**과목명: 시스템프로그래밍**

**2 분반**

**<<Project #3>>**

**서강대학교 공학부 컴퓨터공학과**

**20181703**

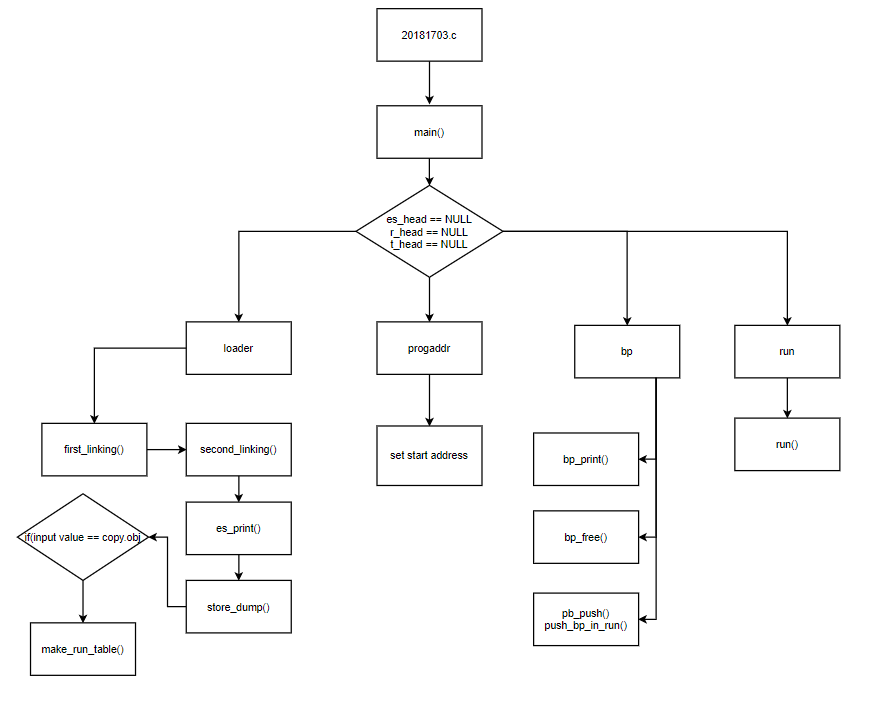
**황동준**

**목 차**

1. **프로그램 개요**
2. **프로그램 흐름도**
3. **모듈 정의 (함수정의)**
4. int main()
5. void first\_linking(int programcount, char \*input\_program);
6. void es\_push(char \*cs, char \*sn, int address, int length);
7. void r\_push(int programcount, int number, char \*symbol\_name);
8. void es\_print();
9. void second\_linking(int programcount, char \*input\_program);
10. void store\_dump();
11. void t\_push(int location, char \*objcode, int format);
12. void t\_free();
13. void r\_free();
14. void es\_free();
15. void bp\_push(int point\_loc);
16. void bp\_print();
17. void bp\_free();
18. void push\_bp\_in\_run(int breakpoint);
19. void run();
20. void make\_run\_table();
21. void run\_push(int location, int opcode, int format, int address, int breakpoint, int check\_imm\_idr, int check\_base\_relative);
22. **전역 변수 정의**
23. typedef struct \_ES\_TABLE \*es\_ptr;
24. es\_ptr es\_head;
25. typedef struct \_R\_TABLE \*r\_ptr;
26. r\_ptr r\_head[3];
27. typedef struct \_T\_TABLE \*t\_ptr;
28. t\_ptr t\_head;
29. typedef struct \_BP\_TABLE \*bp\_ptr;
30. bp\_ptr bp\_head;
31. typedef struct \_RUN\_TABLE \*run\_ptr;
32. run\_ptr run\_head;
33. int progaddr;
34. int csaddr;
35. **코드**
36. **프로그램 개요**

프로젝트 1, 2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가하는 프로그램이다. 프로젝트 2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모 리에 올리는 일을 수행한다.

1. **프로그램 흐름도**



1. **모듈 정의(함수 정의)**
2. **int main()**

* 함수 설명

이번 과제를 수행할 때에는 총 4개의 linked List가 쓰였다. 각각의 의미는 아래의 전역변수 설명에서 설명할 것이다. main() 함수 안에서는 이 구조체들을 초기화 시켜주고 input을 받는 역할로 이용이 되었다. 받는 input은 progaddr, loader, bp, run이 있고 각각 이 input을 받았을 때 적절하게 함수를 수행하여 준다. progaddr 이라는 변수 또한 전역으로 선언 된 것이므로 아래에서 설명할 것이다.

쓰이는 변수

|  |
| --- |
| es\_head=NULL;  for(int i=0;i<3;i++){  r\_head[i]=NULL;  }  run\_head=NULL  t\_head = NULL; |
| 추가로 선언된 변수는 없으며 다음과 같은 구조체 들만 초기화를 해주었다. |

1. **void first\_linking(int programcount, char \*input\_program);**

* 함수 설명

이 함수를 통해서 linking\_loader의 pass 1을 구현할 수 있다. 첫번째 pass에서는 H와 D, R의 구조를 받아내어 각각의 symbol들이 다른 파일 어디에 위치하였는지를 파악하여 es table을 만들어내는 역할을 한다. 그래서 먼저 파일을 읽어낸 후, 각각 H,D,R의 경우에 따라서 파일을 읽어낸 후 es\_table에 저장하는 역할을 한다. 저장하는 방법은 보통 char형으로 되어있는 object code들을 int형으로 바꿔주어 표현할 때만 16진법으로 print할 수 있도록 하였다. 이때 여기 안에서 쓰는 함수가 es\_push, es\_print, r\_push가 있다. 이들은 모두 linked list를 출력하고 입력하는 방식을 고수하는 함수이다.

쓰이는 변수

|  |
| --- |
| FILE \*fp;  char input[100];  char \*input\_value;  char \*subinput;  char cs[10];  char sn[10];  char cs\_main[10];  char ch\_address[9];//into char and change to decimal address(int)  int address;  int length; |
| fp는 파일을 받아오기 위한 파일 포인터, input은 파일의 한줄 한줄을 받아내기 위해 쓴 변수이며, input\_value는 input 을 strtok하기 위해 쓰는 포인터, subinput은 fgets로 input을 받아낼 때 마지막의 값을 NULL 로 바꿔주기 위해 쓰는 포인터 이다.  cs, sn은 각각 control section, symbol name을 의미하여, es\_table에 저장할 변수들을 char 형태로 push할 때 쓰는 변수이고, cs\_main은 record 라는 r\_table에 저장을 할 때 쓰이는 변수로, 다른 테이블에 있는 것들을 해당 파일로 옮길 때 이용하게 된다. cs\_main인 이유는 record에서 첫번째 값 즉, number가 1인 경우 프로그램의 첫번째 주소를 가르키게 되기 때문이다. ch\_address는 address라는 변수에 address의 int 값을 넣어주기 위해 만든 변수이다. length는 이 프로그램의 길이를 말한다. |

1. **void es\_push(char \*cs, char \*sn, int address, int length);**

**●** 함수 설명

이 함수는 es\_table에 해당 값을 push하기 위해 만들어 진 것으로 Linked List에 저장하기 위해 만들어 졌다. 다른 push와 차이점은 없다.

