**과목명: 시스템프로그래밍**

**2분반**

**<<Project #3>>**

**서강대학교 수학과**

**20171290**

**이승은**

목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. 프로그램 흐름도
3. **모듈 정의**
   1. Int loader\_pass1(char\* filename)
   2. Int loader\_pass2(char\* filename);
   3. void link\_to\_estab(estab\* newestab)
   4. void link\_to\_refer(refer\* newrefer);
   5. void free\_refer()
   6. void print\_estab()
   7. int check\_bound()
   8. estab\* find\_progname(char\* name)
   9. estab\* find\_progsymb(char\* name)
   10. refer\* find\_refer(char\* number)
   11. void free\_estab()
   12. void free\_bp()
   13. bpoint\* find\_bpoint(int number)
   14. void print\_bp()
   15. void print\_registers(Boolean end, int bp)
   16. op\_node\* find\_opinst(char\* hexa)
   17. void set\_register(int reg, int value)
   18. int get\_register(int reg)
   19. int progaddr\_func(int index)
   20. int loader\_func(int index)
   21. void bp\_func()
   22. int run\_func()
4. **전역 변수 정의**
5. **코드 설명**

**1. 프로그램 개요**

이 프로그램은 SIC/XE의 assembly program source file를 이용하여 생성 된 object file을 이용하여 해당 프로그램을 link하고 프로그램에 load한다. 이 기능을 사용하기 위해서 다음과 같은 명령어가 사용된다.

1) progaddr address

프로그램의 시작주소를 지정하는 명령어로 16진수로 들어온 address의 값을 load할

프로그램의 시작주소로 설정한다.

2) loader filename1 filename2 filename3

Parameter로 들어온 최대 3개의 파일을 링킹하고, 또 메모리에 올리는 일을 수행하기

위한 명령어이다.

3) run

메모리에 load 된 프로그램을 수행하는 명령어이다.

4) bp loc

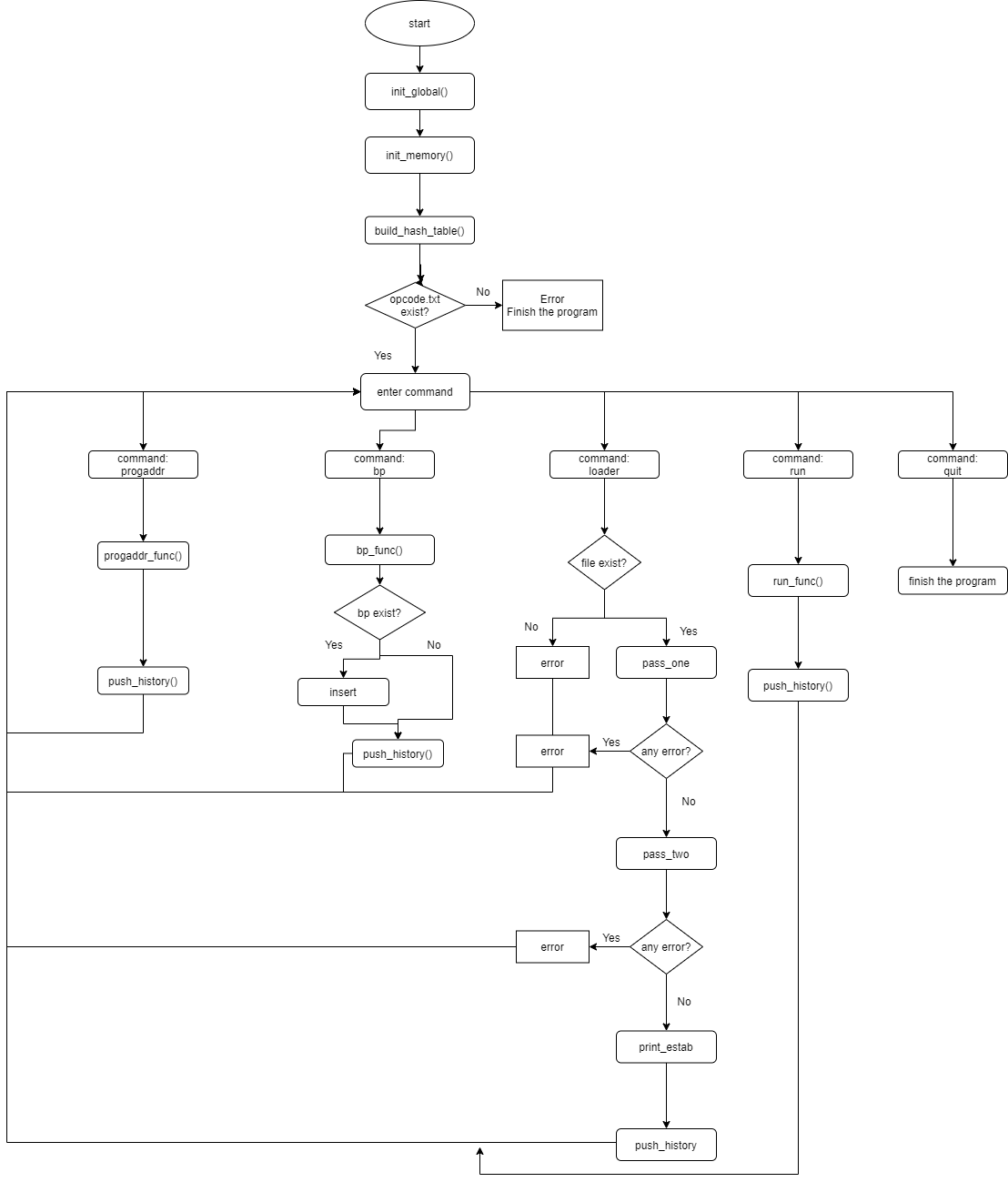
Debug 명령어로, 인자로 들어온 loc에 해당하는 부분에서 프로그램의 수행을 멈춘다.

그리고 해당 break point까지 수행됐을 때의 register의 값을 보여준다.

breakpoint에서 멈춘 후 다시 run을 수행하면, breakpoint 이후부터 수행이 된다.

**2. 프로그램 설명**

2.1 프로그램의 흐름도



Quit 명령어를 받을 때까지 프로그램은 계속 수행되며, 3차 프로젝트에 추가 된 내용만 추가적으로 흐름도를 작성하였다.

**3. 모듈 정의**

3.1 Int loader\_pass1(char\* filename)

입력: load할 filename

출력: NULL

사용변수: estab\* new: ESTAB에 추가 될 새로운 node

Before\_prog\_length: 현재 프로그램이나 symbol의 시작 주소를 구하기 위해서

사용되는 이전에 linking & load 된 프로그램의 길이

Arrend: 프로그램이 가상 메모리 범위를 벗어나서 load되는지 여부를 판단하기

위해서 사용된다.

기능: 함수의 인자로 들어오는 파일명의 파일을 읽어서, object code의 head record,

Define record를 해석하여 프로그램의 정보, 프로그램에서 사용되는 변수의 정보를

저장한다. 여기서의 정보는 프로그램의 길이, 시작주소, 변수 명 등이다.

3.2 Int loader\_pass2(char\* filename)

입력: memory에 load할 filename

출력: NULL

사용변수: estab\* prog: 현재 프로그램의 이름과, 길이 등의 정보를 가진 ESTAB node를

저장하는 변수

Refer\* new: reference symbol의 정보를 저장 할 node.

