**과목명: 시스템프로그래밍**

**2분반**

**<<Project #1>>**

**서강대학교 수학과**

**20171290**

**이승은**

목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. 프로그램 흐름도
3. **모듈 정의**
   1. Init\_memory()
   2. Help\_func()
   3. Dir\_func()
   4. Quit\_func()
   5. History\_func()
   6. Dump\_func()
   7. Edit\_func()
   8. Fill\_func()
   9. Opcode\_list\_func()
   10. Opcode\_func(char\* command)
   11. Dump\_slice\_str(int index)
   12. Edit\_slice\_str(int index)
   13. Fill\_slice\_str(int index)
   14. Build\_hash\_table()
   15. Push(char\* str, int index)
   16. Subcmp(int start, int end, char\* str1, char\*str2)
   17. Init\_global\_variable()
   18. Main()
4. **전역 변수 정의**
5. **코드 설명**

**1. 프로그램 개요**

이 프로그램은 SIC/XE 구현 전 단계 프로그램으로, 어셈블러, 로더, 링크 등을 실행하게 될 셸(shell)과 object code가 적재되고 실행 될 메모리공간과 mnemonic (ADD, COMP, FLOAT, etc)를 opcode값으로 변환하는 opcode테이블과 관련 명령어들을 수행할 수 있도록 구현 된 프로그램이다. 이 프로그램은 shell과 메모리공간, 그리고 opcode table로 구성되어있으며, 각각 관련 된 명령어들은 다음과 같이 정리된다.

` 1) shell 관련 명령어

(1) h[elp]

shell에서 실행 가능한 모든 명령어들의 리스트를 화면에 출력 해 준다.

(2) d[ir]

현재 directory에 있는 파일들을 출력 해 준다.

이 때, 폴더의 경우 ‘파일명/’, 실행파일의 경우 ‘파일명\*’의 형태로 출력 해 준다.

(3) q[uit]

Shell을 종료한다.

(4) hi[story]

현재까지 사용한 명령어들을 순서대로 번호와 함께 보여준다.

이 때 가장 최근 사용한 명령어가 리스트의 가장 하단에 나온다.

2) memory 관련 명령어

(1) du[mp] [start, end]

할당되어있는 메모리의 내용을 주소/메모리값/아스키코드값 의 순서로 보여준다.

이 때, end인자가 없으면 start값으로부터 160개의 메모리를,

start와 end가 모두 있으면 start부터 end까지의 메모리를 보여준다.

아무런 인자 없이 dump만 입력되는 경우, 처음 dump가 입력될 때는 시작주소부터

160개의 메모리를 보여주고, 이후에 dump가 입력될 때는 저번 dump입력시 보여주었던

메모리의 마지막 부분 이후의 메모리부터 160개를 보여준다.

(2) e[dit] address, value

할당되어있는 메모리의 address값에 해당하는 주소의 값을 value로 바꿔준다.

(3) f[ill] start, end, value

할당되어있는 메모리의 start값부터 end값에 해당하는 주소의 값을 value로 바꿔준다.

(4) reset

할당되어있는 메모리의 모든 값을 0으로 변경시킨다.

3) opcode 관련 명령어

(1) opcode mnemonic

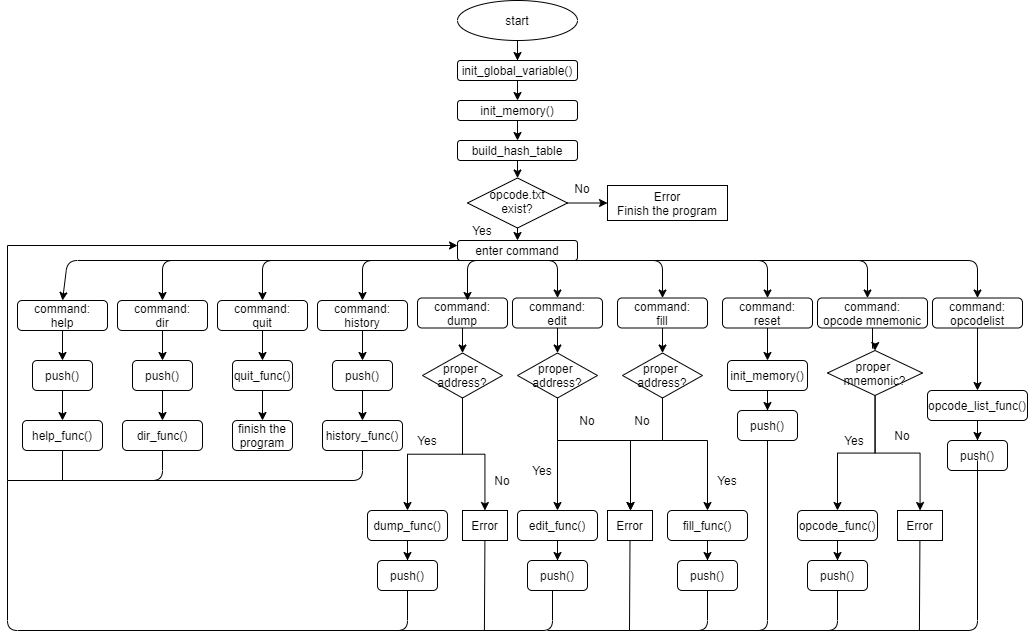
mnemonic(명령어)에 해당하는 opcode의 값을 opcode hash table에서 찾아서 출력한다.

(2) opcodelist

Opcode hash table의 내용을 [mnemonic, opcode]의 형식으로 모두 출력한다.

**2. 프로그램 설명**

**2.1 프로그램의 흐름도**



Quit를 command로 받기 전까지 command 입력을 계속 받는다.3. 모듈 정의

3.1 init\_memory()

입력: NULL

출력: NULL

사용변수: char mem[65536][16] – 프로그램에 할당 된 메모리를 의미하는 변수

기능: 프로그램에 할당 된 메모리 1Mbyte를 0으로 초기화한다.

3.2 help\_func()

입력: NULL

출력: program 명령문 출력

사용변수: NULL

기능: 해당 프로그램에서 사용 가능 한 명령어의 종류를 모두 출력한다.

3.3 dir\_func()

입력: NULL

출력: 현재 directory 내 file의 이름

사용변수: DIR \*directory – 탐색 할 directory의 정보를 저장한다.

Struct stat buf – file의 상태를 저장 해 놓는 buffer

Struct dirent \*file – directory에 있는 file의 정보를 저장한다.

기능: 현재 directory의 정보를 읽어 들여 현재 directory에 있는 file의 정보를 가지고 온다.