쓰이는 변수

|  |
| --- |
| es\_ptr padd, pmove |
| es\_table에 저장하기 위해 만든 변수로 이 pointer들을 이동시켜 해당 LinkedList의 자료를 저장할 수 있도록 도와주는 변수이다. |

1. **void r\_push(int programcount, int number, char \*symbol\_name);**

* 함수 설명

이 함수 또한 마찬가지로 해당 record 부분을 저장하는 함수이다. r이라는 record table을 만들어 Linked List 형태로 구현하였으며 다른 함수와 같이 push를 할 때 쓰이는 pmove, padd 포인터가 이용되었다.

쓰이는 변수

|  |
| --- |
| r\_ptr padd, pmove; |
| 위와 마찬가지로 r\_table에 자료를 push해 주기 위해 만들었다. 이 table이 NULL 일 경우 r\_ptr padd 와 그 값이 같도록 설정하였으며, 그렇지 않을 경우 pmove를 움직여서 Linked List의 끝에 집어 넣은 자료가 저장되는 구조로 만들었다. |

1. **void es\_print();**

* 함수 설명

es\_table에 있는 것들을 print하기 위해 만든 함수로 이 함수가 쓰인 이유은 loader를 실행 하였을 때, control section, symoblo name , address, length가 모두 있는 그런 es\_table을 출력할 수 있도록 하였기 때문이다. pmove를 이용하여 es\_table의 LinkedList의 긑까지 자료를 출력할 수 있도록 만들었다.

|  |
| --- |
| es\_ptr pmove;  int total\_length=0 |
| pmove 는 마찬가지로 Linked List의 끝까지 자료를 돌아보기 위해 만들어진 pointer 이고, total\_length는 각각의 함수에 해당되는 length를 더한 후 이 코드의 전체 길이를 표현하기 위해 만들어진 변수이다. es\_table의 program 최대 3개의 length를 모두 더한 것이다. |

1. **void second\_linking(int programcount, char \*input\_program);**

* 함수 설명

first\_linking에서 만든 es, r table을 이용하여 만든 object code 자료와 symbol의 부분 나열을 이용하는 함수로, 이를 이용하여 dump memory에 해당되는 object code를 입력하는 것이 주 목적이다. 이 함수의 가장 큰 특징은 먼저, Text에 해당하는 부분을 Linked List 에 저장한 후 Modify 되는 부분을 Linked List로 검색하여 수정하여 준다. 이때 modify하는 부분에서 다른 program 파일과 관련이 생기는 부분은 record table을 통해 접근 할 수 있도록 한다. 이때 record table이 전역 변수로 총 3개가 선언 되어 있는데, 이는 입력을 받을 때 각각의 program 마다 다른 record table을 받을 수 있도록 설정한 것이다. 또한 여기에서 Text 부분의 object code를 수정할 때 format을 찾아서 format 4이며 8개의 문자를 입력받고, 나머지는 6개의 입력을 받을 수 있도록 하였다. 그래서 이 함수가 끝날 때 까지 총 두 개의 경우의 수가 있는데 하나는 Text 인 경우와 Modify인 경우 인 것이다.

쓰이는 변수

|  |
| --- |
| FILE \*fp;  char input[255]={0};  char \*input\_value;  char \*subinput;  char store\_input[9];  int location=0;  int mod\_location;  int format;  int record\_num;  int mod\_result;  int minus\_or\_plus;//minus=0 plus=1  char record\_num\_store[3];  char text\_size[3]={0};  int count\_text\_size=0;  int count\_location;  r\_ptr pmove\_r;  t\_ptr pmove\_t;  es\_ptr pmove\_es; |
| fp 는 파일 포인터로 파일을 열기 위해 쓰이는 변수이고, input은 파일을 읽은 것을 담는 변수이며, input\_value는 strtoke를 하고 input 값의 포인터를 움직이기 위한 변수, subinput은 fgets로 받은 input의 끝에 enter을 없애 주는 역할을 하는 변수이다. store\_input 같은 경우 받은 object 코드들을 해석하고 strtok로 자른 input을 strcpy로 다시 옮겨서 따로 분석하게 만드는 변수 역할을 하게 된다.  location은 object code를 해석할 때 해당 code 들이 주소 값을 가지게 되는데, 그 주소를 뜻하며, mod\_location은 modified된 record들의 location 값을 의미하게 된다. format은 T table에 넣은 것들을 format으로 분리하여 저장하기 위해 선언된 변수이며, record\_num은 이 파일 말고 다른 파일에 record된 symbol들의 번호들을 알려주는 변수로, r\_table에서 그 값을 찾아 modify 해준다.  mod\_result는 modify이후 나타낸 값들을 말하며, minus\_or\_plus는 negative한 지 아닌지 판단하여 modify 해주는 변수를 말한다.  text\_size는 T옆의 location 옆에 한 줄에 얼마 크기의 text 가 적혀있는지를 의미하며, 나머지 count 또한 해당 변수의 개수를 세주기 위해 나타낸 변수이다.  pmove는 r\_table에 있는 변수들을 쓰기 위해 만들었다. |

1. **void store\_dump();**

* 함수 설명

수정한 object code를 t\_table을 통해 자료에 접근한 후 접근한 자료를 dump에 실제로 저장하는 역할을 한다. t\_table에 modify된 location이 저장되어 있으므로 이 location에 접근하여 dump 의 해당 주소 부분에 접근한 후 object코드를 집어 넣게 된다. 이때 format을 구분하여 얼마만큼의 크기를 dump에 넣을지 확인하고 넣는다.

쓰이는 변수

|  |
| --- |
| t\_ptr pmove;  pmove = t\_head;  int location;  int address;  char store\_obj[3]={0}; |
| pmove는 t\_table이라는 Linked List에 접근 하기 위해 만든 변수로 먼저 pmove에 t\_head를 넣어 첫번째 노드에 접근한 후 줄줄이 자료에 접근하면 된다. location은 pmove에 있는 location을 먼저 집에 넣고, 2차원으로 되어있는 dump부분에 각각 넣어주기 위해서 나머지 부분과 몫 부분의 위치의 dump를 찾기 위해 location이 만들어 졌다. address 또한 마찬가지 이다. object code는 char 형으로 되어있으므로 이를 change\_to\_dec이라는 함수를 이용하여 int형으로 바꿔줘야 하기 때문에 store\_obj를 이용하여 먼저 char 자료형에 값을 받았다. |

1. **void t\_push(int location, char \*objcode, int format);**

● 함수 설명

이 함수는 t\_table에 push를 해주기 위해 만든 함수로 linking\_pass\_2에서 이용할 수 있게 된다. 쓰이는 변수 또한 다른 Linked List에 push해 줄 때와 같다.