Int memnum: object code의 hexadecimal string이 integer의 값으로 변환된

값을 임시로 저장하는 변수

Int mem[][]: 가상 메모리를 표현하는 변수

Int bytenum: relocation이 필요한 byte의 수

Char mNumber[2]: reference number를 저장하는 변수

Refer\* estabSYM: reference number에 해당되는 symbol의 이름 저장한다.

Estab\* set: estabSYM으로 구한 symbol의 이름으로 해당 symbol의 정보를 저장

하는 변수

Int execution\_addr: program의 execution start address의 value를 저장한다.

기능: 인자로 들어온 program의 object file을 읽으면서 memory에 object code를 직접 load

한다. 그리고 relocation이 필요한 부분은 직접 relocation을 해주며 수행 메모리에

수행 가능한 program을 load 해 준다.

3.3 void link\_to\_estab(estab\* newestab)

입력: ESTAB linked list에 새로 추가 될 node

출력: NULL

사용변수: estab\* estab\_head: ESTAB linked list의 head를 가리키는 포인터

Estab\* estab\_tail: ESTAB linked list의 tail을 가리키는 포인터

기능: 인자로 들어온 estab node를 ESTAB linked list에 추가 해 준다.

3.4 void link\_to\_refer(refer\* newrefer)

입력: reference linked list에 새로 추가 될 node

출력: NULL

사용변수: refer\* refer\_head: reference linked list의 head를 가리키는 포인터

Refer\* refer\_tail: reference linked list의 tail을 가리키는 포인터

기능: 인자로 들어온 refer node를 reference linked list에 추가 해 준다.

3.5 void free\_refer()

입력: NULL

출력: NULL

사용변수: refer\* refer\_head: reference linked list의 head를 가리키는 포인터

Refer\* refer\_tail: reference linked list의 tail을 가리키는 포인터

기능: reference linked list에 있는 모든 node를 메모리 해제 해 준다.

3.6 void print\_estab()

입력: NULL

출력: linking과 loading이 진행된 뒤의 ESTAB

사용변수: estab\* estab\_head: ESTAB의 head를 가리키는 포인터

Int A, B, X, S, T, PC, L: SIC/XE에서 사용되는 register 값을 저장하는 변수

기능: linking과 loading 작업이 끝나면 해당 작업 이후에 생성되는 ESTAB을 출력한다.

그리고 program을 수행하기 위해서 register의 값을 모두 초기화 해 준다.

3.7 int check\_bound()

입력: NULL

출력: NULL

사용변수: estab\* head: ESTAB의 head를 가리키는 포인터

Int progaddr: program의 시작 주소를 저장해 놓은 변수

Int totalLength: 프로그램의 길이를 저장하는 변수

기능: 저장 된 ESTAB를 이용하여 프로그램의 시작주소와 길이를 합한 후, 프로그램이 load

되는 과정에서 가상메모리의 범위를 넘어서면 오류를 return한다.

3.8 estab\* find\_progname(char\* name)

입력: program name

출력: NULL

사용변수: estab\* estab\_head: ESTAB의 head를 가리키는 포인터

기능: ESTAB linked list를 순회하면서 parameter로 들어 온 이름의 프로그램 명이 있는지

확인하고, 있다면 해당 estab node를 return한다.

3.9 estab\* find\_progsymb(char\* name)

입력: symbol name

출력: NULL

사용변수: estab\* estab\_head: ESTAB의 head를 가리키는 포인터

기능: ESTAB linked list를 순회하면서 parameter로 들어 온 이름의 symbol 명이 있는지

확인하고, 있다면 해당 estab node를 return한다.

3.10 refer\* find\_refer(char\* number)

입력: reference number

출력: NULL

사용변수: refer\* refer\_head: reference linked list의 head를 가리키는 포인터

기능: reference linked list를 순회하면서 parameter로 들어 온 reference number에 대응

되는 symbol이 있는지 확인하고, 있다면 해당 refer node를 return한다.

3.11 void free\_estab()

입력: NULL

출력: NULL

사용변수: estab\* estab\_head: ESTAB의 head를 가리키는 포인터

기능: ESTAB linked list를 구성하는 모든 node의 메모리를 해제 해 준다.

3.12 void free\_bp()

입력: NULL

출력: NULL

사용변수: bpoint\* bpoint\_head: breakpoints linked list의 head를 가리키는 포인터

기능: breakpoint linked list를 구성하는 모든 node의 메모리를 해제 해 준다.

3.13 bpoint\* find\_bpoint(int number)

입력: PC의 값

출력: NULL

사용변수: bpoint\* bpoint\_head: breakpoint linked list의 head를 가리키는 포인터

기능: parameter로 들어온 PC의 값이 breakpoint인지 여부를 확인하고, 만약 breakpoint

라면 해당 bpoint node를 return 해 준다.

3.14 void print\_bp()

입력: NULL

출력: 저장 된 breakpoints들을 출력한다.

사용변수: bpoint\* bpoint\_head: breakpoint linked list의 head를 가리키는 포인터

기능: breakpoint linked list를 순회하면서 저장 되어있는 breakpoint들의 값을 모두

출력한다.

3.15 void print\_registers(Boolean end, int bp)

입력: 프로그램이 멈춘 break point 값과, 프로그램이 다 수행된 것인지 여부를 알려주는

Boolean 값

출력: 각 register의 값

사용변수: int A, X, L, PC, B, S, T: SIC/XE machine에서 사용되는 register

기능: 현재까지 프로그램이 수행 된 뒤의 register값을 16진수로 출력한다.

3.16 op\_node\* find\_opinst(char\* hexa)

입력: instruction을 알고자 하는 opcode

출력: NULL

사용변수: hash\_table: hash table의 형태로 opcode를 저장 해 놓은 변수

Boolean findInst: instruction을 찾았는지 여부를 저장하는 변수

기능: opcode가 저장 된 hash\_table를 순회하면서, parameter로 들어온 opcode에 해당되는

Instruction이 있다면, 해당 node를 return 해 준다.

3.17 void set\_register(int reg, int value)

입력: 값을 setting 하고자 하는 register와 setting 하는 값

출력: NULL

사용변수: int A, X, L, PC, B, S, T: SIC/XE machine에서 사용되는 register

기능: parameter로 들어오는 register number에 따라서 해당 register의 값을 value로

바꿔준다.

3.18 int get\_register(int reg)

입력: 값을 알고자 하는 register

출력: NULL

사용변수: int A, X, L, PC, B, S, T: SIC/XE machine에서 사용되는 register

기능: parameter의 값에 해당하는 register의 값을 return한다.

3.19 int progaddr\_func(int index)

입력: command창에 입력 된 string의 size

출력: NULL

사용변수: int before\_addr: 이전에 setting 되어있던 program 시작 주소를 저장하는 변수

Int progaddr: program의 시작주소를 저장하는 변수

Addrend: 가상메모리의 크기를 나타내는 변수

Boolean parameter\_error: address의 값이 올바른 16진수인지 여부를 저장하는

변수

기능: command 창에 입력 된 16진수를 parsing하여, 해당 값을 program address(program

의 시작주소)로 설정 해 준다. 만약, 이 주소 값이 잘못 된 값이라면, program

Address의 값을 이전에 저장되어있던 값으로 복구시킨다.

3.20 int loader\_func(int index)

입력: command창에 입력 된 string의 size

출력: NULL

사용변수: int res: 각 load pass가 끝날 때마다, 해당 과정이 성공적으로 수행 되었는지

결과를 저장하는 변수

Char filename[]: file명을 저장하는 변수

기능: command로 입력 받은 file을 link하고 메모리에 load한다. 또한, loading까지 성공

적으로 수행되면 ESTAB을 출력하는 함수를 호출한다.