그리고 file의 이름을 모두 출력하는데, 실행파일은 이름 끝에 \*을, directory파일은

이름 끝에 /를 추가로 붙여준다.

3.4 quit\_func()

입력: NULL

출력: NULL

사용변수: str\_node \*history\_head – history linked list의 head로 history linked list의 정보를

위해 사용한다.

Str\_node \*temp – free 시켜 줄 history linked list node의 정보를 갖고 있다.

Op\_node \*hash\_table[20] – opcode hash table을 가지고 있는 변수

Op\_node \*walk – i번째 index에 해당하는 opcode hash table list의 정보를 위해

사용되는 변수

Op\_node \*temp - free시켜 줄 opcode hash table node의 정보를 갖고 있다.

기능: 프로그램에서 동적으로 할당 된 linked list의 메모리를 모두 해제 해 주는 역할을

한다. 동적으로 할당 된 linked list는 history linked list와 opcode hash table이 있다.

3.5 history\_func()

입력: NULL

출력: 지금까지 입력 되었던 command와 command가 입력 된 순서

사용변수: str\_node \*walk – 출력 할 history linked list node의 정보를 가지고 있는 포인터

Str\_node \*history\_head – 출력 할 history linked list의 head pointer

기능: 지금까지 입력 되었던 command의 정보를 저장하고 있는 history linked list를 순회

하면서 history linked list에 저장되어 있는 모든 node의 정보를 출력한다.

3.6 dump\_func(int startidx, int endidx)

입력: 시작 주소와 마지막 주소

출력: 시작 주소부터 마지막 주소까지에 해당되는 메모리의 주소를 16진수로, 해당되는

메모리의 내용을 16진수로, 해당되는 메모리 값에 대응하는 아스키코드의 값을

출력한다.

사용변수: int startx – 16\*65536의 메모리 배열에서 시작 주소의 row값을 나타낸다.

Int starty – 16\*65536의 메모리 배열에서 시작 주소의 column값을 나타낸다.

Int i - for문의 iterator의 기능을 하는 변수

Int temp – 주소 값을 16진수로 변형하는데 사용되는 변수

Int upper – 1byte의 메모리의 upper half bit값

Int lower – 1byte의 메모리의 lower half bit값

Char addr[5] – 메모리 주소의 값을 16진수로 저장 해 놓는 배열

Int address – 메모리의 각 row의 시작 주소의 값을 저장하는 변수

Int endx – 16\*65536의 메모리 배열에서 마지막 주소의 row값을 나타낸다.

Int endy – 16\*65536의 메모리 배열에서 마지막 주소의 column값을 나타낸다.

Char mem[16][65536] – 메모리에 접근하기 위해서 사용된다.

Int index = 메모리 주소의 값을 인덱싱 하기 위해서 사용되는 변수

기능: 입력으로 들어 온 시작주소부터 마지막 주소까지의 메모리 내용을 출력한다. 이 때

가장 좌측에는 16개의 메모리 단위로 메모리 주소를 16진수로 보여준 뒤, 출력해야

하는 주소의 범위 내에서 해당 메모리 주소에 포함되는 메모리의 값을 보여준다.

메모리의 값을 보여줄 때는 1byte의 값을 상위 4bit와 하위 4bit의 값으로 보여준다.

그리고 가장 우측에는 각 메모리 값에 해당하는 아스키 코드의 값을 보여준다. 해당

되는 아스키 코드의 값이 없는 경우에는 ‘.’을 출력한다.

3.7 edit\_func()

입력: NULL

출력: NULL

사용변수: int x – 변경해야 하는 메모리 주소의 row값을 저장한다.

Int y – 변경해야 하는 메모리 주소의 column값을 저장한다.

Char mem[16][65536] – 변경해야 하는 메모리 주소에 접근하기 위해서 사용한다.

기능: command로 들어온 주소에 해당하는 메모리의 값을 command로 들어온 value값으로

변경 해 준다.

3.8 fill\_func()

입력: NULL

출력: NULL

사용변수: int startx – 16\*65536의 메모리 배열에서 시작 주소의 row값을 나타낸다.

Int starty -16\*65536의 메모리 배열에서 시작 주소의 column값을 나타낸다.

Int endx – 16\*65536의 메모리 배열에서 마지막 주소의 row값을 나타낸다.

Int endy – 16\*65536의 메모리 배열에서 마지막 주소의 column값을 나타낸다.

Int i, j - for문의 iterator 변수

Char mem[16][65536] – 주어진 주소에 해당하는 메모리에 접근하기 위해서

사용된다.

기능: command로 들어온 시작 주소부터 마지막 주소까지 command로 들어온 value의

값으로 변경 해 준다.

3.9 opcode\_list\_func()

입력: NULL

출력: opcode를 저장해 놓은 hash table을 출력한다.

사용변수: int i - for문의 iterator 변수

Opcode \*hash\_table – hash table을 출력하기 위해서 hash table의 내용을

읽어 들이기 위해 사용된다.

Op\_node\* walk – hash table을 순회하기 위한 pointer

기능: 프로그램 내에 저장 된 mnemonic의 opcode가 저장 된 hash\_table을 순회하며

hash\_table의 내용을 출력한다.

3.10 opcode\_func(char\* command)

입력: command

출력: 입력으로 들어 온 mnemonic에 해당하는 opcode의 값을 출력한다.

사용변수: int index – hash function을 이용하여 입력으로 들어 온 command가 저장 된

Index의 정보를 가지고 있는다.

Int ret – command가 opcode를 가진 command인지 여부를 return 해주는 변수

Op\_node \*walk – command로 들어온 mnemonic을 hash table에서 찾는 포인터

기능: 입력으로 들어온 mnemonic에 해당하는 opcode를 찾아서 출력 해 준다. 만약 해당

Mnemonic이 hash table에 저장되지 않았다면 해당 명령어에 대한 opcode가 없다는

문장을 출력 해 준다.

3.11 dump\_slice\_str(int index)

입력: command의 길이

출력: NULL

사용변수: char str[100] - shell에서 입력 받은 command를 copy하기 위한 변수

Int i - for문에서 iterator의 기능을 하는 변수

Int size – command의 각 argument size를 저장 해 놓는 변수

Char \*startstr – dump 명령어와 함께 들어오는 start 주소 argument를

저장하는 변수

Char \*endstr – dump 명령어와 함께 들어오는 end 주소 argument를 저장하는

변수

Int sixteen – 각 주소의 값을 16진수로 표시하기 위해 사용되는 자릿수를 표현

하는 변수

Boolean start\_appear – command에 start argument가 들어왔는지 여부를 저장

Boolean end\_appear – command에 end argument가 들어왔는지 여부를 저장

Int start – character형으로 들어 온 start argument의 값을 10진수로 변환하여

저장하는 변수

Int end – character형으로 들어 온 end argument의 값을 10진수로 변환하여

저장하는 변수

기능: dump 명령어가 입력되었을 때, start와 end address argument가 같이 명령어로

입력된 경우, 어떤 argument가 입력되었는지 확인하고, 해당 주소의 값을 parsing

하여 10진수의 값으로 변환하여 저장한다.