쓰이는 변수

|  |
| --- |
| t\_ptr padd, pmove; |
| padd를 통해 Linked List의 노드를 만들고 pmove를 통해 테이블을 이동시키면서 맨 끝 자리의 노드를 탐색해 내게 된다. |

1. **void t\_free();**

● 함수 설명

만들었던 t\_table을 free해 주기 위해 만든 함수로 매우 간단하며 변수 또한 다른 free 함수와 비슷하다. 이는 free.c라는 free 전용 파일에 적어 놓았다.

쓰이는 변수

|  |
| --- |
| t\_ptr pfree; |
| pfree 를 이용하여 먼저 t\_head에서 값을 받고 이를 pfree에 저장해 free시켜주는 변수이다. 항상 free를 당하는 변수인 것이다. |

1. **void r\_free();**

● 함수 설명

이 또한 r\_table을 free시키기 위한 함수이다. free.c 파일 안에 접근 해야 한다.

쓰이는 변수

|  |
| --- |
| r\_ptr pfree; |
| pfree 를 이용하여 먼저 r\_head에서 값을 받고 이를 pfree에 저장해 free시켜주는 변수이다. 항상 free를 당하는 변수인 것이다. |

1. **void es\_free();**

**●** 함수 설명

이 또한 es\_table을 free시키기 위한 함수이다. free.c 파일 안에 접근 해야 한다.

쓰이는 변수

|  |
| --- |
| es\_ptr pfree; |
| pfree 를 이용하여 먼저 es\_head에서 값을 받고 이를 pfree에 저장해 free시켜주는 변수이다. 항상 free를 당하는 변수인 것이다. |

1. **void bp\_push(int point\_loc);**

● 함수 설명

breakpoint 를 지정하는 부분에서 쓰이는 함수로 bp 라는 명령어를 이용하여 호출이 가능하다. 예를 들어 bp 3이라고 했을 때 3이라는 값이 bp\_table의 노드로 push되는 것이다. 다른 push와 똑 같은 원리로 진행이 된다.

쓰이는 변수

|  |
| --- |
| bp\_ptr padd, pmove; |
| pmove를 이용하여 Linked List안의 노드 값에 접근 하고 맨 마지막에 padd 를 추가함으로써 노드를 추가하게 된다. |

1. **void bp\_print();**

● 함수 설명

모든 breakpoint를 출력하는 함수로 간단하게 pmove를 이용하여 bp\_table에 접근하여 모든 노드들이 탐색 될 때 즉, pmove가 NULL이 될 때 까지 breakpoint를 출력한다.

쓰이는 변수

|  |
| --- |
| bp\_ptr pmove; |
| pmove 만을 이용하여 모든 노드들에 접근하여 출력하는 변수로 pmove를 이용하였다. |

1. **void bp\_free();**

● 함수 설명

위에서 push로 만든 LinkedList를 다시 전부 free해주는 함수이다.

쓰이는 변수

|  |
| --- |
| bp\_ptr pfree; |
| pfree를 통해 bp\_head안에 있는 값을 받고 bp\_head를 다음 link로 움직인 후, pfree를 free하는 형식으로 메모리를 해제하게 된다. |

1. **void push\_bp\_in\_run(int breakpoint);**

● 함수 설명

이는 breakpoint를 object code가 진행할 때 관여 시킬 수 있도록 하기 위해 만들어진 push 함수이다. 이는 bp의 location이 push 된 object code 즉, run\_table에 있는 location과 같을 경우 breakpoint를 표시하는 변수에 1값을 넣어줌으로서 나중에 코드가 진행될 때 이 부분을 만나면 자동으로 밖으로 나갈 수 있도록 설정을 해 놓았다.

쓰이는 변수

|  |
| --- |
| run\_ptr pmove; |
| run\_table에 접근하여 처음 pmove = run\_head로 시작점을 찾은 후, 하나 하나씩 location과 똑 같은 경우의 노드를 찾아내게 된다. 찾아내기 위해 이용할 노드를 pmove로 선언하였다. |

1. **void run();**

● 함수 설명

이 함수는 main에서 run이라는 명령어를 입력하였을 때, 실행되는 함수이다. 이 명령어가 실행되면 아래에서 말하겠지만 이미 만들어 놓은 run\_table에 접근하여 다음과 같은 opcode들이 어떠한 역할을 가지고 있는지 역 어셈블 하게 된다. 즉, object code를 보고 어떠한 실행결과가 register에 들어가게 되는지를 알려주는 함수인 것이다. 이 run이 진행될 때에는 breakpoint까지 진행되며 breakpoint가 없어질 경우 코드가 끝까지 진행되게 된다. 각각의 opcode마다 다른 기능을 가지고 있어서 if else문으로 전부 처리해 주었다.

|  |
| --- |
| int reg1;  int breakpoint;  int end\_program = 0;  int a=0,x=0,l,pc,b=0,s=0,t=0,cc;  run\_ptr pmove; |
| 여기에서는 많은 변수를 가지고 있지 않다. 왜냐하면 run\_table에서 이미 다 만들었기 때문이다. 먼저 reg1은 여러 opcode들 중에서 register이 특정되지 않은 opcode들이 있는데 그러한 경우에 register을 의미하는 변수로 쓰인다. breakpoint는 breakpoint가 어느 지점에서 발견되어 run이 종료되었는지를 말해준다. end\_program은 만약 프로그램이 끝까지 실행되어 종료되었을 경우에는 1이라는 값으로 만들어진다.  그리고 나머지 a,x,l,b,s,t,pc,cc 모드 register을 의미하며 이 중에서 cc는 compare 명령어로 인해 바뀔 수 있는 것이다.  pmove는 object code의 노드들에 대해 접근하기 위해 만들어진 변수이다. |

1. **void make\_run\_table();**

● 함수 설명

이것은 위의 함수 run에서 쓰일 object code list를 만들어 주는 함수이다. 보고서의 저자는 dump에서 memory를 받아오지 않고 한번 읽혔던 input\_program을 한번 더 읽어서 표절률을 낮추었다. 이는 object file을 다시 읽어서 해당 프로그램에 맞는 구조체를 가지게 된 것이다. 따라서 이 함수는 run\_table의 Linked List를 만들기 위해 만든 함수이며, 따라서 run\_push또한 만들고 이 안에서 실행하게 된다. 경우는 format에 따라서 다르게 표현했으며, byte와 format2는 먼저 if문에서 처리할 수 있도록 하였고 나머지 format들 또한 차례로 run\_table에 노드로 추가될 수 있도록 Linked List를 작성하였다. 여기에서는 run에 필요한 성분들만을 뽑아서 구조체에 저장하게 되는데, 전역 변수들은 전부 아래에서 설명할 것이다.