3.21 void bp\_func()

입력: NULL

출력: NULL

사용변수: char sentence[]: 입력되는 command를 저장하는 변수

Int bp: string hexadecimal의 breakpoint의 값이 integer로 변환 된 값을 저장하는

변수

Boolean parameter\_error: breakpoints의 값이 올바른 16진수의 값인지 여부를

저장하는 변수

기능: command로 들어온 breakpoint의 값을 parsing하여 breakpoint linked list에

추가한다. 이 때, 이미 추가 된 breakpoint가 있다면, 더 추가하지 않는다.

3.22 int run\_func()  
 입력: NULL

출력: NULL

사용변수: char inst[]: object code에서 instruction에 해당하는 값을 저장하는 변수

Int loc: 현재 수행되는 주소를 가리키는 변수

Int A, X, L, PC, B, S, T: SIC/XE machine에서 사용되는 register

Op\_node\* opcode: inst에 해당되는 instruction의 정보를 저장하는 변수

Bpoint\* findbp: 현재의 PC가 breakpoint인지 여부를 저장하는 변수

Int reg1, reg2: format 2에서 사용된 register를 나타내는 변수

Int nb, ib, xb, bb, pb, eb: format 3, 4에서 nixbpe bit의 정보를 저장하는 변수

Int x, y: 접근하는 메모리의 행과 열의 정보를 가진 변수

Int compare: comparison 연산의 결과를 저장하는 변수

Boolean endflag: break point에 의해 수행이 중단되었음을 명시하는 변수

Int data: display값으로 얻는 value, 또는 immediate 연산으로 얻는 value

Int address: 접근해야 하는 메모리의 주소

Int totalLength: 수행하는 program의 길이를 나타내는 변수

기능: 현재 메모리에 load되어있는 프로그램을 수행한다. 그리고 수행하는 line이 breakpoint

라면, 프로그램을 계속해서 수행하지 않고, 중단한 뒤 현재까지의 register의 값을 보여

주는 함수를 수행한다. 그리고 프로그램이 다 수행되었다면, 프로그램이 다 수행된

뒤의 register값을 출력 해 주는 함수를 호출한다.

**4. 전역 변수 정의**

(프로젝트 2이후에 추가 된 전역변수)

Typedef struct estab{

Char control[10];

Char symbol[10];

Int address;

Int length;

Struct estab\* ptr;

}

-ESTAB을 구성하는 node로, control의 정보를 저장하는 경우, control의 이름, 주소, 길이 정보를, symbol의 정보를 저장하는 경우 symbol의 이름, 주소의 값을 저장한다.

Typedef struct refer{

Char number[3];

Char name[10];

Struct refer\* ptr;

}

-reference linked list를 구성하는 node로, object file에서 reference number를 읽어와서, reference number에 해당되는 symbol name을 저장한다.

Typedef struct bpoint{

Int breakpoint;

Struct bpoint\* ptr;

}

-breakpoints linked list를 구성하는 node로, 입력 된 breakpoint의 정보를 저장한다.

Int progaddr: program의 start address를 저장하는 변수이다.

Int before\_prog\_length: 전에 link한 program들의 length합을 저장하는 변수다.

Int current\_prog\_addr: 현재 program의 length를 저장하는 변수

Int execution\_addr: program의 execution address를 저장하는 변수

Estab\* estab\_head: ESTAB의 head를 가리키는 포인터

Estab\* estab\_tail: ESTAB의 tail을 가리키는 포인터

Refer\* refer\_head: reference linked list의 head를 가리키는 포인터

Refer\* refer\_tail: reference linked list의 tail를 가리키는 포인터

Bpoint\* bpoint\_head: breakpoint linked list의 head를 가리키는 포인터

Bpoint\* bpoint\_tail: breakpoint linked list의 tail을 가리키는 포인터

Int totalLength: 현재 프로그램의 총 길이를 저장하는 변수

Int A, X, L, PC, B, S, T: SIC/XE machine에서 사용되는 register의 값을 저장하는 변수

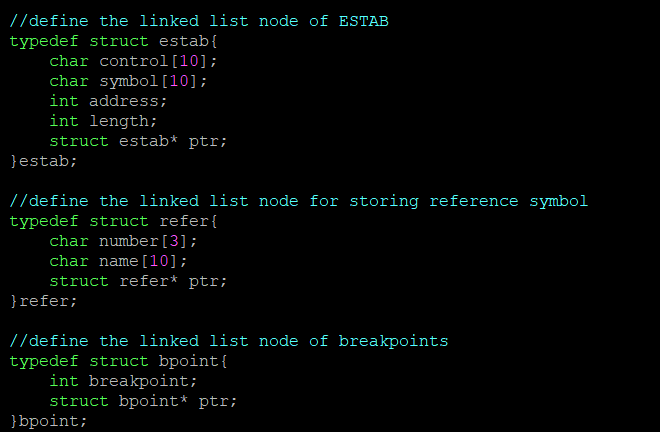
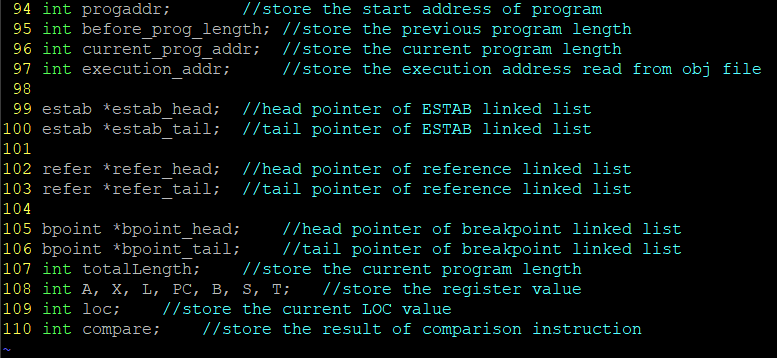
Int loc: 현재 수행중인 프로그램의 주소를 가리키는 변수

Int compare: comparison instruction이 수행 된 결과를 저장하는 변수

**5. 코드설명**

20171290.h : 프로그램 내에서 사용하는 전역변수들을 저장 해 놓은 header file

3차 프로젝트에서 필요한 전역변수를 추가로 선언하였다.

