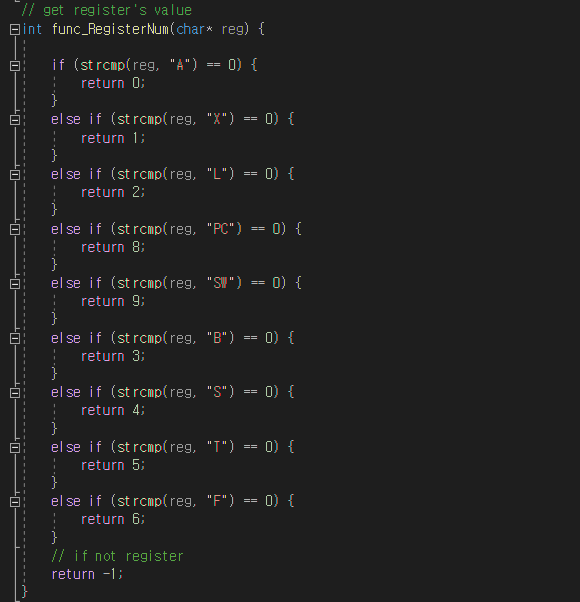
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_RegisterNum \*/

/\*목적: 레지스터 고유 값 반환.\*/

/\*리턴 값: 레지스터 고유 값 레지스터가 아니면 -1 \*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

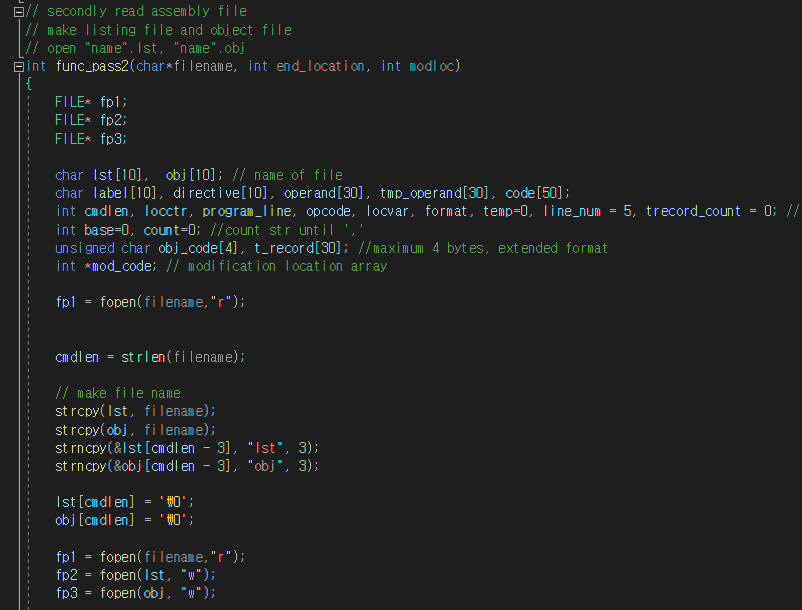
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