|  |
| --- |
| FILE \*fp;  char \*input;  char \*subinput;  int location;  char store\_loc[7]={0};  char store\_address[7]={0};  int address;  char store\_opcode[3]={0};  int opcode;  char check\_format\_store[2]={0};  int check\_format;  int check\_imm\_indr;  int check\_base\_relative;  int before\_location;  int breakpoint;  int check\_byte=0;//if 0, byte is not exist, 1 is exist |
| 먼저 fp와 input 그리고 subinput은 모두 file을 읽을 때 이용하게 된다. 그리고 나머지 location, store\_loc은 location을 object 파일에서 읽어서 int로 변환하고 저장하기 위함이며, address, store\_address 또한 address를 읽어서 int형으로 변환하고 저장하기 위함이다. store opcode와 opcode, 그리고 check\_format\_store과 check\_format또한 그렇게 하기 위함이다. check가 붙은 것은 전부 구조체의 변수라 push하기 위해 만든 변수이며, before\_location은 location이 증가 된 후에 push하고 싶지 않아서 만든 변수이다. |

1. **void run\_push(int location, int opcode, int format, int address, int breakpoint, int check\_imm\_idr, int check\_base\_relative);**

● 함수 설명

이 함수는 run\_table의 LinkedList에 노드를 push하기 위해 만들어진 함수이다. 노드 구조체 안에 들어있는 변수가 많아 넣을 정보가 많기 때문에 변수 또한 많아지게 된다. 그러나 다른 push함수와 알고리즘 자체는 변한 것이 없다.

|  |
| --- |
| run\_ptr padd, pmove; |
| 이 또한 padd에 정보를 집어넣고 pmove로 노드의 끝으로 간 다음 그 끝에 padd를 추가하는 방식으로 Linked List가 만들어지게 되는 것이다. |

1. **전역 변수 정의**

앞서 설명할 전역 변수들은 다음과 같다.

|  |
| --- |
| //homework 3  typedef struct \_ES\_TABLE \*es\_ptr;  typedef struct \_ES\_TABLE{  char cs[10];  char sn[10];  int address;  int length;  es\_ptr link;  }es\_table;  es\_ptr es\_head;  typedef struct \_R\_TABLE \*r\_ptr;  typedef struct \_R\_TABLE{  int number;  char symbol\_name[10];  r\_ptr link;  }r\_table;  r\_ptr r\_head[3];  typedef struct \_T\_TABLE \*t\_ptr;  typedef struct \_T\_TABLE{  int location;  char objcode[9];  int format;  t\_ptr link;  }t\_table;  t\_ptr t\_head;  int progaddr;  int csaddr;  char input\_program1[MAX\_INPUT\_SIZE];  char input\_program2[MAX\_INPUT\_SIZE];  char input\_program3[MAX\_INPUT\_SIZE];  //run  typedef struct \_BP\_TABLE \*bp\_ptr;  typedef struct \_BP\_TABLE{  int point\_loc;  bp\_ptr link;  }bp\_table;  bp\_ptr bp\_head;  typedef struct \_RUN\_TABLE \*run\_ptr;  typedef struct \_RUN\_TABLE{  int location;  int opcode;  int format;  int address;  int breakpoint;//if it is breakpoint, value is one  int check\_imm\_indr;//1 is imm, 2 is indr, 3 is normal  int check\_base\_relative;//1 is only base, 3 is x and base  run\_ptr link;  }run\_table;  run\_ptr run\_head; |

1. **typedef struct \_ES\_TABLE \*es\_ptr;**

이 전역변수는 다음과 같은 구조체의 포인터를 의미한다.

|  |
| --- |
| typedef struct \_ES\_TABLE{  char cs[10];  char sn[10];  int address;  int length;  es\_ptr link;  }es\_table; |

이 구조체는 여기에서 object file에 있는 여러가지 주소들과 label symbol들을 저장하고 이 코드의 길이를 저장하며, 그에 맞는 주소들을 저장한다.

1. **es\_ptr es\_head;**

이 전역변수는 위에 있던 구조체를 선언한 전역변수로, LINKED LIST의 가장 첫번째 구조체가 된다.

1. **typedef struct \_R\_TABLE \*r\_ptr;**

이 전역변수는 다음과 같은 구조체의 포인터를 나타내게 된다.

|  |
| --- |
| typedef struct \_R\_TABLE{  int number;  char symbol\_name[10];  r\_ptr link;  }r\_table; |

이는 object부분의 R(Record)부분을 저장하기 위함으로, 다른 파일에 있는 것들에 접근할 수 있도록 하기 위해 각각의 symbol에 번호를 붙여 저장해 놓은 방식이다. 항상 구조체의 첫번째는 자기 자신의 파일 첫번째 주소가 들어가게 된다.

1. **r\_ptr r\_head[3];**

이 전역변수는 위에 있던 구조체를 선언한 전역변수로, LINKED LIST의 가장 첫번째 구조체가 된다. 3개를 선언한 이유는 파일이 최대 3개 까지 들어갈 수 있기 때문이다.

1. **typedef struct \_T\_TABLE \*t\_ptr;**

이 전역변수는 다음과 같은 구조체의 포인터를 나타내게 된다.

|  |
| --- |
| typedef struct \_T\_TABLE{  int location;  char objcode[9];  int format;  t\_ptr link;  }t\_table; |

이 Linked List는 object 파일의 T부분을 저장하는 함수로, object code와 format을 저장한다고 볼 수 있다. location은 처음 T 바로 옆에 나오는 object code를 이용하여 주소를 측정한다. 또한 세번째 부분의 코드를 보고 format을 맞추게 된다. 이때, format =5 와 0은 저자가 직접 만든 byte 판독기 이다.

1. **t\_ptr t\_head;**

이 전역변수는 위에 있던 구조체를 선언한 전역변수로, LINKED LIST의 가장 첫번째 구조체가 된다.

1. **typedef struct \_BP\_TABLE \*bp\_ptr;**

이 전역변수는 다음과 같은 구조체의 포인터를 나타내게 된다.

|  |
| --- |
| typedef struct \_BP\_TABLE{  int point\_loc;  bp\_ptr link;  }bp\_table; |

이 구조체는 BREAKPOINT를 저장하기 위하여 만든 구조체로 LinkedList 형태를 띄고 있으며, bp 의 명령에서 사용되는 노드이다.