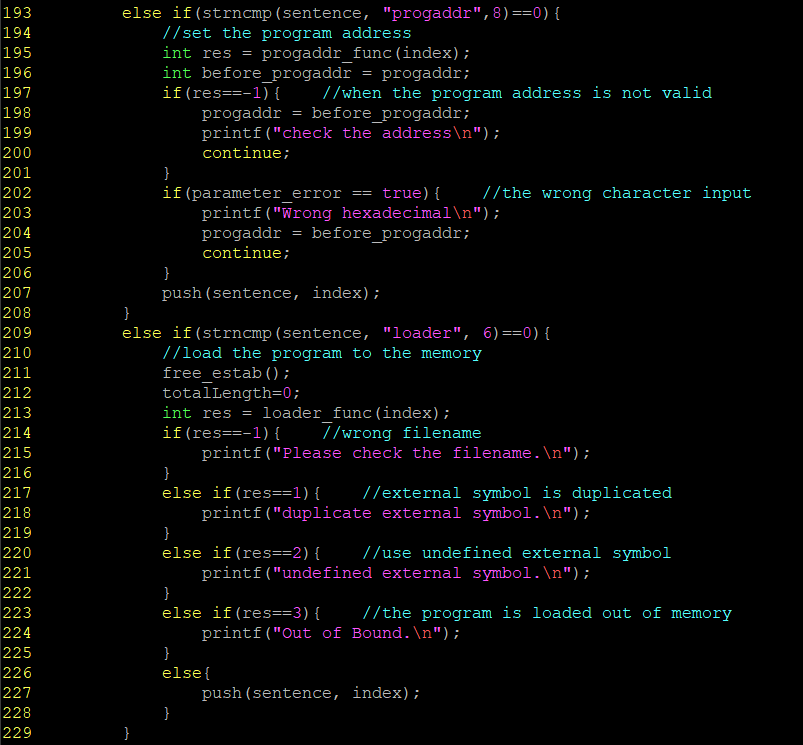
 

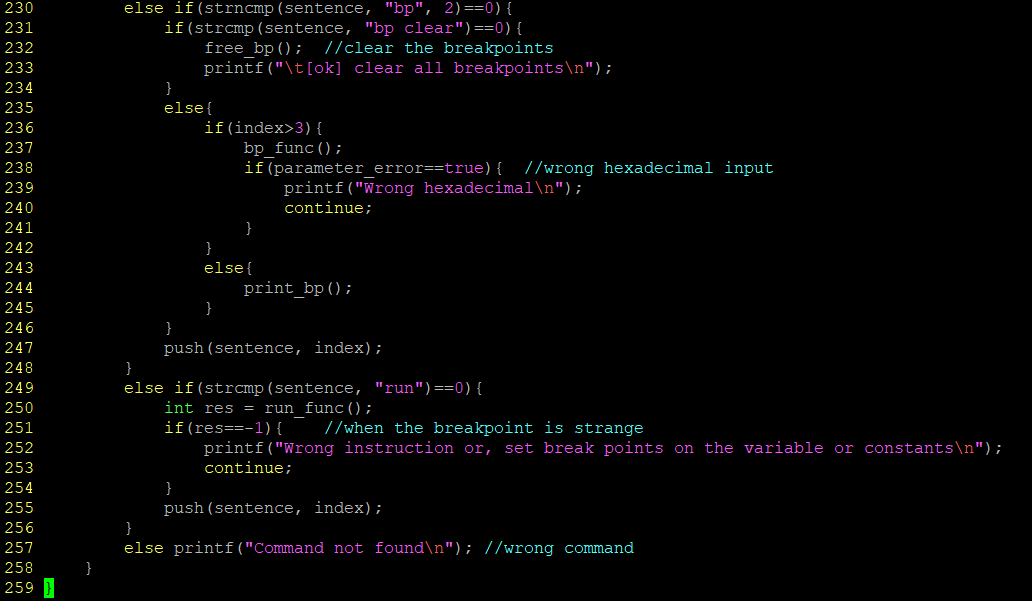
20171290.c: command로 들어 온 명령어를 읽어 알맞은 함수를 수행하도록 하는 코드

2차 프로젝트에서 필요한 코드를 추가하였다. main함수는 1,2주차 프로젝트에서

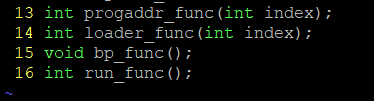
구현 된 명령어 외에 progaddr, loader, bp, run 명령어를 추가적으로 인식하여

각 명령에 맞게 필요한 기능을 수행하도록, 해당 기능과 관련 된 함수들을 호출한다.