3.12 edit\_slice\_str(int index)

입력: command의 길이

출력: NULL

사용변수: char str[100] - shell에서 입력 받은 command를 copy하기 위한 변수

Int i - for문에서 iterator의 기능을 하는 변수

Int size – command의 각 argument size를 저장 해 놓는 변수

Char \*startstr – dump 명령어와 함께 들어오는 start 주소 argument를

저장하는 변수

Char \*valuestr – dump 명령어와 함께 들어오는 value 주소 argument를

저장하는 변수

Int sixteen – 주소와 value의 값을 10진수로 변환하기 위해 사용되는 자릿수를

표현하는 변수

Boolean start\_appear – command에 start argument가 들어왔는지 여부를 저장

Boolean value\_appear – command에 value argument가 들어왔는지 여부를

저장

Int start – character형으로 들어 온 start argument의 값을 10진수로 변환하여

저장하는 변수

Int value– character형으로 들어 온 value argument의 값을 10진수로 변환

하여 저장하는 변수

기능: edit 명령어가 입력되었을 때, 같이 입력 된 address와 value argument를 parsing

하여 각 argument를 10진수의 값으로 변환하여 저장한다. 그리고 필요한

argument가 모두 입력되었는지 확인하는 작업도 진행한다.

3.13 fill\_slice\_str(int index)

입력: command의 길이

출력: NULL

사용변수: char str[100] - shell에서 입력 받은 command를 copy하기 위한 변수

Int i - for문에서 iterator의 기능을 하는 변수

Int size – command의 각 argument size를 저장 해 놓는 변수

Char \*startstr – dump 명령어와 함께 들어오는 start 주소 argument를

저장하는 변수

Char \*valuestr – dump 명령어와 함께 들어오는 value 주소 argument를

저장하는 변수

Char \*endstr - dump 명령어와 함께 들어오는 end 주소 argument를 저장하는

변수

Int sixteen – 주소와 value의 값을 10진수로 변환하기 위해 사용되는 자릿수를

표현하는 변수

Boolean start\_appear – command에 start argument가 들어왔는지 여부를 저장

Boolean value\_appear – command에 value argument가 들어왔는지 여부를

저장

Boolean end\_appear - command에 end argument가 들어왔는지 여부를 저장

Int start – character형으로 들어 온 start argument의 값을 10진수로 변환하여

저장하는 변수

Int value– character형으로 들어 온 value argument의 값을 10진수로 변환

하여 저장하는 변수

Int end – character형으로 들어 온 end argument의 값을 10진수로 변환하여

저장하는 변수

기능: fill 명령어가 입력되었을 때, 같이 입력 된 address와 end, value argument를

parsing하여 각 argument를 10진수의 값으로 변환하여 저장한다. 그리고 필요한

argument가 모두 입력되었는지 확인하는 작업도 진행한다.

3.14 build\_hash\_table()

입력: opcode.txt file

출력: NULL

사용변수: FILE \*fp – opcode.txt파일을 읽어 들일 변수

Boolean opcode\_file\_error – directory 내 읽어 들일 opcode.txt의 존재 여부를

저장하는 flag 변수

Op\_node\* node – opcode.txt에서 정보를 읽어와 해당 정보를 저장 할 hash

Table의 node

Int index – hash function을 거쳐 해당 opcode와 mnemonic이 저장 될 index를

저장하는 변수

Op\_node\* hash\_table[20] – 생성 할 hash\_table의 구조를 가지고 있는 변수

기능: directory에 있는 opcode.txt 파일을 읽어와 opcode hash table을 생성한다.

3.15 push(char\* str, int index)

입력: linked list에 추가 할 command와 해당 command의 길이

출력: NULL

사용변수: str\_node\* new – 새로운 linked list의 node

Str\_node\* history\_head – history linked list의 head pointer

Str\_node\* history\_tail – history linked list의 tail pointer

기능: 입력으로 들어 온 str을 command 순서를 저장하는 historynum과 함께

history linked list의 가장 뒤부분에 저장한다

3.16 subcmp(int start, int end, char\* str1, char\* str2)

입력: 비교 할 두 문자열 str1, str2, 그리고 str2와 비교 할 str1의 start index와

끝 index

출력: NULL

사용변수: int I - for문에서 iterator역할을 위한 변수

Boolean ret – 두 문자열의 비교 결과를 저장하는 변수

기능: 입력으로 들어 온 str1의 start index부터 end-1 index까지를 str2와 비교한다.

그리고 비교 결과가 같다고 나오면 true값을, 비교 결과가 다르다고 나오면

false값을 return한다.

3.17 init\_global\_variable()

입력: NULL

출력: NULL

기능: 초기화가 필요한 history\_head, historynum, startaddress, opcode\_file\_error

변수를 알맞게 초기화한다.

3.18 main()

입력: shell 명령어

출력: 해당 명령어에 해당하는 출력

기능: 입력으로 들어 온 명령어를 분석하여 해당 명령어의 기능을 수행하는 함수를

호출한다. 이 때, dump, fill, edit과 같이 address 또는 value argument를 같이

가지고 오는 경우, 해당 address나 변수가 범위를 벗어나지 않는지 체크하는 과정

도 거친다. 그리고 argument가 comma로 구분되어있지 않은 경우도 확인한다.

또 해당 프로그램에서 해석할 수 없는 command가 들어오는 경우도 확인한다.

**4. 전역 변수 정의**

Typedef enum Boolean{false, true} boolean;

-c언어에 정의되어 있지 않은 Boolean 형식을 간편하게 사용하기 위한 enum.

Typedef struct str\_node{

Int index;

Char sentence[150];

Struct str\_node\* ptr;

}str\_node;

-history linked list를 구성하는 node. Index는 해당 command의 순서를 저장하고,

sentence는 command를 저장하고, ptr는 다음 node를 가리키기 위해 정의되었다.