1. **bp\_ptr bp\_head;**

이 전역변수는 위에 있던 구조체를 선언한 전역변수로, LINKED LIST의 가장 첫번째 구조체가 된다.

1. **typedef struct \_RUN\_TABLE \*run\_ptr;**

이 전역변수는 다음과 같은 구조체의 포인터를 나타내게 된다.

|  |
| --- |
| typedef struct \_RUN\_TABLE{  int location;  int opcode;  int format;  int address;  int breakpoint;//if it is breakpoint, value is one  int check\_imm\_indr;//1 is imm, 2 is indr, 3 is normal  int check\_base\_relative;//1 is only base, 3 is x and base  run\_ptr link;  }run\_table; |

이는 object code를 run이라는 함수를 위해서 재 개편하기 위해 만든 구조체로, link를 통해 이어져 있다. 이때 object code만을 넣은 게 아니라 그것에 대한 모든 값들을 분석하여 만든 table이다.

1. **run\_ptr run\_head;**

이 전역변수는 위에 있던 구조체를 선언한 전역변수로, LINKED LIST의 가장 첫번째 구조체가 된다.

1. **int progaddr;**

progaddr라는 명령어를 통해 바뀌는 변수로, 이는 가장 첫번째 단위의 주소가 되며, 프로그램을 시작할 때 시작 주소 + progaddr를 하게 된다. 즉 프로그램 자체를 시작할 대 해당되는 첫번째 주소인 것이다.

1. **int csaddr;**

progaddr가 절대 바뀌지 않는다면 그것을 대신하여 주는 변수가 csaddr 함수이다. csaddr 함수가 변하면서 loader을 해줄 때 값이 변하여, 다음 파일을 다시 loader할 때도 변한 주소가 똑같이 유지 될 수 있도록한다.

1. **코드**

|  |
| --- |
| 20181703.c |
| #include "20181703.h"  int main(){  char input[MAX\_INPUT\_SIZE];  char inputing[MAX\_INPUT\_SIZE];//temporarily store the input and save the history.  char \*input\_mnemonic;//store the mnemonic input  char \*subinput;//remove the enter end of the line  char input\_range[100];//dump input  char \*input\_dump;//use pointer in strtok  char \*input\_value;//use pointer in strtok  char input\_address[100];//use in fill\_dump  int programcount;  int input\_point\_loc;  int start, end;  int address, value;  int error\_in\_asm;//if occur error in running assemble  hi\_head=NULL;  sym\_head=NULL;  opt\_head=NULL;  es\_head=NULL;  for(int i=0;i<3;i++){  r\_head[i]=NULL;  }  run\_head=NULL;  t\_head=NULL;  hi\_count=0;  dump\_count=0;  base\_operand[0] = '\0';  programlength=0;  progaddr=0;  programcount=0;  make\_op\_list();  reset\_dump();  while(1){  printf("sicsim > ");  fgets(input, MAX\_INPUT\_SIZE, stdin);  if((subinput = strchr(input, '\n'))!= NULL)\*subinput = '\0';//remove the enter end of the line  strcpy(inputing, input);//temporarily store input for history  if(strcmp(input, "quit")==0 || strcmp(input, "q")==0){  hi\_free();  op\_free();  sym\_free();  opt\_free();  t\_free();  es\_free();  r\_free();  return 0;  }  else if(strcmp(input, "help")==0 || strcmp(input, "h")==0){  printf("h[elp]\nd[ir]\nq[uit]\nhi[story]\ndu[mp] [start, end]\ne[dit] adress, value\nf[ill] start, end, value\nreset\nopcode mnemonic\nopcodelist\nassemble filename\ntype filename\nsymbol\n");  }  else if(strcmp(input, "dir")==0 || strcmp(input, "d")==0){  directory();  }  else if(strcmp(input, "history")==0 || strcmp(input, "hi")==0){  hi\_push(input);  history();//history.c  continue;  }  else if(strcmp(input, "opcodelist")==0){  print\_oplist();//opcode\_list.c  }  else if(strncmp(input, "opcode ", 5)==0){  if((input\_mnemonic = strtok(input," "))==NULL){  printf("This command is not valid. Try to command help.\n");  continue;  }  if((input\_mnemonic = strtok(NULL," "))==NULL){  printf("This command is not valid. Try to command help.\n");  continue;  }  opcode\_search(input\_mnemonic);//opcode\_search.c  }  else if(strncmp(input, "dump", 4)==0 || strncmp(input, "du", 2)==0){  if((input\_dump = strtok(input," "))==NULL){//Save the unfortune error  printf("This command is not valid. Try to command help.\n");  continue;  }  if((input\_dump = strtok(NULL," "))==NULL){//only dump  if(strcmp(input, "dump")!=0 && strcmp(input, "du")!=0){//express like dumps, dumpsdawda is not command  printf("This command is not valid. Try to command help.\n");  continue;  }  if(dump\_count>=65536){//if we went to the end of dump, return to zero position dump  dump\_count = 0;  }  print\_dump(dump\_count\*16, (dump\_count+10)\*16-1);//dump.c  dump\_count += 10;  }  else{  strcpy(input\_range,input\_dump);  if((input\_dump = strtok(NULL," "))==NULL){//dump start  start = change\_to\_dec(input\_range);  if(start == -1){  printf("Input is not hexademical.\n");  continue;//if start input is not hex, error print.  }  if(start+159 >= 1048576){//if start address close to end, overlap range is not available.  end = 1048575;  }  else{//i want use only one function to print the dump, so that we print 160 mem  end = start+159;  }  if(start >= 1048576 || start < 0){  printf("Out of Range!!\n");  continue;  }//range check  print\_dump(start, end);  }  else{//dump start, end  input\_range[strlen(input\_range)-1] = '\0';  start = change\_to\_dec(input\_range);  end = change\_to\_dec(input\_dump);  if(start == -1 || end == -1){//same as upward  printf("Input is not hexademical.\n");  continue;  }  if(start >= 1048576 || start < 0 || start > end || end >= 1048576 || end < 0){  printf("Out of Range!!\n");  continue;  }  print\_dump(start, end);  }  }  }  else if(strncmp(input, "edit", 4)==0 || strncmp(input, "e", 1)==0){  if((input\_value = strtok(input," "))==NULL){  printf("This command is not valid. Try to command help.\n");  continue;  }  if((input\_value = strtok(NULL," "))==NULL){//I want to prevent error that only command edit is input case.  printf("This command is not valid. Try to command help.\n");  continue;  }  strcpy(input\_address, input\_value);  input\_address[strlen(input\_address)-1] = '\0';  if((input\_value = strtok(NULL," "))==NULL){//same as upward  printf("This command is not valid. Try to command help.\n");  continue;  }  address = change\_to\_dec(input\_address);  value = change\_to\_dec(input\_value);  if(address == -1 || value == -1){  printf("Input is not hexademical.\n");  continue;  }  if(address >= 1048576 || value >= 256 || address < 0 || value < 0){  printf("Out of Range!!\n");  continue;  }  edit\_dump(address, value);//edit\_dump.c  }  else if(strncmp(input, "fill", 4)==0 || strncmp(input, "f", 1)==0){  //range problem exist  if((input\_value = strtok(input," "))==NULL){//I want to prevent error that only command fill is input case.  printf("This command is not valid. Try to command help.\n");  continue;  }  if((input\_value = strtok(NULL," "))==NULL){// same as upward  printf("This command is not valid. Try to command help.\n");  continue;  }  strcpy(input\_range, input\_value);  input\_range[strlen(input\_range)-1] = '\0';  if((input\_value = strtok(NULL," "))==NULL){  printf("This command is not valid. Try to command help.\n");  continue;  }  strcpy(input\_address, input\_value);  input\_address[strlen(input\_address)-1] = '\0';  if((input\_value = strtok(NULL," "))==NULL){  printf("This command is not valid. Try to command help.\n");  continue;  }  start = change\_to\_dec(input\_range);  end = change\_to\_dec(input\_address);  value = change\_to\_dec(input\_value);  if(start == -1||end == -1||value == -1){  printf("Input is not hexademical.\n");  continue;  }  if(start >= 1048576 || start < 0 || start > end || end >= 1048576 || end < 0 || value >= 256 || value < 0){  printf("Out of Range!!\n");  continue;  }  fill\_dump(start, end, value);//edit\_dump.c  }  else if(strcmp(input, "reset")==0){  reset\_dump();//dump.c  }  //homework2  else if(strncmp(input, "type", 4)==0){  input\_value = strtok(input," ");  input\_value = strtok(NULL," ");  type\_print(input\_value);  }  else if(strncmp(input, "assemble", 8)==0){  error\_in\_asm = 0;  input\_value = strtok(input," ");  input\_value = strtok(NULL," ");//filename strtok  if((error\_in\_asm = first\_assemble(input\_value))==1){  opt\_free();  continue;  }//Wrong Instruction  symbol();  if((error\_in\_asm = sort\_symbol())==1){  opt\_free();  sym\_free();  //symbol free, optable free  continue;  }//Symbol overlapping  if((error\_in\_asm = second\_assemble())==1){  opt\_free();  sym\_free();  continue;  }//undefined symbol use  print\_assemble\_lst(input\_value);  print\_assemble\_obj(input\_value);  }  else if(strcmp(input, "symbol")==0){  symbol\_print();  }  else if(strncmp(input, "progaddr", 8)==0){  input\_value = strtok(input," ");  input\_value = strtok(NULL," ");//addr input  progaddr = change\_to\_dec(input\_value);  }  else if(strncmp(input, "loader", 6)==0){  input\_program1[0] = '\0';  input\_program2[0] = '\0';  input\_program3[0] = '\0';  csaddr = progaddr;  input\_value = strtok(input," ");  input\_value = strtok(NULL," ");  strcpy(input\_program1,input\_value);  if((input\_value = strtok(NULL," "))!=NULL){  strcpy(input\_program2,input\_value);  }  if((input\_value = strtok(NULL," "))!=NULL){  strcpy(input\_program3,input\_value);  }  programcount=1;  first\_linking(programcount,input\_program1);  if(input\_program2[0]!='\0'){  programcount++;  first\_linking(programcount,input\_program2);  if(input\_program3[0]!='\0'){  programcount++;  first\_linking(programcount,input\_program3);  }  }  es\_print();  csaddr = progaddr;  second\_linking(1,input\_program1);  if(programcount==2){  second\_linking(2,input\_program2);  }  if(programcount==3){  second\_linking(2,input\_program2);  second\_linking(3,input\_program3);  }  store\_dump();  if(!strcmp(input\_program1, "copy.obj")){  make\_run\_table();  }  /\*t\_free();  es\_free();  r\_free();\*/ //not use because of run  }  else if(strncmp(input, "bp", 2)==0){  input\_value = strtok(input," ");  if((input\_value = strtok(NULL," "))==NULL){  bp\_print();  }  else{  if(strcmp(input\_value, "clear")==0){  bp\_free();  printf("\t\t[ok] clear all breakpoints\n");  }  else{  input\_point\_loc = change\_to\_dec(input\_value);  bp\_push(input\_point\_loc);  push\_bp\_in\_run(input\_point\_loc);  printf("\t\t[ok] create breakpoint %s\n", input\_value);  }  }  }  else if(strcmp(input, "run")==0){  run();  }  else{  printf("This command is not valid. Try to command help.\n");  continue;  }  hi\_push(inputing);//store history  }  } |