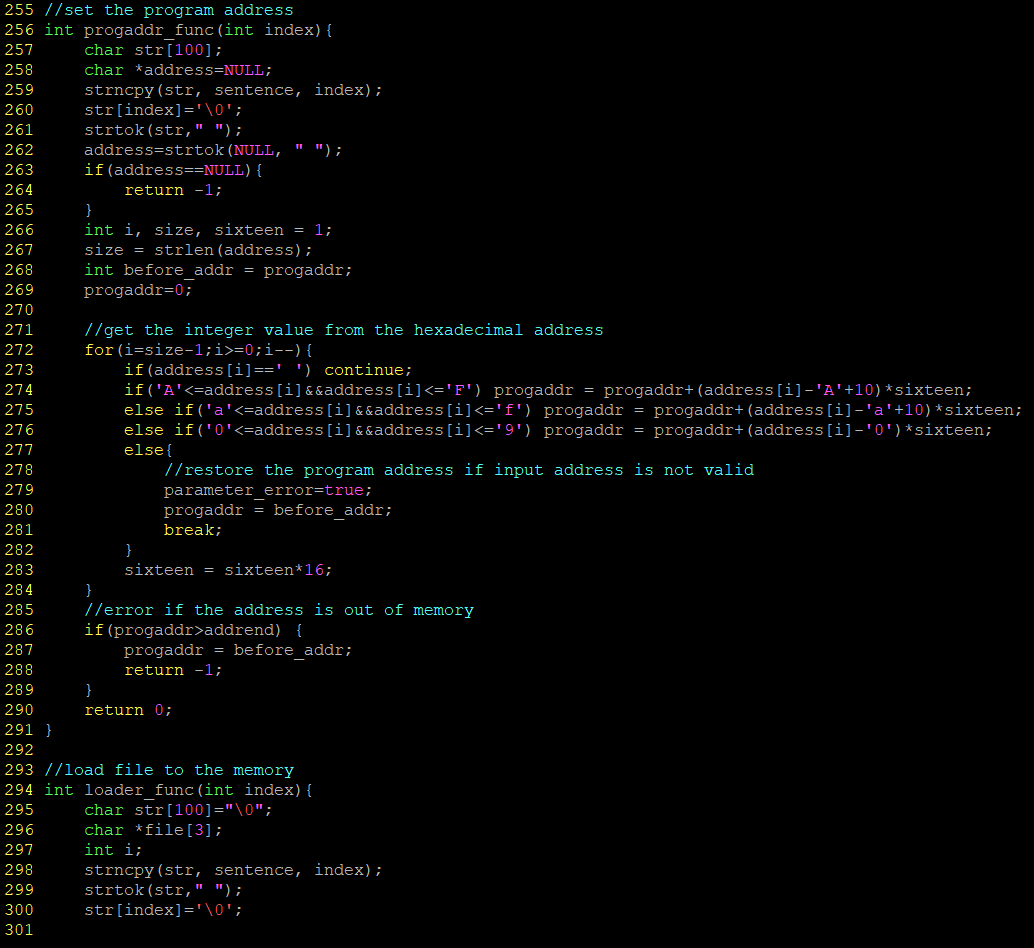
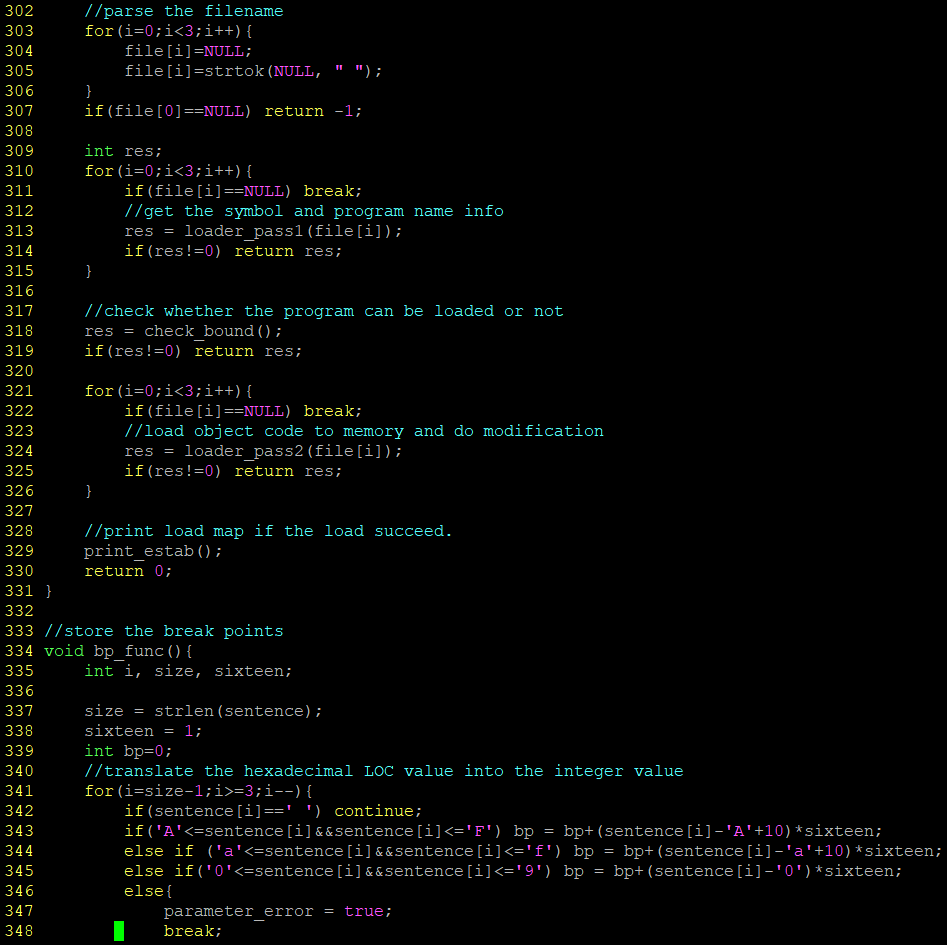
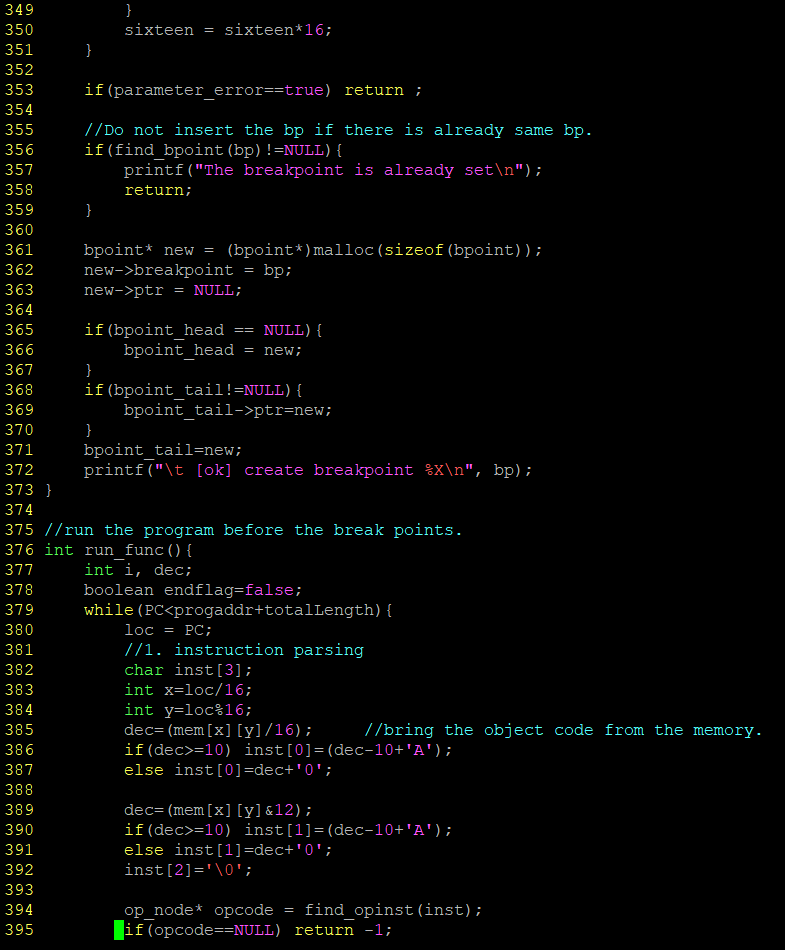
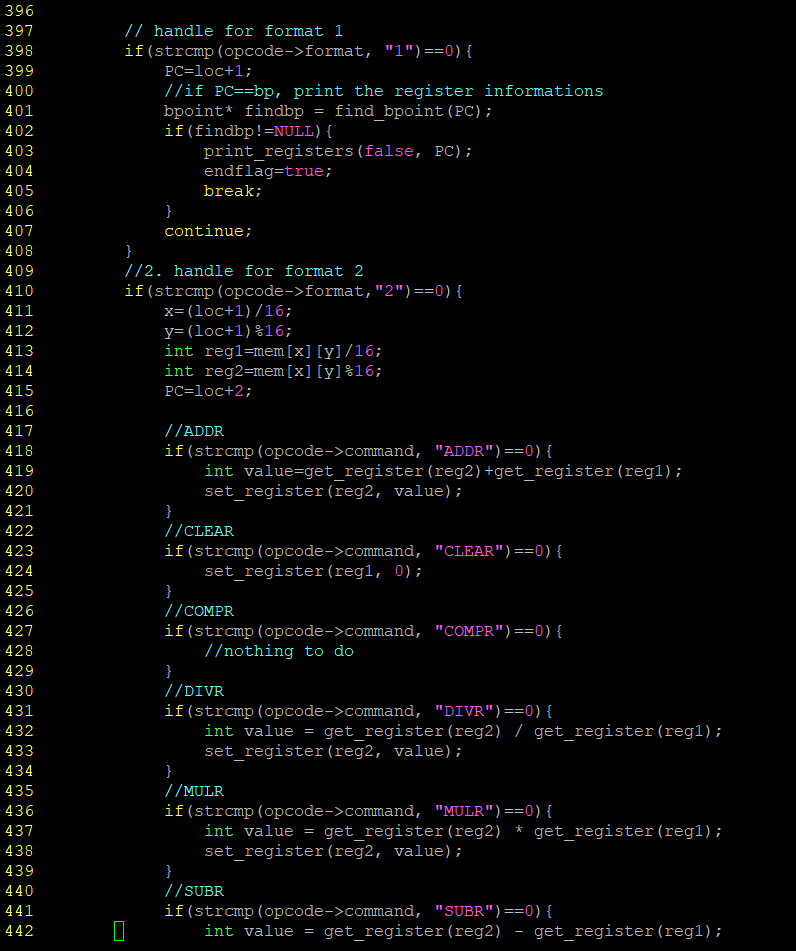
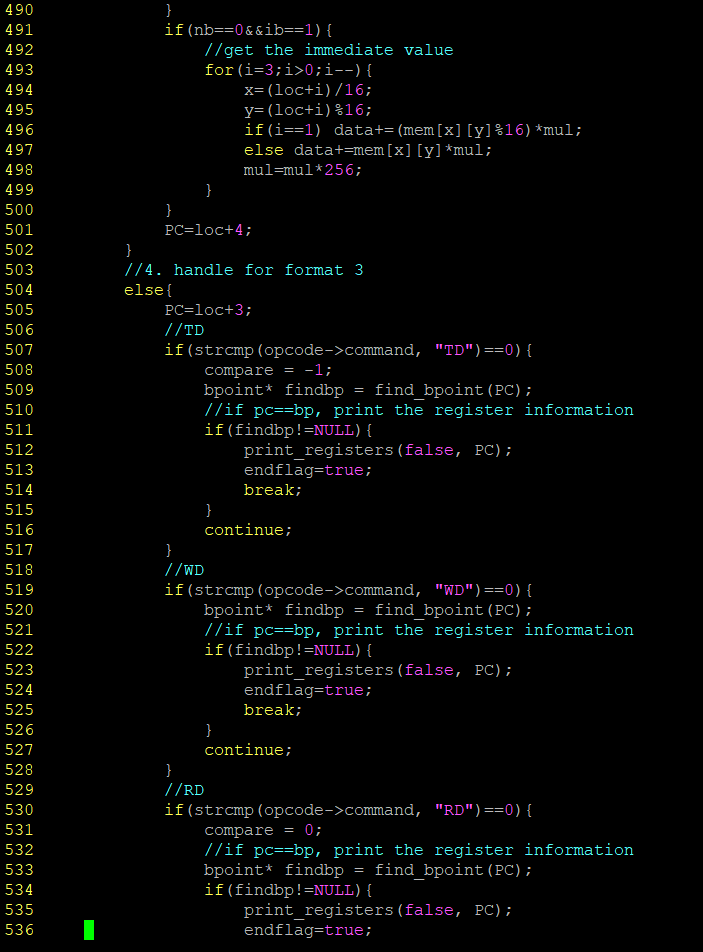