Typedef struct op\_node{

Char hexa[3];

Char command[10];

Char format[10];

Struct op\_node\* ptr;

}op\_node;

-opcode hash table linked list를 구성하는 node. Hexa는 opcode의 값을 저장하고,

Command는 mnemonic의 값을 저장하고, format는 해당 mnemonic의 format을 저장하고,

Ptr은 다음 node를 가리키기 위해 정의되었다.

Str\_node \*history\_head;

-history linked list의 가장 head를 가리키는 포인터

Str\_node \*history\_tail;

-history linked list의 tail을 가리키는 포인터

Int historynum;

-history linked list에 새로운 node가 추가될 때, 해당 ndoe가 몇 번째 node인지 정보를

담고있는 변수

Int startaddress;

-dump명령어가 아무런 argument 없이 들어오는 경우, 처음에는 0번째부터 시작하여 160개

메모리를 출력하고, 그 다음부터는 이전에 출력된 마지막 메모리의 다음 메모리부터 160개

메모리를 출력한다. 따라서 dump명령어만 들어온 경우를 위해 매 출력 시 시작 address의

정보를 저장하고 있다.

Char sentence[100];

-shell에서 입력 받은 command를 저장하는 변수

unsigned char mem[65536][16];

-이 프로그램에서 할당 된 가상 메모리 공간

op\_node\* hash\_table[20];

-opcode의 정보를 저장 할 hash\_table

Boolean opcode\_file\_error;

-읽어 들일 opcode.txt파일이 존재하는지 여부에 대한 결과를 저장하는 flag 변수

Boolean start\_appear;

-명령어 이외의 argument가 들어 온 경우 start address에 해당하는 argument가 들어왔는지

여부에 대한 결과를 저장하는 flag 변수

Boolean end\_appear;

-명령어 이외의 argument가 들어 온 경우 end address에 해당하는 argument가 들어왔는지

여부에 대한 결과를 저장하는 flag 변수

Boolean value\_appear;

-명령어 이외의 argument가 들어 온 경우 value에 해당하는 argument가 들어왔는지 여부에

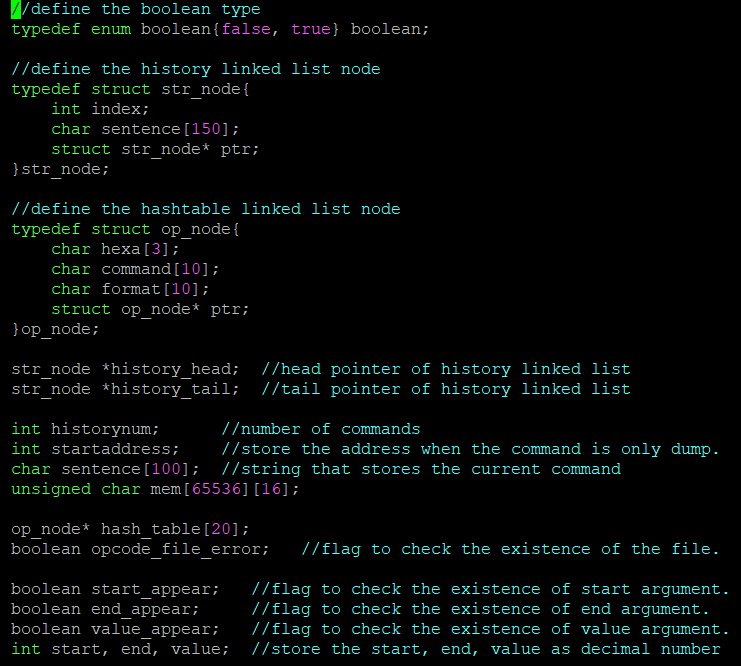
대한 결과를 저장하는 flag 변수

Int start, end, value;

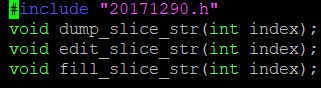
-argument로 들어온 start, end, value값을 10진수로 변환하여 저장하는 변수

**5. 코드 설명**

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

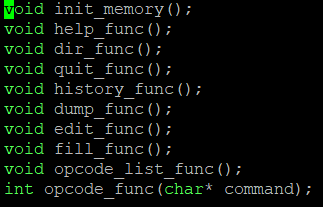


Slice.h : argument를 parsing하는 함수의 prototype을 저장 해 놓은 파일



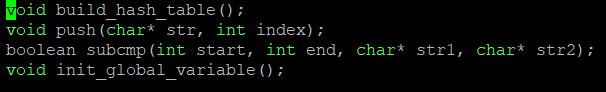
Command.h : 프로그램 내에서 인식하는 명령어에 따른 기능을 수행하는 함수의 prototype을