|  |
| --- |
| 20181703.h |
| #include<stdio.h>  #include<string.h>  #include<dirent.h>  #include<sys/stat.h>  #include<stdlib.h>  #include<math.h>  #define MAX\_INPUT\_SIZE 100  //History Data Structure  typedef struct \_HI\_LINK \*hi\_ptr;  typedef struct \_HI\_LINK{  int number;  char command[100];  hi\_ptr link;  }hi\_link;  hi\_ptr hi\_head;  int hi\_count;  //Opcode Data Structure  typedef struct \_OP\_LINK \*op\_ptr;  typedef struct \_OP\_LINK{  char opcode[3];  char mnemonic[7];  char hash[4];  op\_ptr link;  }op\_link;  op\_ptr op\_list[20];  int dump[65536][16];  int dump\_count;  typedef struct \_SYM\_TABLE \*sym\_ptr;  typedef struct \_SYM\_TABLE{  char label[7];  int locctr;  sym\_ptr link;  }sym\_table;  sym\_ptr sym\_head;  typedef struct \_OP\_TABLE \*opt\_ptr;  typedef struct \_OP\_TABLE{  int locctr;//END or BASE : -1, REMARK : -2  char label[7];  char opcode[7];  char operand[8];  char opcode\_hex[3];  int base; //if it is base, value is one  int format; //if operand need suboperand, value is one.  int char\_or\_hex;//(char=1 / hex=2)  int hex;  char bytechar[10];  char suboperand[3];  int obj\_opcode;//object  int obj\_r1;  int obj\_r2;  int obj\_xbpe;  int obj\_address;//byte address or format 3,4  int check\_modify;//if value is 1, constant value, else not  char remark\_text[256];//remark text  opt\_ptr link;  }op\_table;  opt\_ptr opt\_head;  char base\_operand[8];  int programlength;  //homework 3  typedef struct \_ES\_TABLE \*es\_ptr;  typedef struct \_ES\_TABLE{  char cs[10];  char sn[10];  int address;  int length;  es\_ptr link;  }es\_table;  es\_ptr es\_head;  typedef struct \_R\_TABLE \*r\_ptr;  typedef struct \_R\_TABLE{  int number;  char symbol\_name[10];  r\_ptr link;  }r\_table;  r\_ptr r\_head[3];  typedef struct \_T\_TABLE \*t\_ptr;  typedef struct \_T\_TABLE{  int location;  char objcode[9];  int format;  t\_ptr link;  }t\_table;  t\_ptr t\_head;  int progaddr;  int csaddr;  char input\_program1[MAX\_INPUT\_SIZE];//if the input\_program is right.  char input\_program2[MAX\_INPUT\_SIZE];  char input\_program3[MAX\_INPUT\_SIZE];  //run  typedef struct \_BP\_TABLE \*bp\_ptr;  typedef struct \_BP\_TABLE{  int point\_loc;  bp\_ptr link;  }bp\_table;  bp\_ptr bp\_head;  typedef struct \_RUN\_TABLE \*run\_ptr;  typedef struct \_RUN\_TABLE{  int location;  int opcode;  int format;  int address;  int breakpoint;//if it is breakpoint, value is one  int check\_imm\_indr;//1 is imm, 2 is indr, 3 is normal  int check\_base\_relative;//1 is only base, 3 is x and base  run\_ptr link;  }run\_table;  run\_ptr run\_head;  void directory();  void history();  void hi\_push(char \*input);  void hi\_free();  void make\_op\_list();  void op\_push(char \*opcode, char \*mnemonic, int hashnum, char \*hash);  void op\_free();  void print\_oplist();  void opcode\_search(char \*input);  int change\_to\_dec(char \*input);  void reset\_dump();  void print\_dump(int start, int end);  void edit\_dump(int address, int value);  void fill\_dump(int start, int end, int value);  void type\_print(char \*filename);  //assemble  char \*opcode\_search\_for\_asm(char \*input, int \*format);  int first\_assemble(char \*filename);  void opt\_push(int locctr, char \*label, char \*opcode, char \*operand, char \*suboperand, int base, int format, char \*opcode\_hex, int char\_or\_hex, int hex, char \*bytechar, char \*remark\_text);  void sym\_push(char \*label, int locctr);  int second\_assemble();  void print\_assemble\_lst(char \*filename);  void print\_assemble\_obj(char \*filename);  void symbol();  int sort\_symbol();  void symbol\_print();  void sym\_free();  void opt\_free();  //linnking\_loader  void first\_linking(int programcount, char \*input\_program);  void es\_push(char \*cs, char \*sn, int address, int length);  void r\_push(int programcount, int number, char \*symbol\_name);  void es\_print();  void second\_linking(int programcount, char \*input\_program);  void store\_dump();  void t\_push(int location, char \*objcode, int format);  void t\_free();  void r\_free();  void es\_free();  //run  void bp\_push(int point\_loc);  void bp\_print();  void bp\_free();  void push\_bp\_in\_run(int breakpoint);  void run();  void make\_run\_table();  void run\_push(int location, int opcode, int format, int address, int breakpoint, int check\_imm\_idr, int check\_base\_relative); |