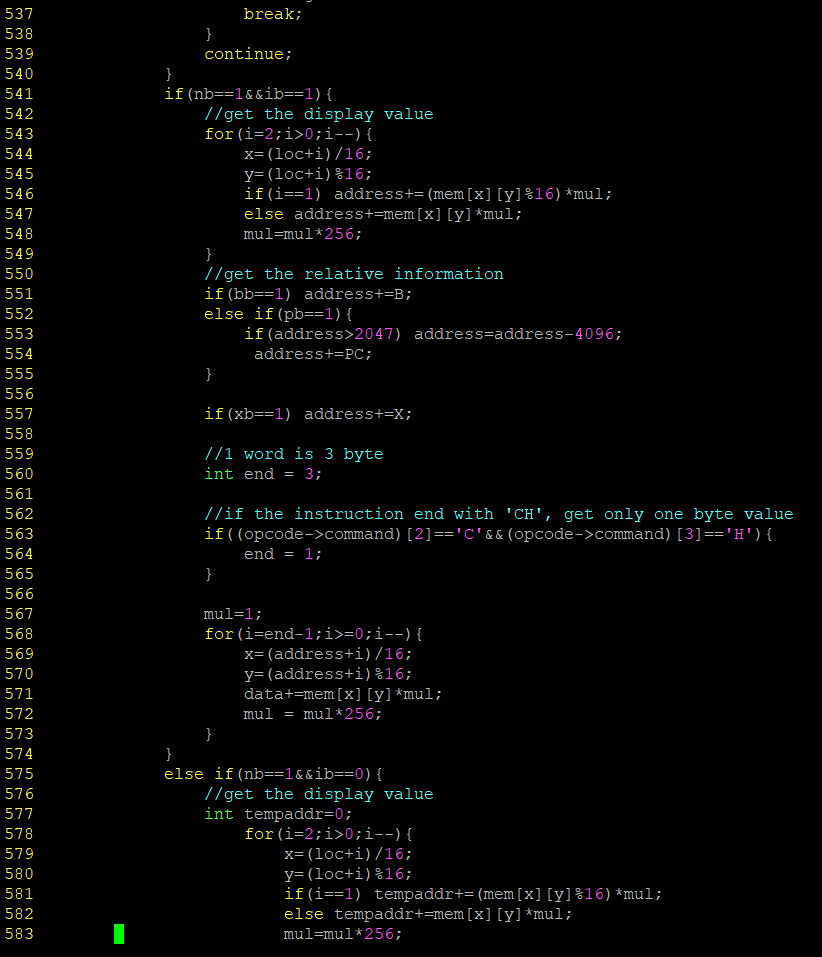
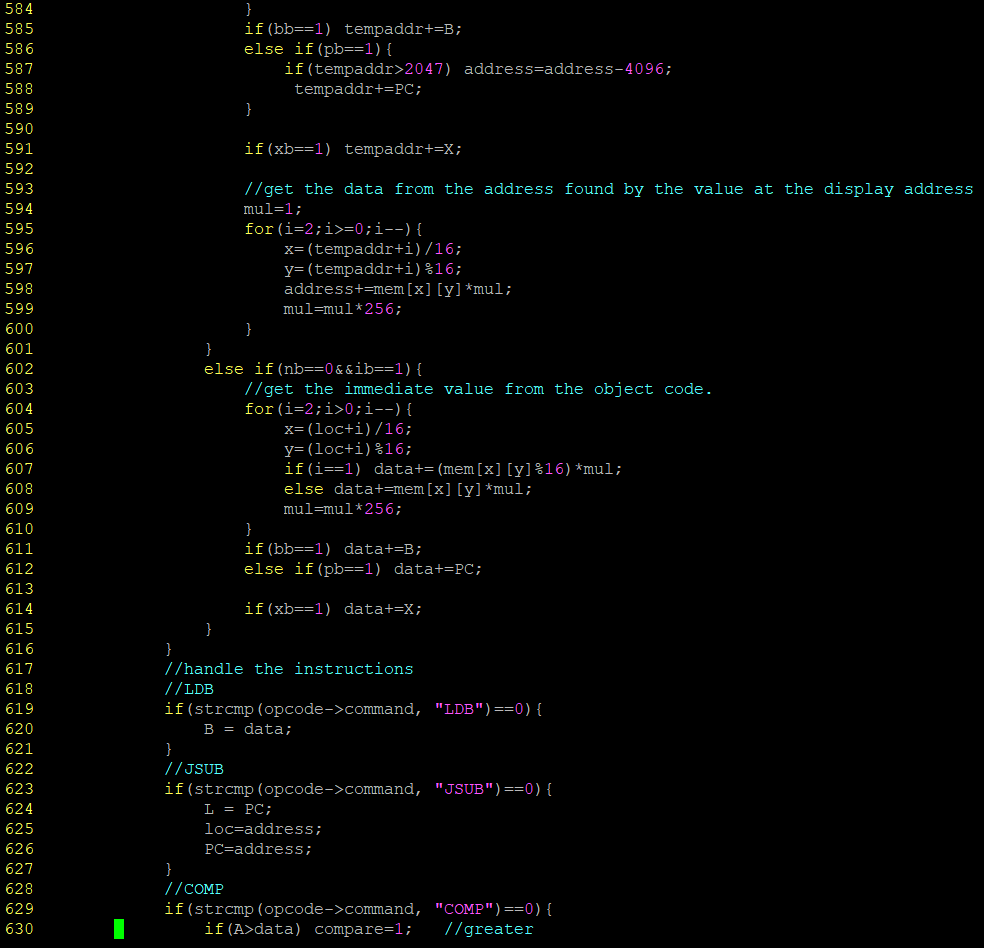
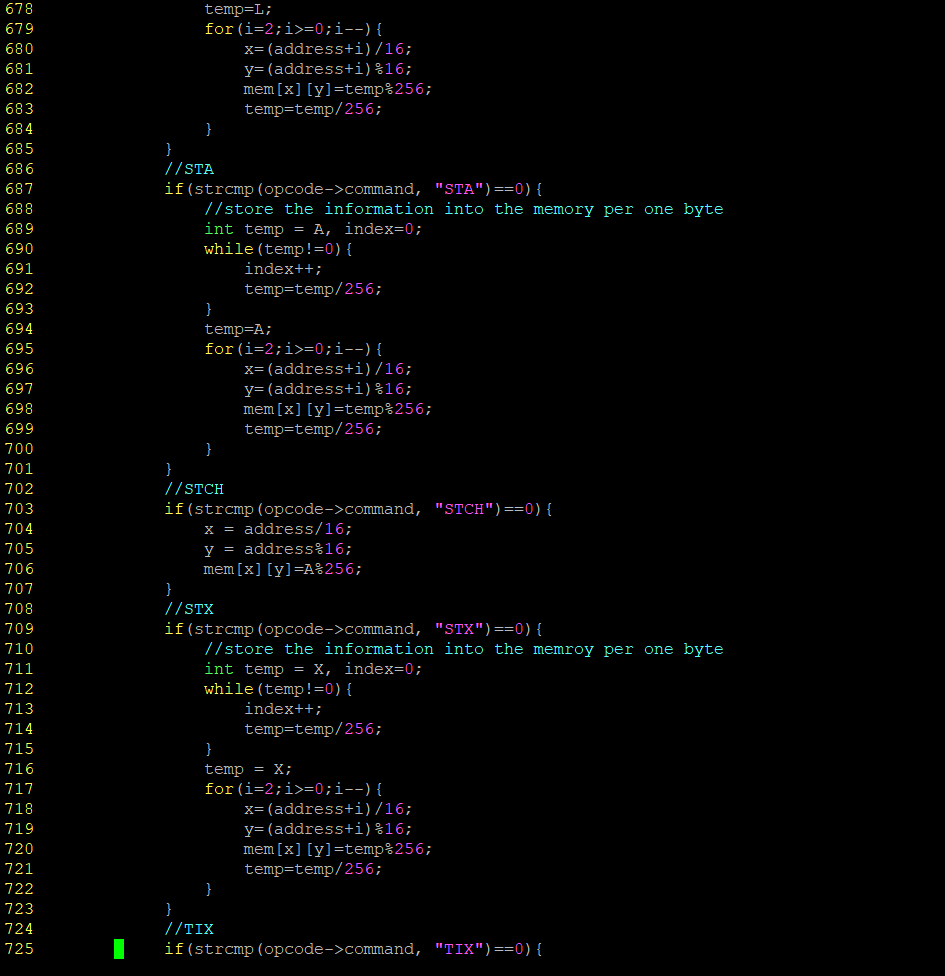
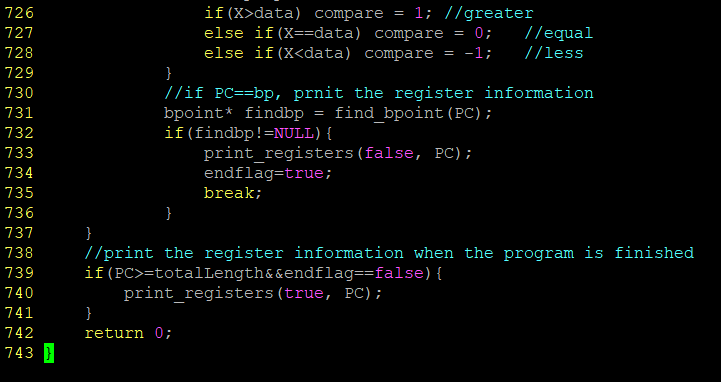