저장 해 놓은 파일



Subfunc.h : 프로그램 내에서 인식하는 명령어에 따른 기능을 수행하는 함수를 작성할 때

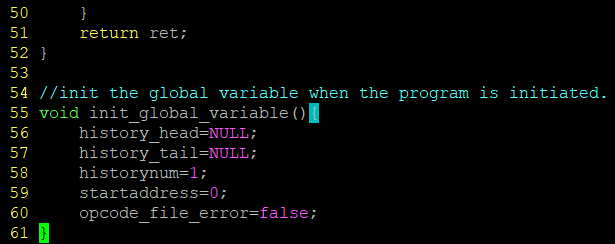
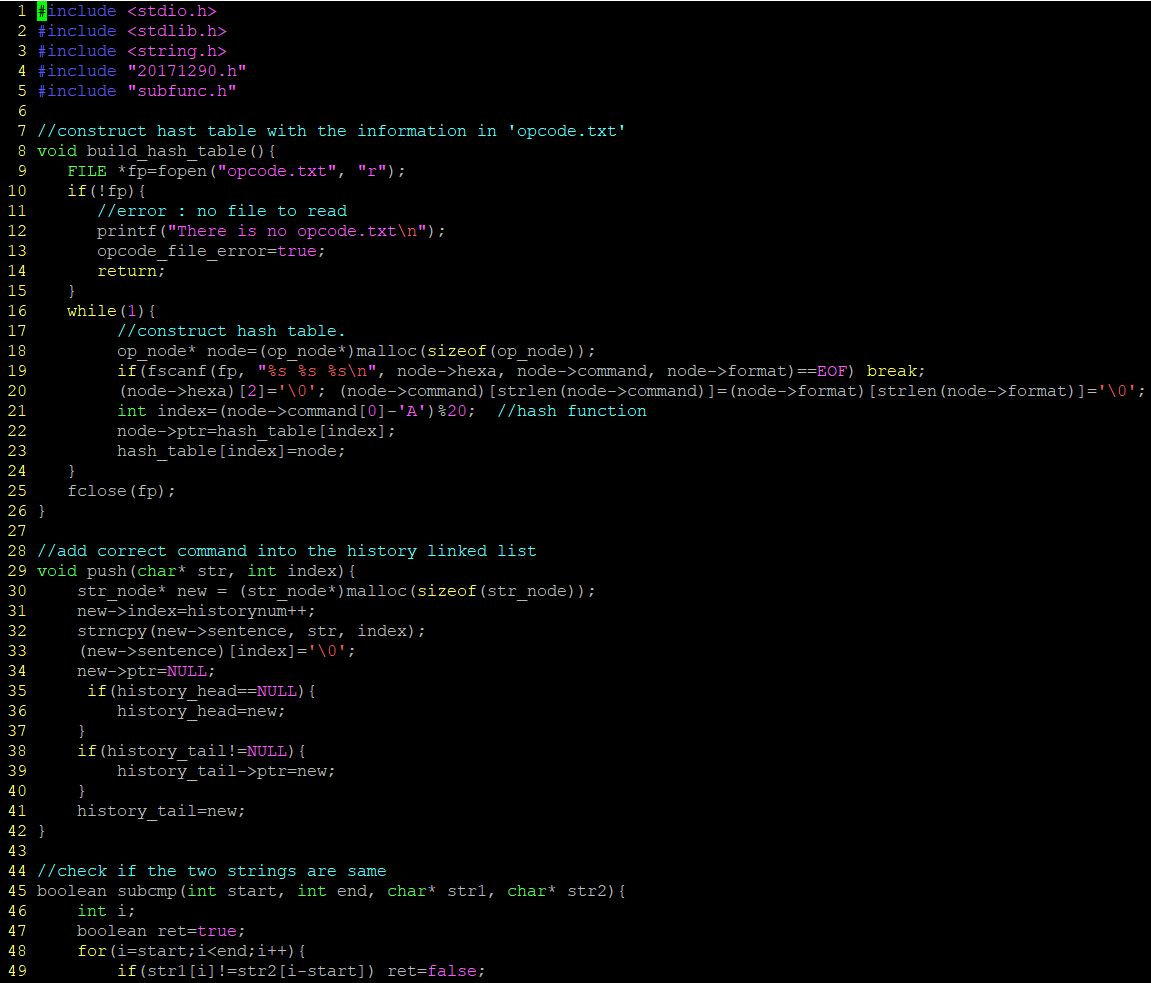
필요한 함수들의 prototype을 저장 해 놓은 파일



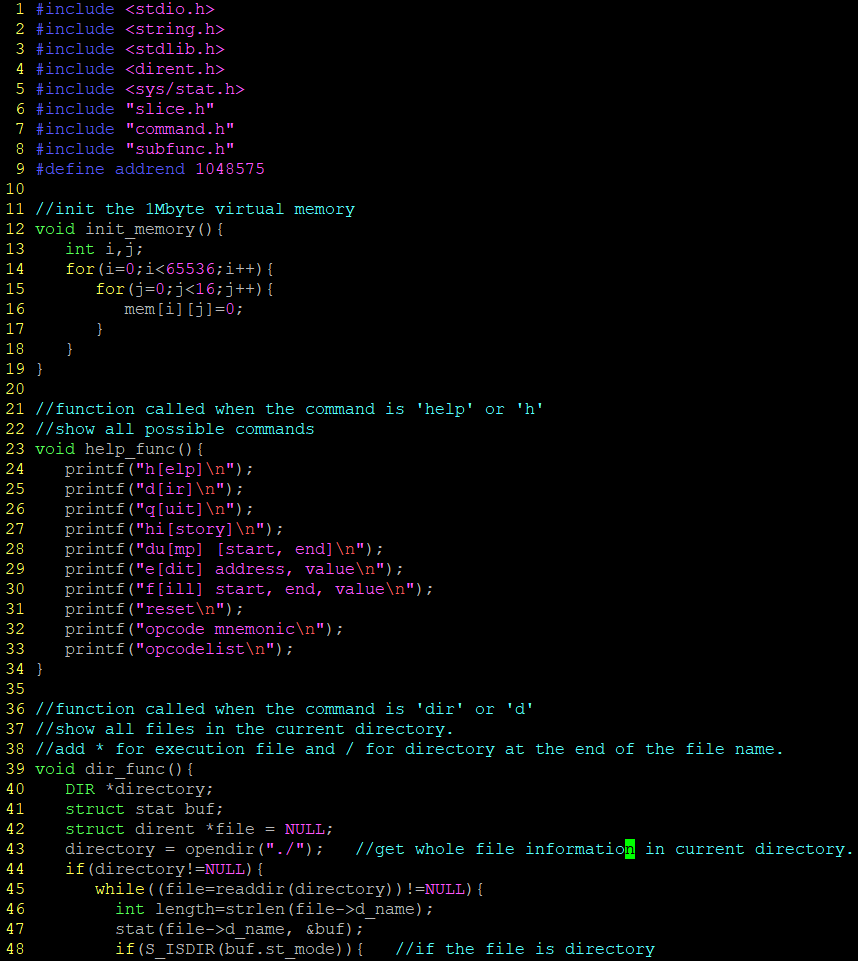
Subfunc.c : 프로그램을 작성하기 위해서 필요한 부가적인 함수들을 구현 한 코드