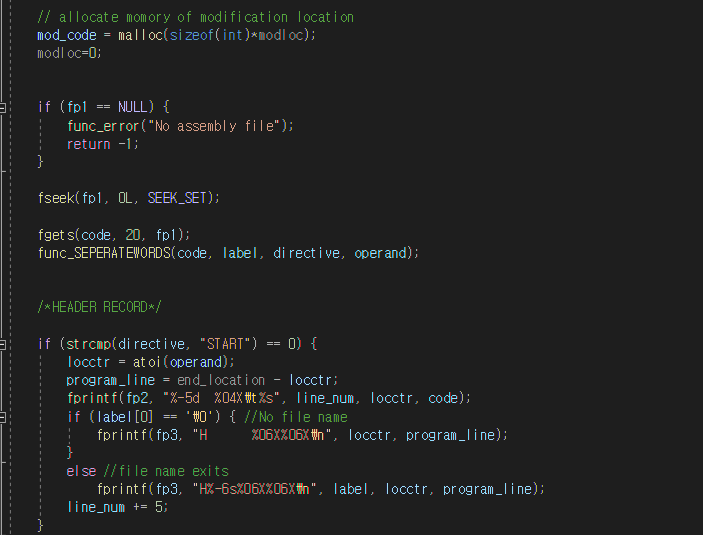
/\*함수: func\_pass2 \*/

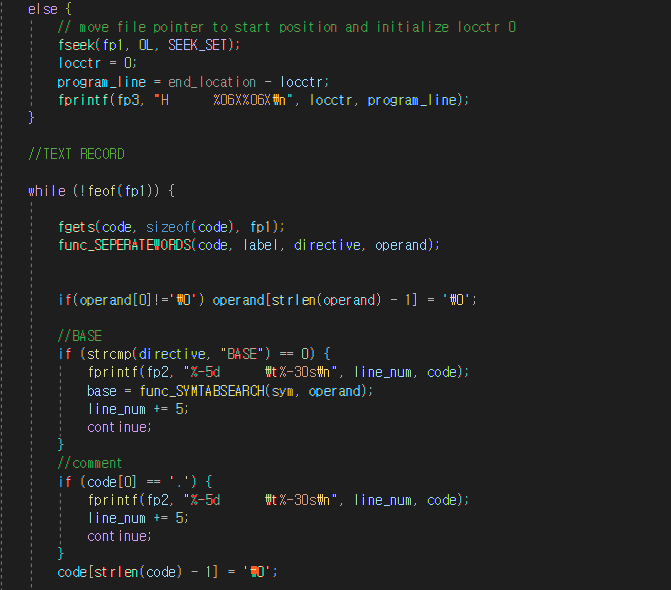
/\*목적: 어셈블의 두 번째 파일 읽는 과정, 오브젝트 파일과

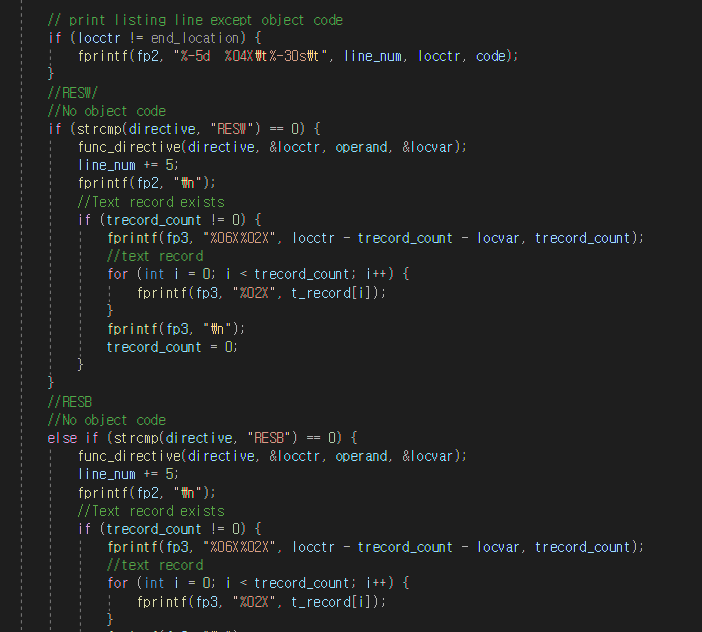
리스팅 파일을 생성한다. \*/

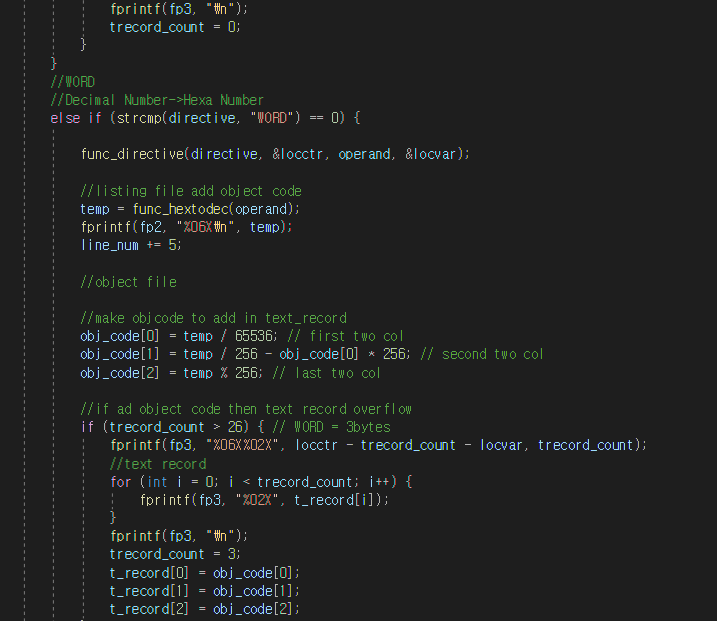
/\*리턴 값: 성공적으로 리스팅 파일과 오브젝트 파일을 생성하면 1을 반환한다.\*/