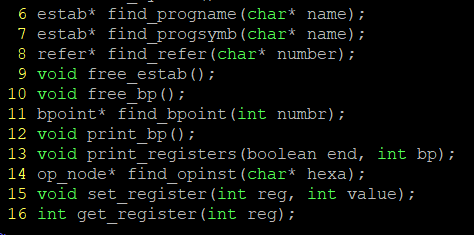
Command.h : command로 들어온 명령어에 맞추어 수행되는 함수들의 prototype을 모아놓은 header file이다. 이번 프로젝트에서 progaddr, loader, bp, run명령어를 추가적으로 구현하였으므로, 해당 명령어가 들어 올 때 수행되는 함수의 prototype을 추가하였다.



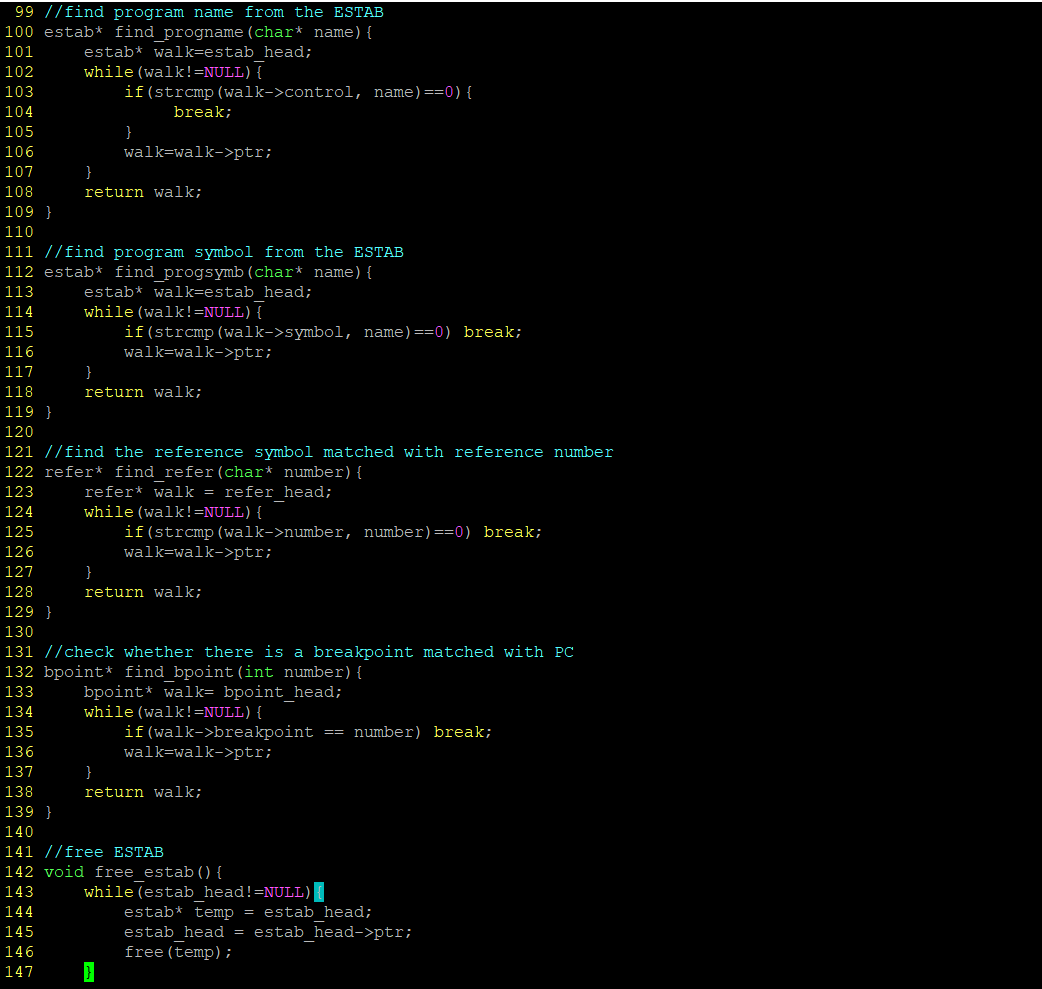
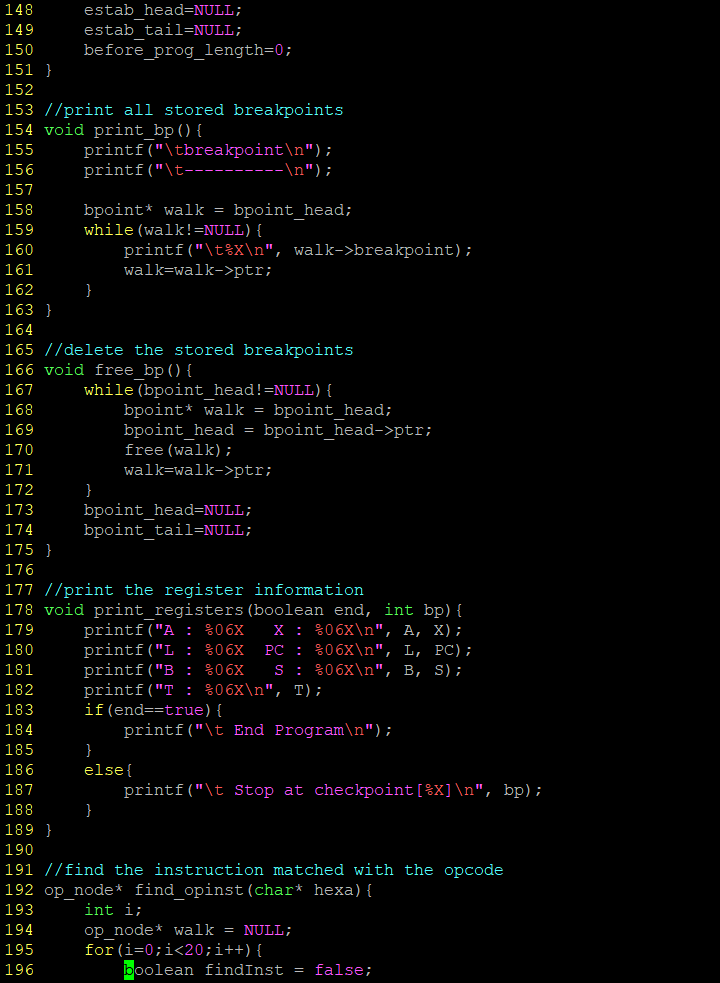
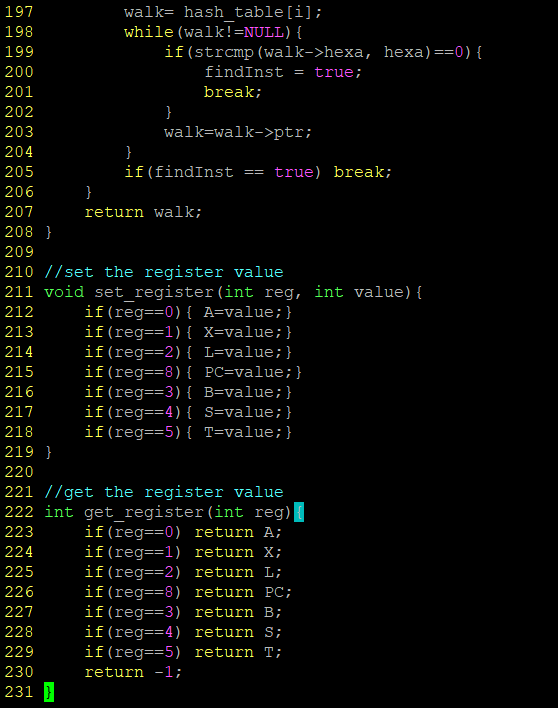
Command.c: command로 들어온 명령어에 맞추어 수행되는 함수들이 구현된 파일이다. Progaddr\_func은 프로그램의 시작주소인 progaddr의 값을 설정해 주는 일을 수행한다. Loader\_func은 파라미터로 들어온 프로그램들을 link하고 메모리에 load하는 일을 수행한다. Bp\_func은 파라미터로 들어온 값을 breakpoint로 저장하는 역할을 수행한다. Run\_func은 메모리에 load 된 프로그램을 수행한다. 그리고 한 프로그램의 수행이 완료되면 run\_func이 종료된다.