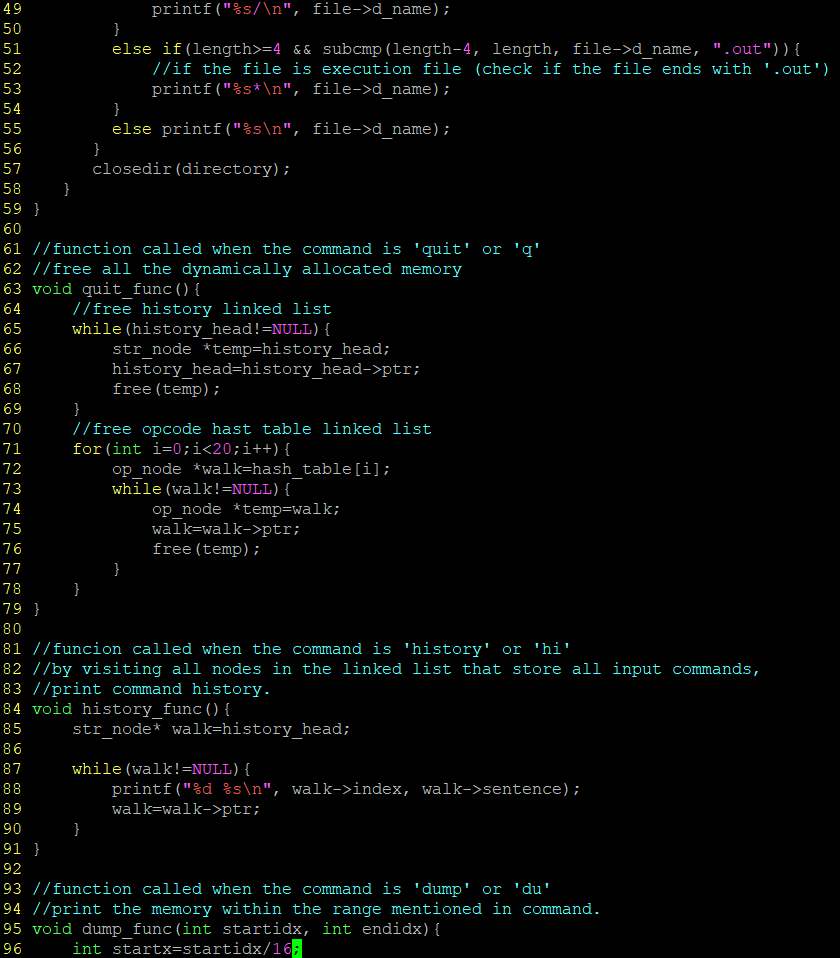
부가적인 함수들로는 hash table을 만드는 함수, history linked list에 command를

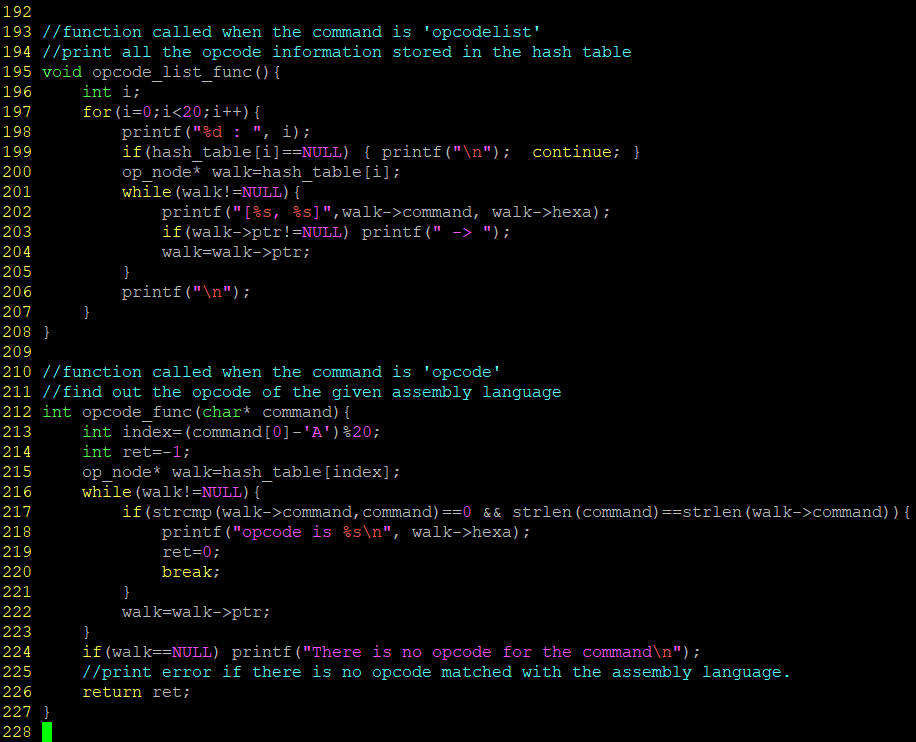