|  |
| --- |
| linking\_pass\_1.c |
| #include "20181703.h"  void first\_linking(int programcount, char \*input\_program){  FILE \*fp;  fp = fopen(input\_program,"r");  char input[100];  char \*input\_value;  char \*subinput;//for the input  char cs[10];//control section  char sn[10];//symbol name  char cs\_main[10];//project first name  char ch\_address[9];//into char and change to decimal address(int)  int address;//change ch\_address to integer use change\_to\_dec  int length;  while(fgets(input, 99, fp)!=NULL){  cs[0]='\0';  sn[0]='\0';  ch\_address[0]='\0';  if((subinput = strchr(input,'\n'))!=NULL)\*subinput = '\0';//last array is NULL  if(input[0]=='H'){//program start point case  input\_value=strtok(input," ");  strcpy(cs,input\_value+1);  input\_value=strtok(NULL," ");  strncpy(ch\_address,input\_value,6);  ch\_address[6] = '\0';  address = change\_to\_dec(ch\_address) + csaddr;  strncpy(ch\_address,input\_value+6,6);  ch\_address[6] = '\0';  length = change\_to\_dec(ch\_address);  es\_push(cs,sn,address,length);  strcpy(cs\_main, cs);  }  else if(input[0]=='D'){//store symbol location  input\_value=strtok(input," ");  input\_value++;  while(1){  strcpy(sn,input\_value);  input\_value=strtok(NULL," ");  strncpy(ch\_address, input\_value, 6);  ch\_address[6] = '\0';  address = change\_to\_dec(ch\_address) + csaddr;  es\_push(cs,sn,address,length);  if(input\_value[6]=='\0'){  break;  }  input\_value+=6;  }  }  else if(input[0]=='R'){//store other object file symbol because of modification  r\_push(programcount, 1, cs\_main);  input\_value=strtok(input," ");  input\_value++;  while(1){  strncpy(ch\_address,input\_value,2);  ch\_address[2]='\0';  address = change\_to\_dec(ch\_address);  strcpy(sn, input\_value+2);  r\_push(programcount, address, sn);  if((input\_value=strtok(NULL," "))==NULL){  break;  }  }  }  }  csaddr += length;  return;  }  void es\_print(){  printf("control\tsymbol\taddress\tlength\n");  printf("section\tname\n");  printf("-----------------------------\n");  es\_ptr pmove;  int total\_length=0;  pmove = es\_head;  if(es\_head==NULL){  return;  }  else{  while(pmove!=NULL){  if(pmove->cs[0] == '\0'){  printf("\t%s\t",pmove->sn);  }  else{  printf("%s\t\t",pmove->cs);  }  printf("%04X\t",pmove->address);  if(pmove->cs[0] != '\0'){  printf("%04X", pmove->length);  total\_length += pmove->length;  }  printf("\n");  pmove = pmove->link;  }  }  printf("-----------------------------\n");  printf("\ttotal length %4X\n", total\_length);  }  void es\_push(char \*cs, char \*sn, int address, int length){  es\_ptr padd, pmove;  padd = (es\_table \*)malloc(sizeof(es\_table));  strcpy(padd->cs, cs);  strcpy(padd->sn, sn);  padd->address = address;  padd->length = length;  padd->link = NULL;  if(es\_head == NULL){  es\_head = padd;  }  else{  pmove = es\_head;  while(pmove->link != NULL){  pmove = pmove->link;  }  pmove->link = padd;  }  }  void r\_push(int programcount, int number, char \*symbol\_name){  r\_ptr padd, pmove;  padd = (r\_table \*)malloc(sizeof(r\_table));  padd->number = number;  strcpy(padd->symbol\_name, symbol\_name);  padd->link=NULL;  if(r\_head[programcount-1]==NULL){  r\_head[programcount-1] = padd;  }  else{  pmove=r\_head[programcount-1];  while(pmove->link != NULL){  pmove = pmove->link;  }  pmove->link = padd;  }  }  //make es free r free |