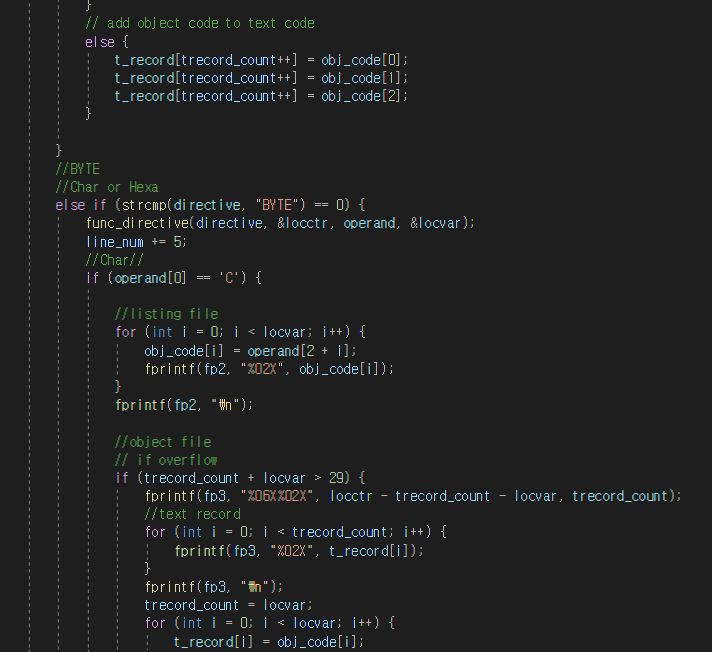


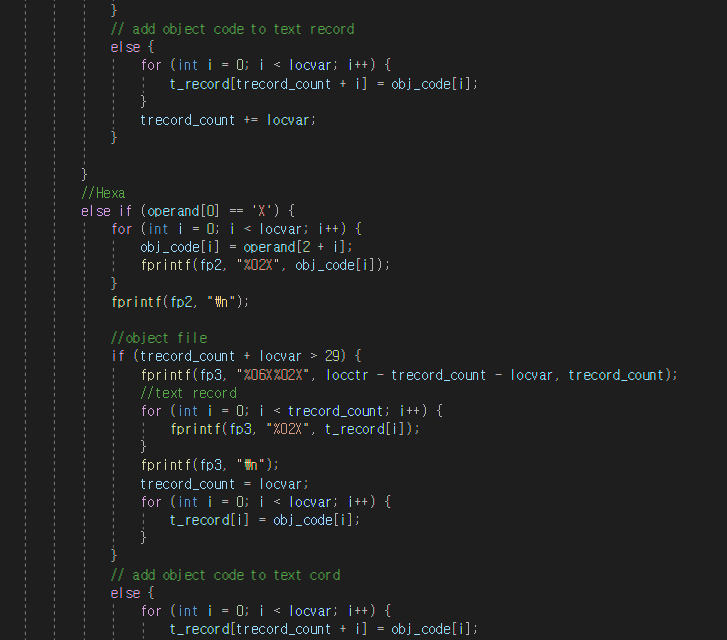


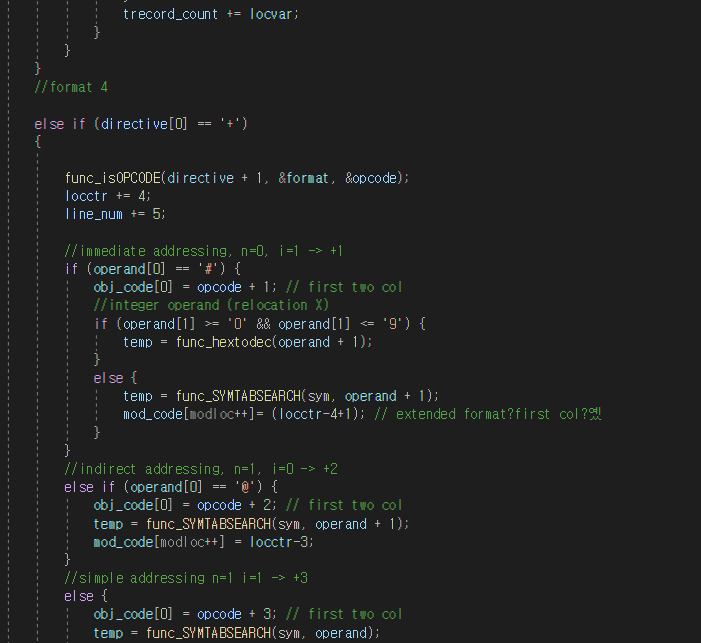


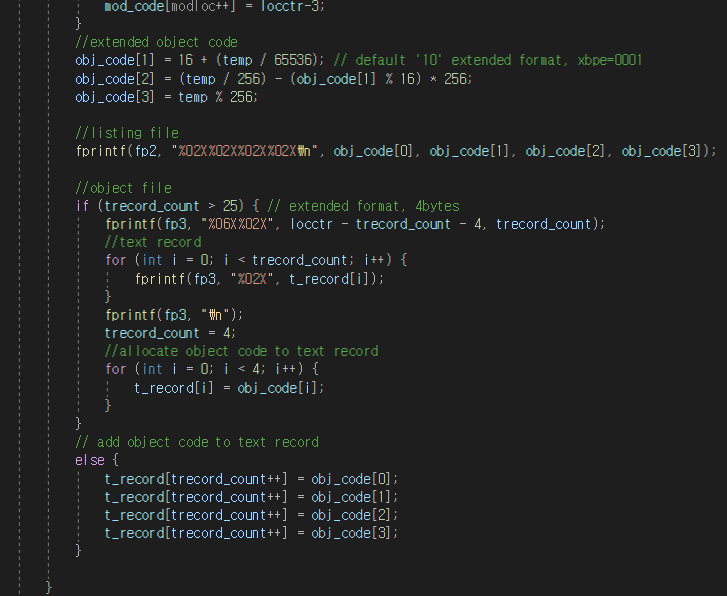


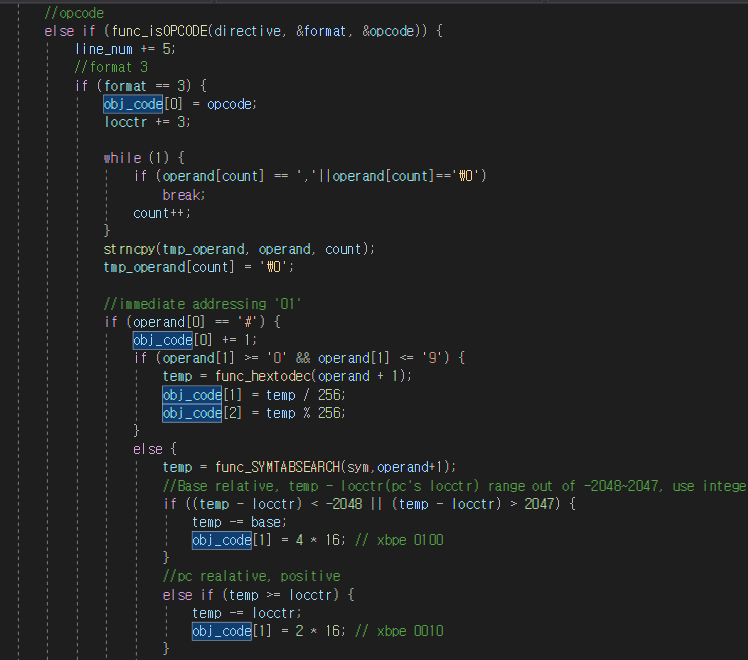


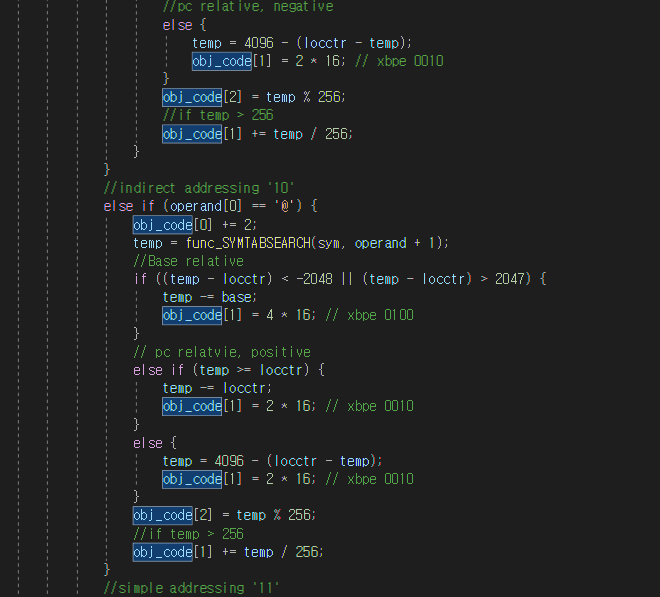


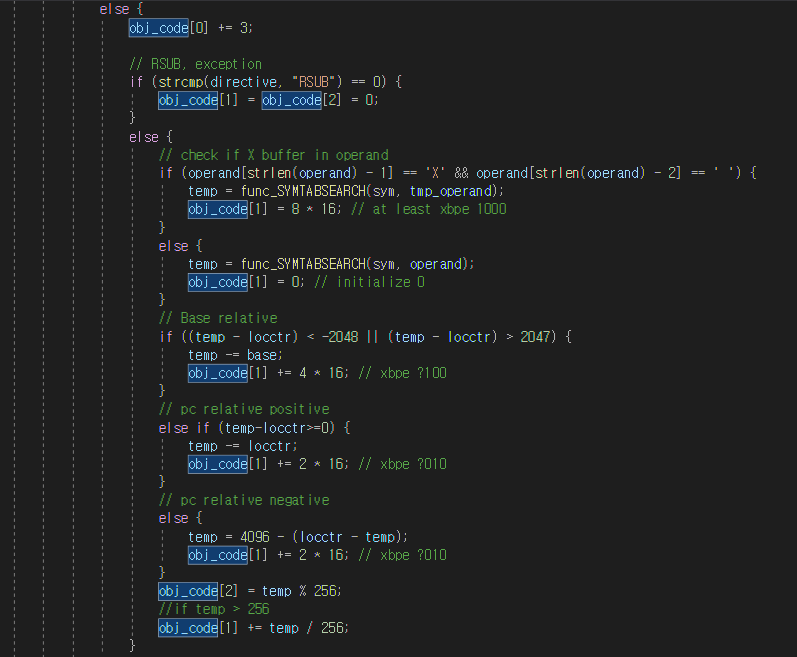
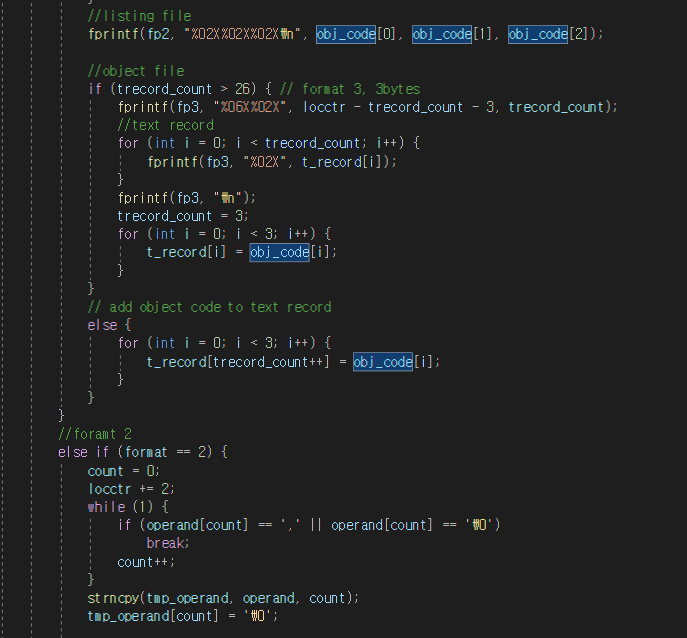