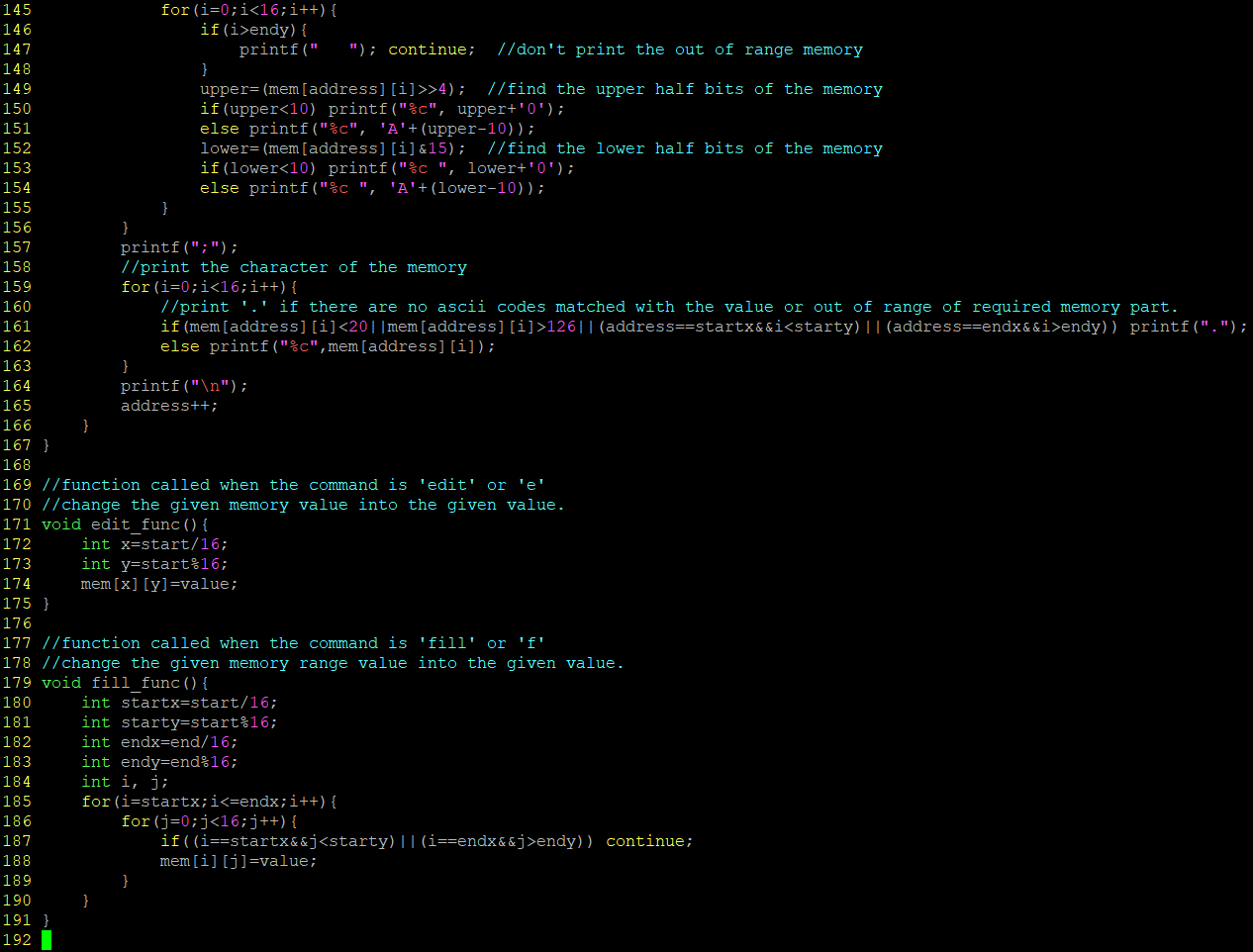
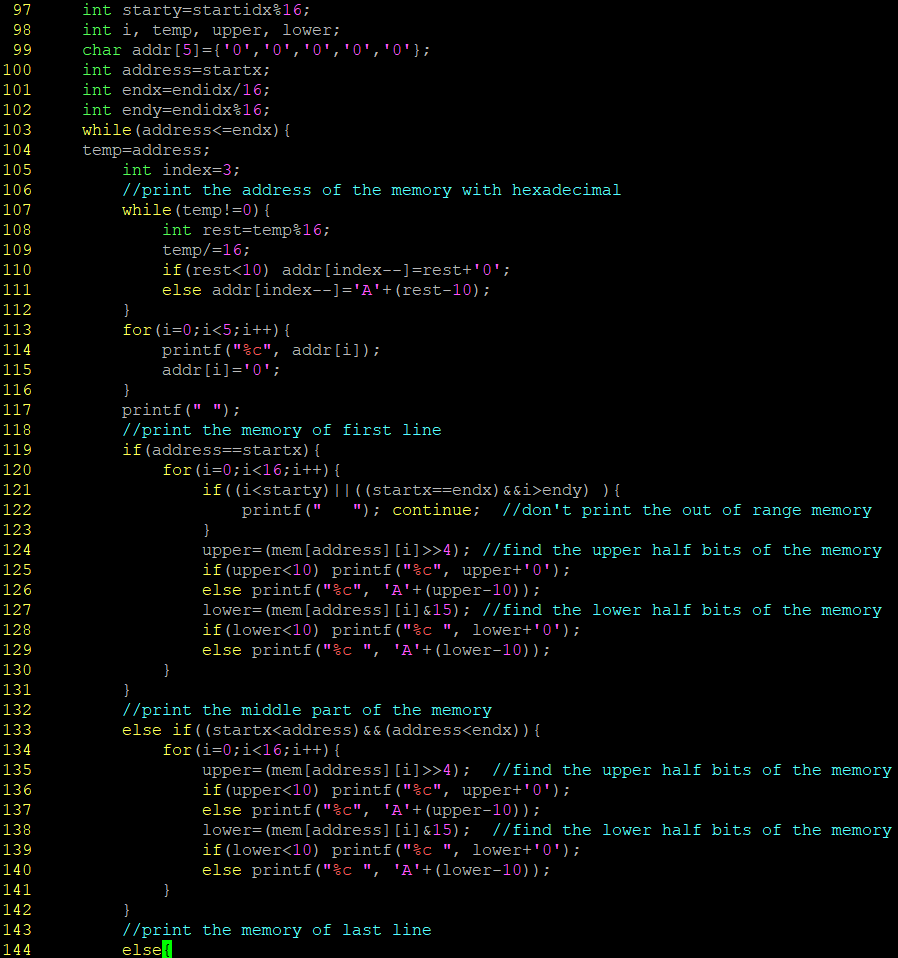
저장하는 함수, 두 문자열을 비교하는 함수, 전역변수를 초기화하는 함수가 있다.