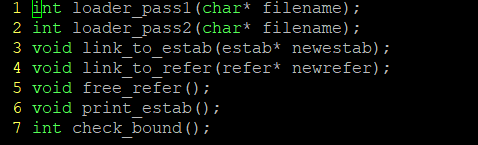
Subfunc.h : SIC/XE machine이 기능을 수행하는데 있어서 필요한 부가적인 함수의 prototype을 가진 파일이다. 프로젝트 3에서는 linker와 loader를 구현하기 위해서 추가적으로 11개의 함수를 구현하였다.



Subfunc.c : SIC/XE machine 기능을 수행하는데 있어서 필요한 부가적인 함수를 구현 한 파일이다. Find\_progname은 parameter로 들어온 program의 정보를 반환하는 함수, find\_progsymb는 parameter로 들어온 symbol의 정보를 반환하는 함수, find\_refer는 parameter로 들어온 reference number의 정보를 반환하는 함수이다. 그리고 프로그램이 수행된 뒤에 메모리를 해제하는 free\_estab, free\_bp 함수가 구현되어있다. 그리고 find\_bpoint는 parameter로 들어온 breakpoint에 관한 정보를 반환하는 함수이다. 또한, breakpoint나 register를 출력하는 print\_bp, print\_registers가 구현되었으며, parameter로 들어온 opcode에 대응되는 instruction을 반환하는 find\_opinst함수가 구현되었다. 그리고 마지막으로 register의 정보를 가져오거나, register의 값을 setting하는 set\_register, get\_register가 구현되었다.

Loader.h: loader와 linker를 구현하는데 필요한 함수의 prototype을 저장하는 파일이다.



Loader.c: linker와 loader를 구현하는데 사용되는 함수가 구현 된 파일이다.

Loader\_pass1에서는 parameter로 들어온 object file에서 Head record와 Definition record를 읽는다. Head record에서는 프로그램의 파일명, 프로그램의 시작 주소를 읽어 ESTAB에 저장한다. Definition record에서는 프로그램 내에서 정의 된 변수 명과, 해당 변수의 주소의 값을 읽어 ESTAB에 저장하는 과정을 거친다. 따라서, loader\_pass1의 함수가 끝나면 ESTAB이 완성되는 것이다. Loader\_pass2에서는 ESTAB와 object file을 이용하여 object code를 직접 메모리에 올리는 역할이 수행된다. 우선 Head record를 읽은 후, 프로그램 파일 명을 ESTAB에서 찾아 프로그램의 시작주소 등의 정보를 찾는다. 그 이후에는 reference record를 읽어 해당 프로그램에서 사용 된 reference symbol과 symbol의 reference number를 저장한다. 그리고 text record를 읽으면서 알맞은 메모리에 object code의 값을 저장한다. 그리고 modification record를 읽어 relocation이 필요한 코드를 relocate 해준다. 이 과정에서는 우선 relocation이 필요한 시작 주소의 값을 찾고, relocation이 필요한 메모리의 값에 reference symbol의 주소를 더해주거나 빼주면서 relocation을 진행한다. 마지막으로 end record를 읽어 프로그램의 수행 address의 정보를 저장한다. Link\_to\_estab는 ESTAB에 정보를 추가하는 함수, link\_to\_refer는 reference linked list에 정보를 추가하는 함수, free\_refer는 reference linked list의 메모리를 해제하는 함수, print\_estab는 ESTAB을 출력하는 함수, check\_bound는 프로그램이 loading되면서 메모리를 초과하지 않았는지 여부를 확인하는 함수이다.