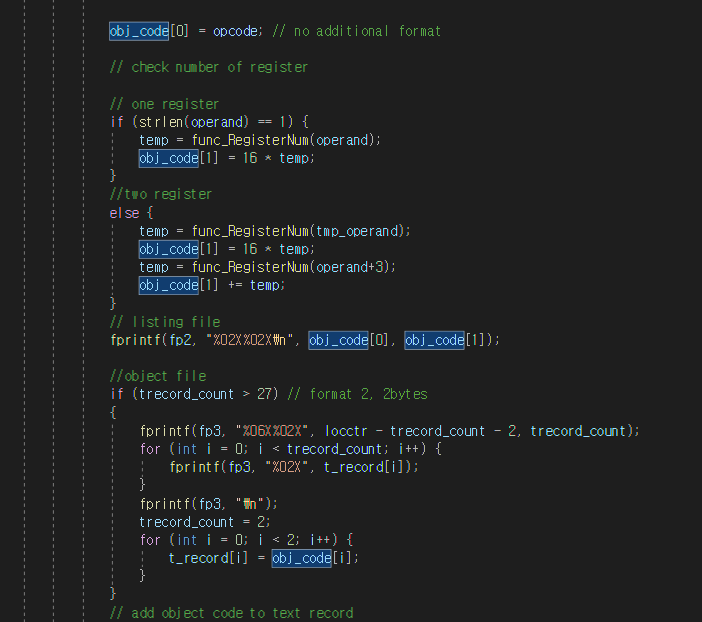


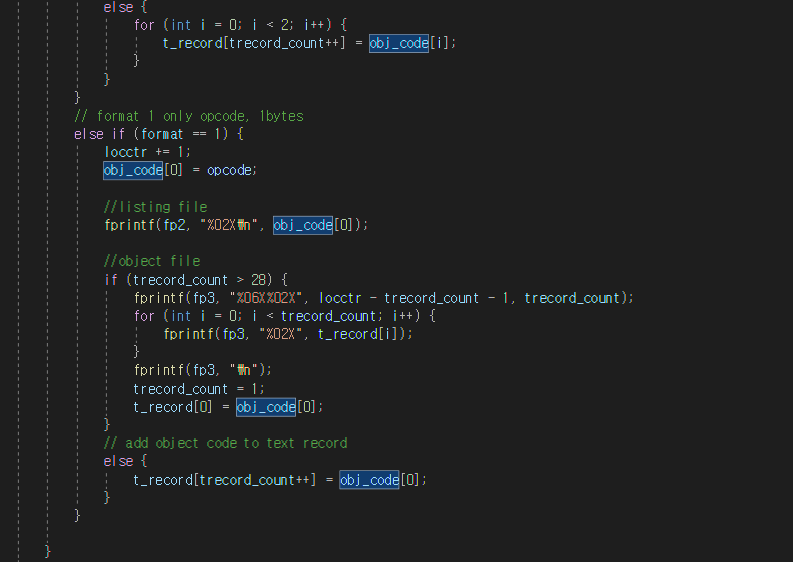


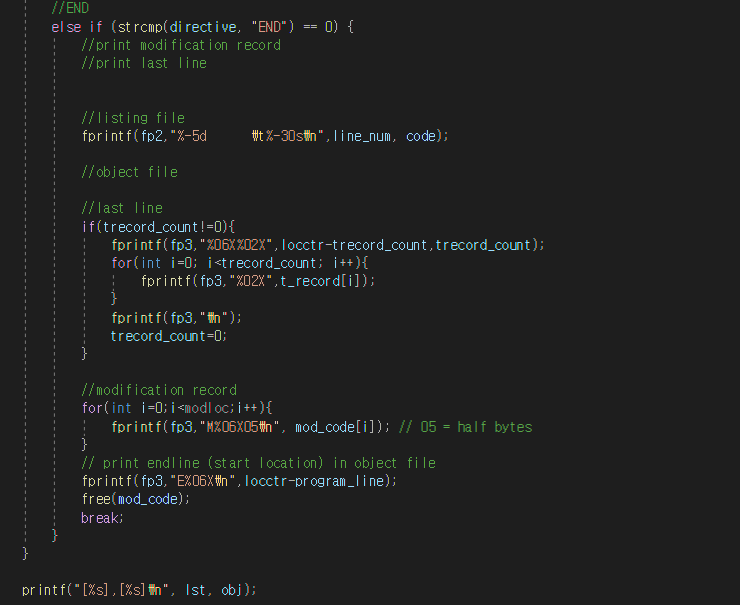












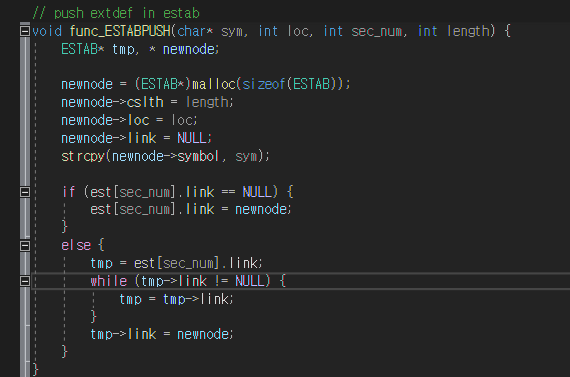
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_ESTABPUSH \*/

/\*목적: ESTAB에 노드 추가\*/

/\*리턴 값: 없음 \*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

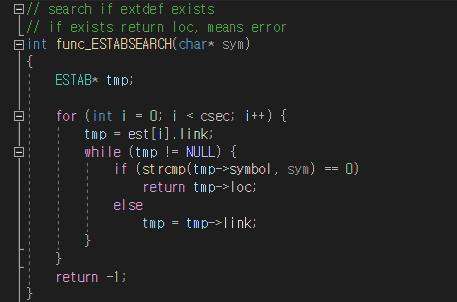
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_ESTABSEARCH \*/

/\*목적: ESTAB에서 심볼 탐색\*/

/\*리턴 값: 심볼이 존재하면 심볼의 location 반환

없으면 -1 반환 \*/



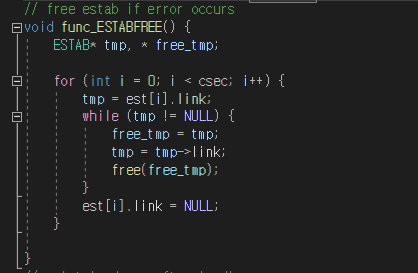
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_ESTABFREE \*/