Command.c: 프로그램이 인식하는 각 명령어들의 기능을 구현 한 코드



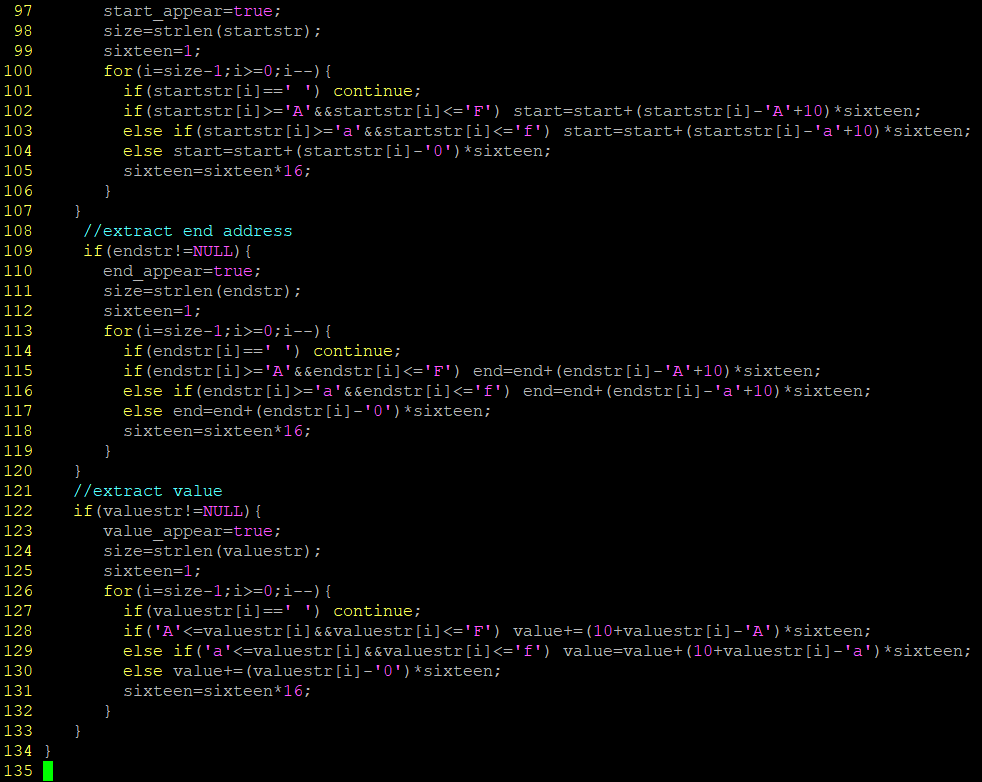
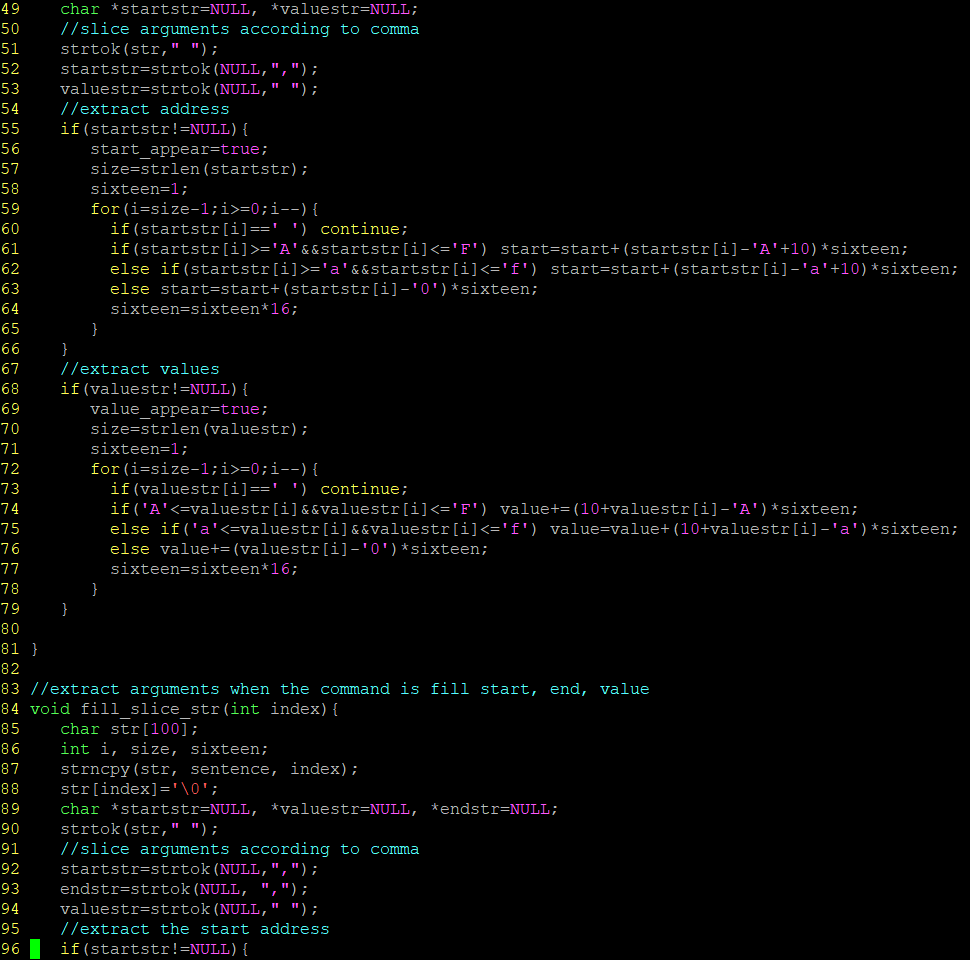
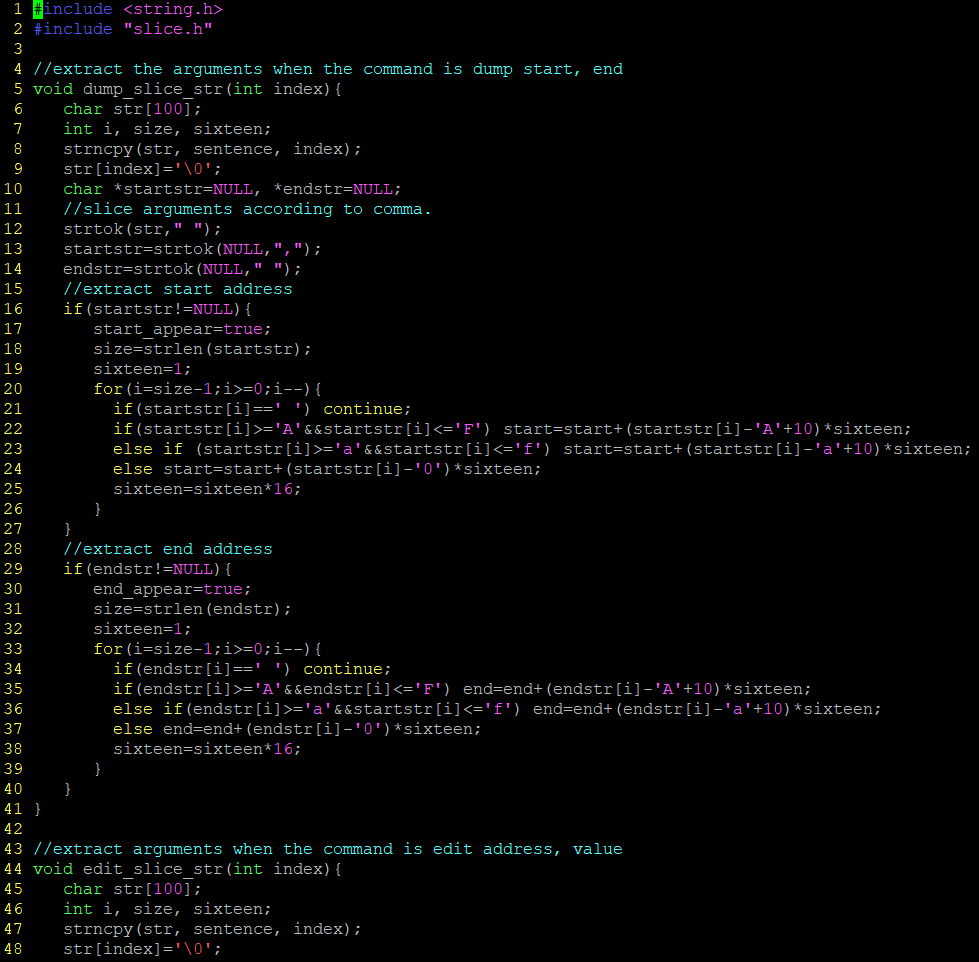




Slice.c : argument parsing이 필요한 command를 parsing 하고, 각 주소나 value값들을

10진수로 변환하여 저장한다.

(코드는 다음 쪽에 있다.)



20171290.c : 프로그램의 main함수. 이 함수에서는 shell 명령어를 입력 받고, 입력 받은 명령어

에 알맞은 함수를 실행하고, 올바른 명령어가 입력되었다면 history에 추가한다.

(코드는 다음 쪽에 있다.)