/\*목적: ESTAB초기화\*/

/\*리턴 값: 없음\*/



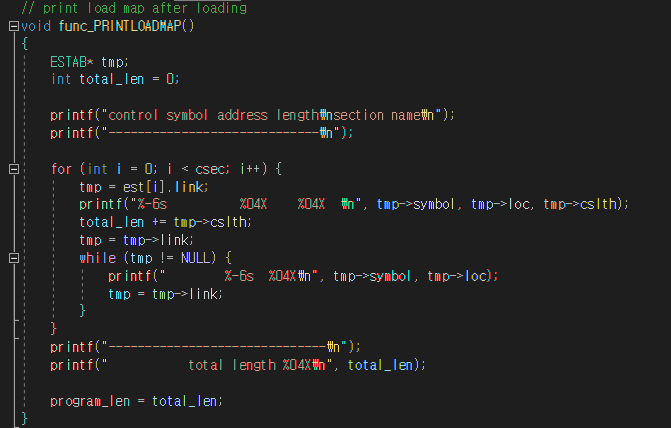
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_PRINTLOADMAP \*/

/\*목적: ESTAB load map 출력\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

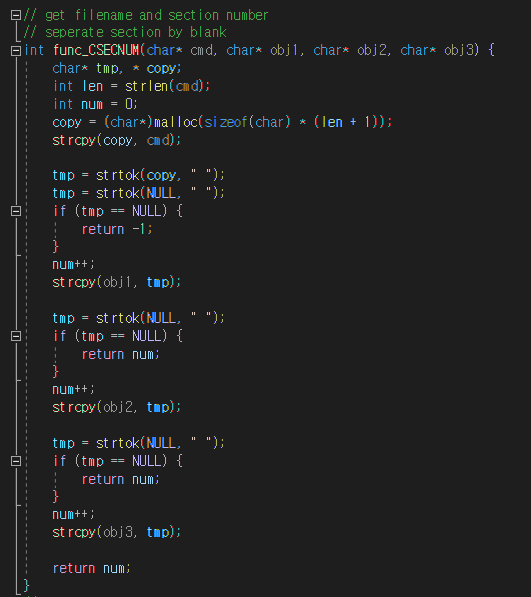
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_CSECNUM \*/

/\*목적: loader 명령어의 오브젝트 파일명 얻고

컨트롤 섹션의 개수를 구함\*/

/\*리턴 값: 컨트롤 섹션의 개수를 반환\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

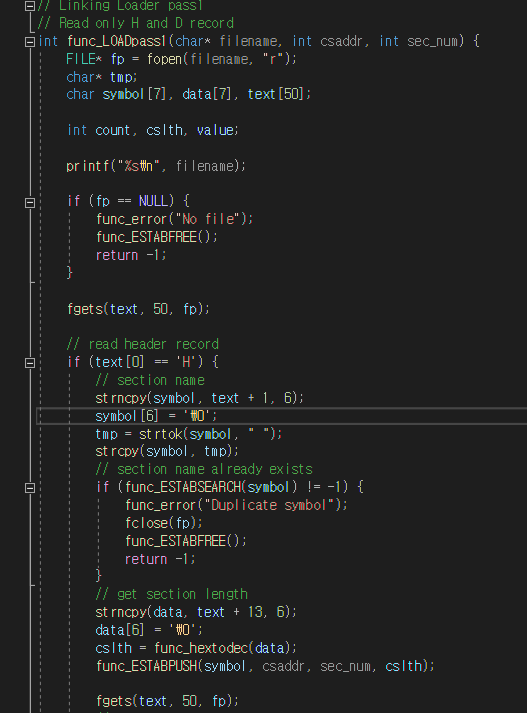
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

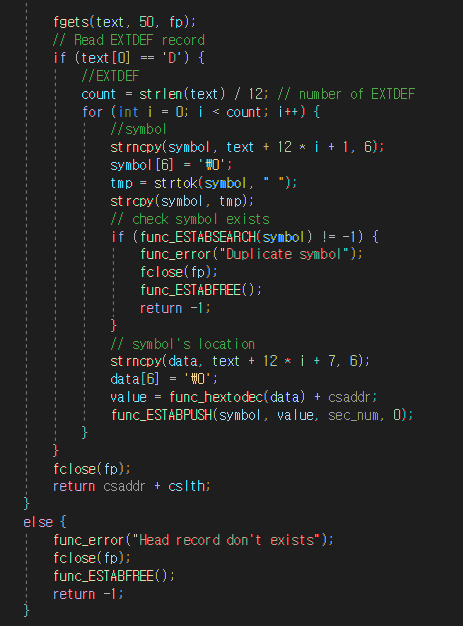
/\*함수: func\_LOADpass1 \*/

/\*목적: loader 명령어의 첫번째 pass, ESTAB을 생성한다\*/

/\*리턴 값: 성공적으로 진행되면 컨트롤 섹션 시작주소에

컨트롤 섹션 길이를 더한 값을 오류가 존재하면 -1을 반환한다\*/





/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_LOADpass2 \*/

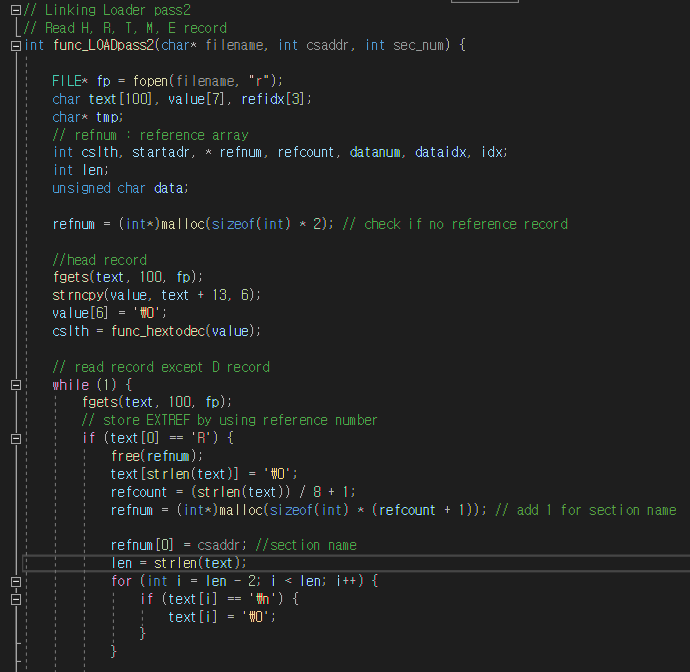
/\*목적: 오브젝트 파일을 읽어서 가상 메모리에 데이터를

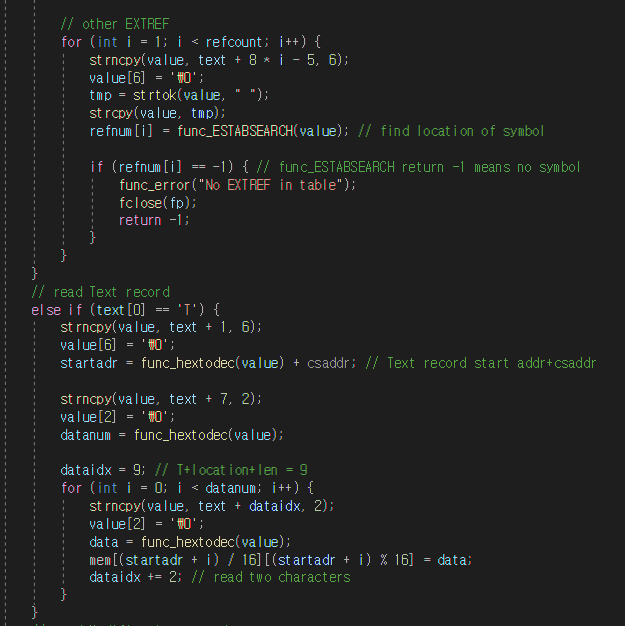
올리는 실질적인 로드 과정을 진행한다\*/

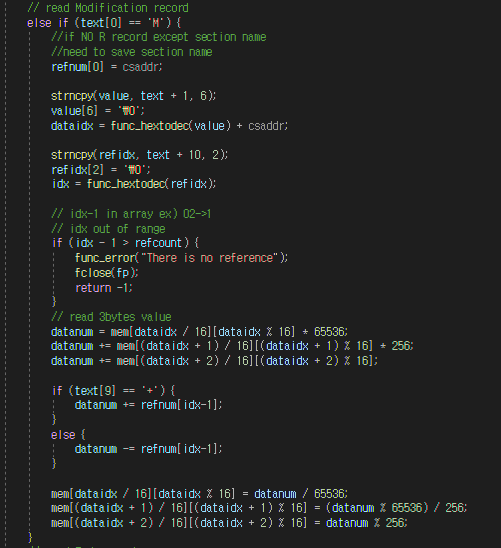
/\*리턴 값: 정상적인 반환은 컨트롤 섹션 시작 주소에

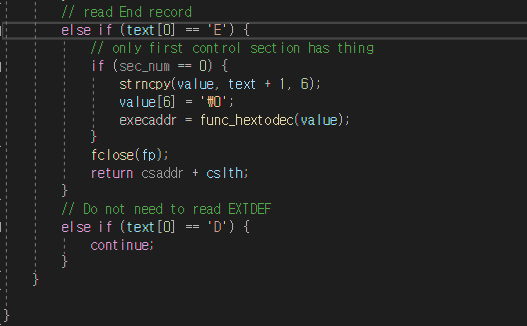
컨트롤 섹션의 길이를 더한 값이고 오류가 발생하면

-1을 반환한다\*/









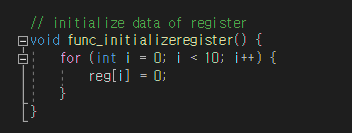
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_initializeregiser \*/

/\*목적: 레지스터 값을 전부 0으로 초기화한다\*/

/\*리턴 값: 없음\*/



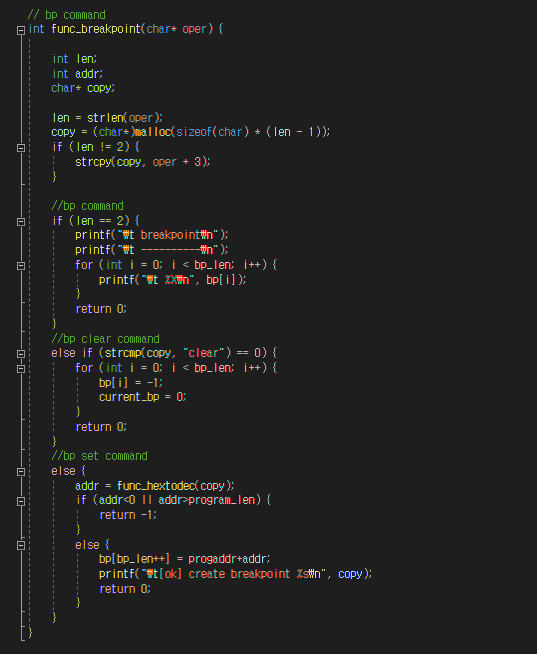
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_breakpoint \*/

/\*목적: 중단점에 관련된 명령을 수행한다\*/

/\*리턴 값: 오류가 있으면 -1을 오류가 없으면 0을 반환\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

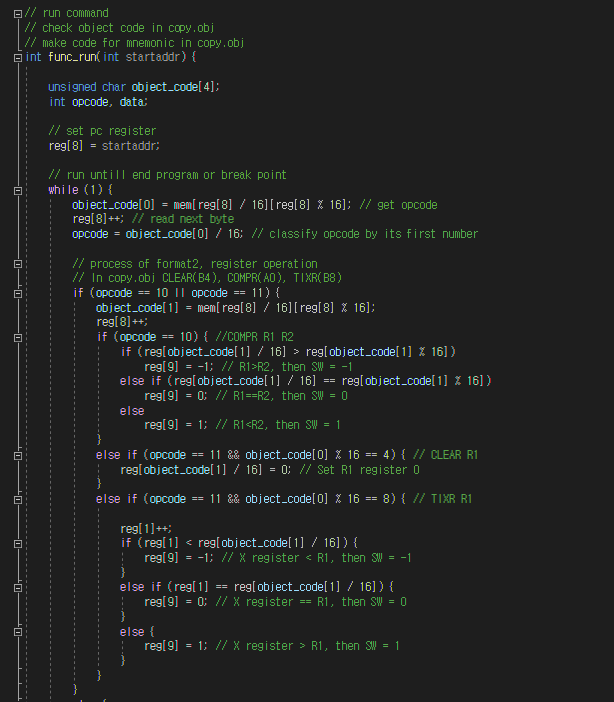
/\*함수: func\_run \*/

/\*목적: 가상 메모리에 로드한 copy.obj를 인자로 받은 startaddr

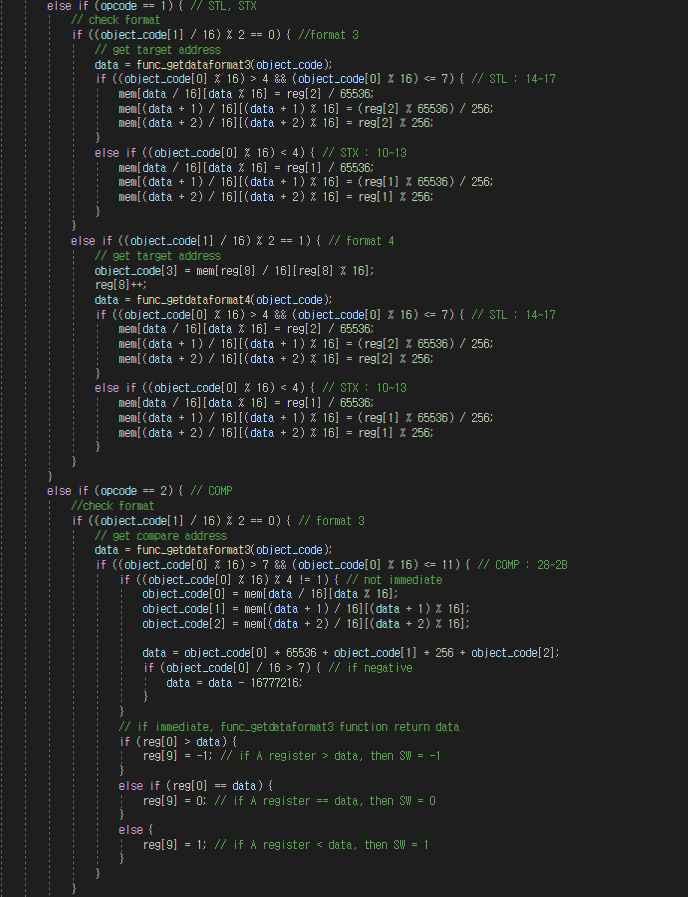
에서부터 실행한다\*/

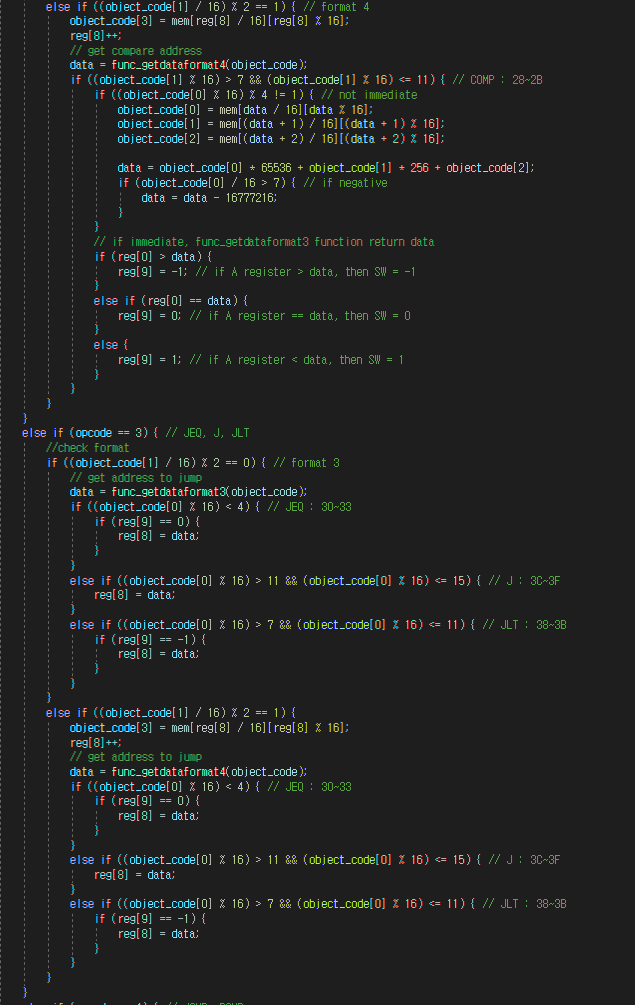
/\*리턴 값: 중단점에서 종료가 되었으면 1을

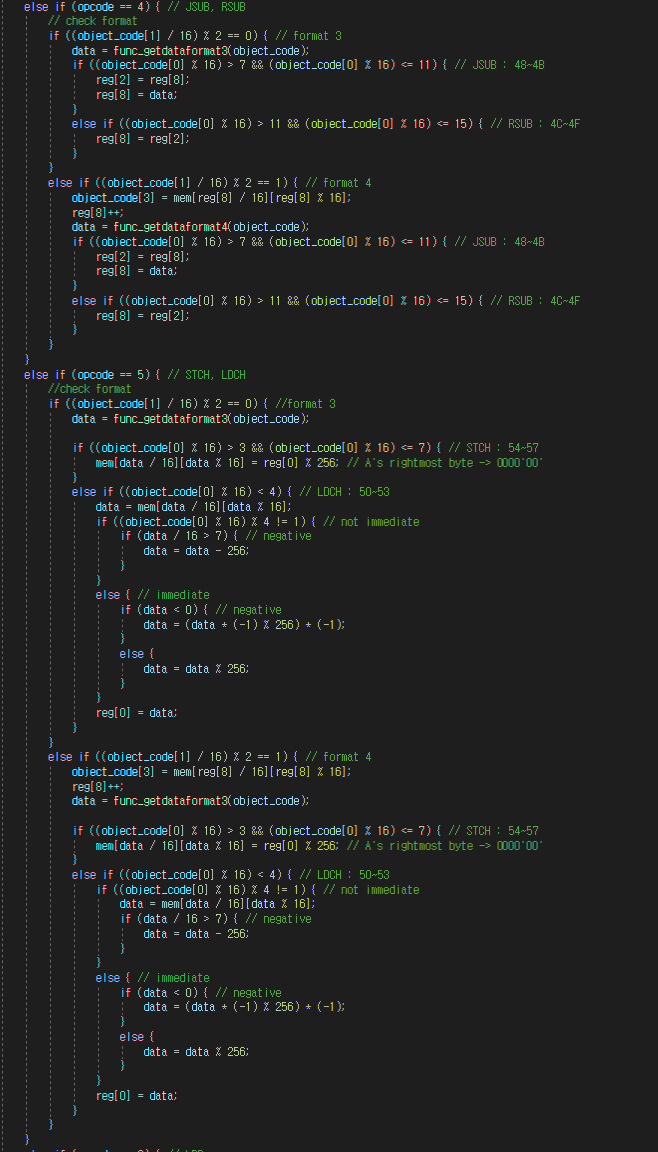
일반적으로 프로그램이 종료되면 0을 반환\*/

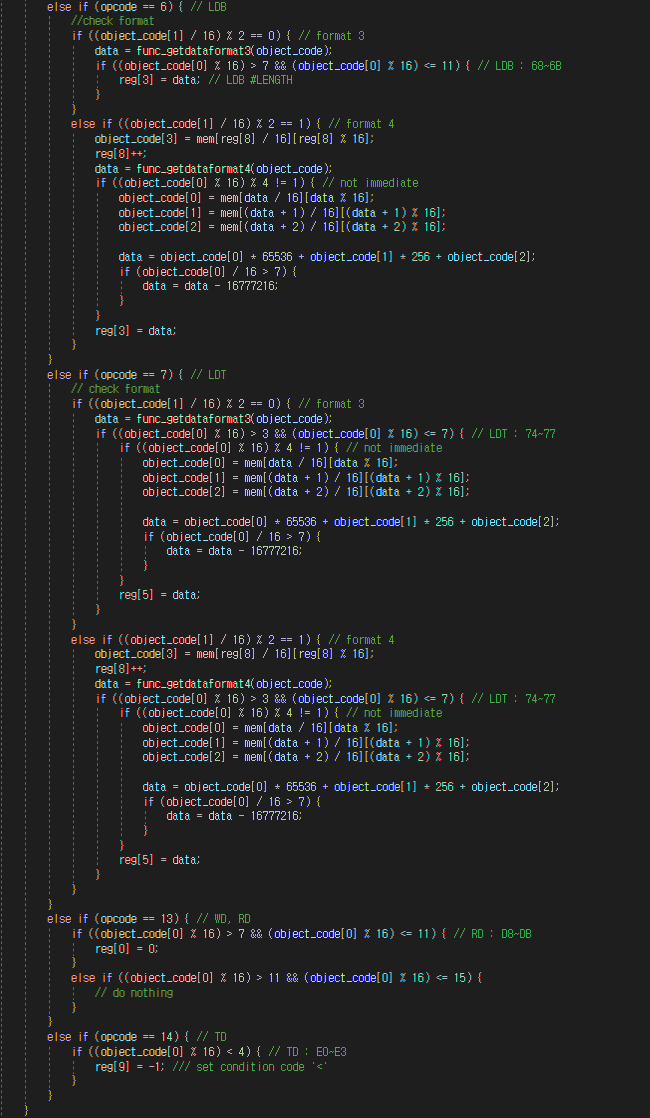


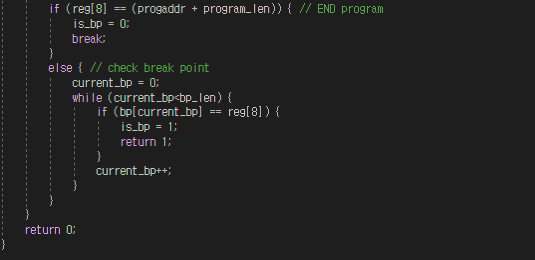












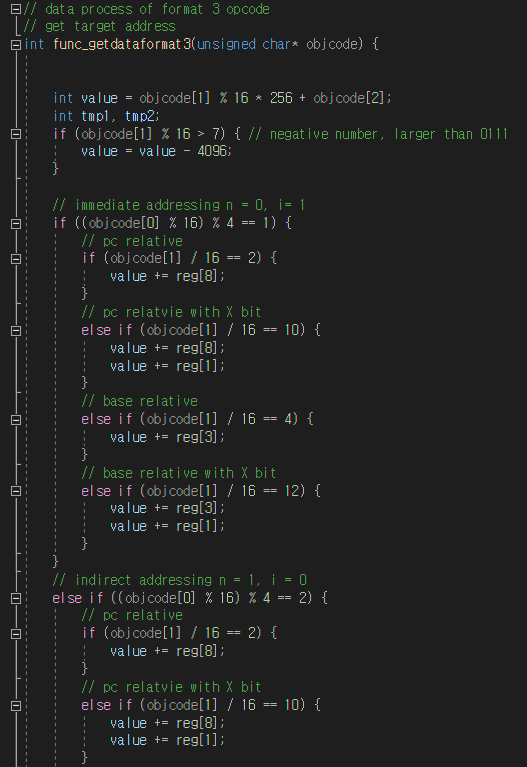
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

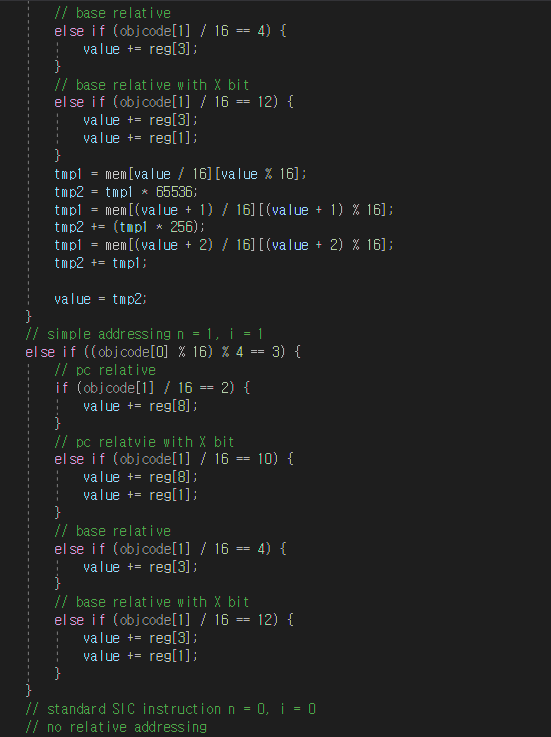
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

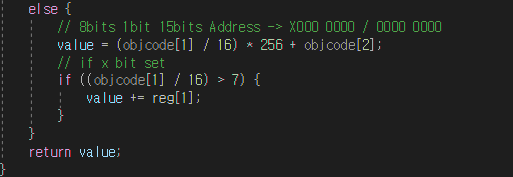
/\*함수: func\_getdataformat3\*/

/\*목적: 3형식 명령어의 TA를 구한다\*/

/\*리턴 값: 연산을 통해 얻은 Tatget address 값을 반환한다\*/







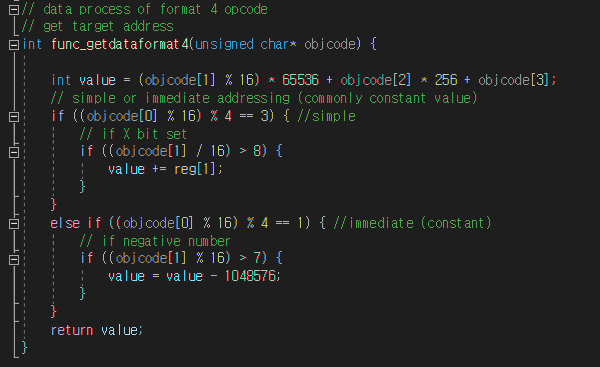
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_getdataformat4\*/

/\*목적: 4형식 명령어의 TA를 구한다\*/

/\*리턴 값: 연산을 통해 얻은 Tatget address 값을 반환한다\*/



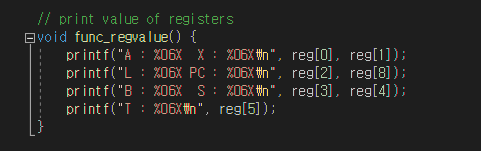
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*함수: func\_regvalue\*/

/\*목적: 실행이 종료되면 register 값을 출력한다\*/

/\*리턴 값: 없음\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 프로그램 작성은 전부 Cspro 환경에서 진행했고, Visual Studio 환경으로 코드를 옮겨와 캡쳐만 진행 하였습니다 \*/