|  |
| --- |
| linking\_pass\_2.c |
| #include "20181703.h"  void second\_linking(int programcount, char \*input\_program){  FILE \*fp;  fp = fopen(input\_program,"r");  char input[255]={0};  char \*input\_value;  char \*subinput;//for the input  char store\_input[9];  int location=0;  int mod\_location;//modify location  int format;  int record\_num;  int mod\_result;  int minus\_or\_plus;//minus=0 plus=1  char record\_num\_store[3];  char text\_size[3]={0};  int count\_text\_size=0;  int count\_location;  r\_ptr pmove\_r;  t\_ptr pmove\_t;  es\_ptr pmove\_es;  while(fgets(input, 255, fp)!=NULL){  format=0;  count\_location=0;  if((subinput = strchr(input,'\n'))!=NULL)\*subinput = '\0';//change enter to NULL  input\_value = strtok(input," ");  if(input\_value[0] == 'T'){//if it is text, store the value in variable and push it.  input\_value++;  strncpy(store\_input,input\_value,6);  store\_input[6]='\0';  location=change\_to\_dec(store\_input)+csaddr;  input\_value+=6;  strncpy(text\_size,input\_value,2);  count\_text\_size = change\_to\_dec(text\_size);  input\_value+=2;  while(1){  if(input\_value[2]=='1'){//if the format is 4  format=4;  strncpy(store\_input,input\_value,8);  store\_input[8]='\0';  t\_push(location, store\_input, format);  location+=4;  input\_value+=8;  count\_location+=4;  }  else if(input\_value[2]=='2'){//if format is 3  format=3;  strncpy(store\_input,input\_value,6);  store\_input[6]='\0';  t\_push(location, store\_input, format);  location+=3;  input\_value+=6;  count\_location+=3;  }  else if(input\_value[0]=='\0'){  break;  }  else{//if format is byte  format=0;  strncpy(store\_input,input\_value,6);  store\_input[6]='\0';  t\_push(location, store\_input, format);  location+=3;  input\_value+=6;  count\_location+=3;  }  if(count\_text\_size < count\_location){  break;  }  }  }  if(input\_value[0] == 'M'){  minus\_or\_plus=0;  input\_value++;  strncpy(store\_input,input\_value,6);  store\_input[6]='\0';  mod\_location = change\_to\_dec(store\_input) + csaddr;  pmove\_t = t\_head;  while(pmove\_t != NULL){  if(pmove\_t->format == 4){  if(pmove\_t->location == mod\_location-1){  break;//because format 4 is modified just 2.5byte  }  }  else{  if(pmove\_t->location == mod\_location){  break;//find the mod\_location  }  }  pmove\_t = pmove\_t->link;  }  input\_value+=8;  if(input\_value[0]=='+'){  minus\_or\_plus=1;  }  input\_value++;  strcpy(record\_num\_store, input\_value);  record\_num=change\_to\_dec(record\_num\_store);  pmove\_r = r\_head[programcount-1];//use record table  while(pmove\_r != NULL){  if(pmove\_r->number==record\_num){  break;  }  pmove\_r = pmove\_r->link;  }  pmove\_es = es\_head;  while(pmove\_es != NULL){  if(strcmp(pmove\_r->symbol\_name,pmove\_es->sn)==0){  break;  }  else if(strcmp(pmove\_r->symbol\_name,pmove\_es->cs)==0){  break;  }  pmove\_es = pmove\_es->link;  }  if(minus\_or\_plus==1){  mod\_result = change\_to\_dec(pmove\_t->objcode) + pmove\_es->address;  }  else{  mod\_result = change\_to\_dec(pmove\_t->objcode) - pmove\_es->address;  }  if(pmove\_t->format == 4){  sprintf(pmove\_t->objcode,"%08X",mod\_result);  }  else{  if(mod\_result >= 16777216){  mod\_result -= 16777216;  }//if it is negative  sprintf(pmove\_t->objcode,"%06X",mod\_result);//store objcode  }  }  }  csaddr = location;  }  void store\_dump(){//store value of text portion in the dump  t\_ptr pmove;  pmove = t\_head;  int location;  int address;  char store\_obj[3]={0};  while(pmove != NULL){  location = pmove->location;  address = location/16;  location = location%16;  if(address == 3){  printf("%s\n", pmove->objcode);  }  if(pmove->format==4){  for(int i=0;i<4;i++){  strncpy(store\_obj, pmove->objcode+(2\*i),2);//cut objcode 2 of them  dump[address][location+i] = change\_to\_dec(store\_obj);  }  }  else{  for(int i=0;i<3;i++){  strncpy(store\_obj, pmove->objcode+(2\*i),2);  dump[address][location+i] = change\_to\_dec(store\_obj);  }  }  pmove = pmove->link;  }  }  void t\_push(int location, char \*objcode, int format){  t\_ptr padd, pmove;  padd = (t\_table \*)malloc(sizeof(t\_table));  padd->link = NULL;  padd->location = location;  strcpy(padd->objcode, objcode);  padd->format = format;  if(t\_head==NULL){  t\_head = padd;  }  else{  pmove = t\_head;  while(pmove->link != NULL){  pmove = pmove->link;  }  pmove->link = padd;  }  } |

|  |
| --- |
| run.c |
| #include "20181703.h"  void bp\_push(int point\_loc){  bp\_ptr padd, pmove;  padd = (bp\_table \*)malloc(sizeof(bp\_table));  padd->point\_loc = point\_loc;  padd->link = NULL;  if(bp\_head==NULL){  bp\_head = padd;  }  else{  pmove = bp\_head;  while(pmove->link!=NULL){  pmove = pmove->link;  }  pmove->link = padd;  }  }  void bp\_print(){//breakpoint print  bp\_ptr pmove;  printf("\t\tbreakpoint\n");  printf("\t\t----------\n");  pmove = bp\_head;  while(pmove != NULL){  printf("\t\t%X\n",pmove->point\_loc);  pmove = pmove->link;  }  }  void bp\_free(){  bp\_ptr pfree;  if(bp\_head == NULL){  return;  }  else{  while(bp\_head != NULL){  pfree = bp\_head;  bp\_head = bp\_head->link;  free(pfree);  }  }  return;  }  void run(){  int reg1;//만약 register을 이용하는 opcode가 있으면 이 변수 이용  int breakpoint;  int end\_program = 0;//만약 프로그램이 끝나면 값이 1이 된다.  int a=0,x=0,l,pc,b=0,s=0,t=0,cc;//cc is compare varialble, so 0 is "=", 1 is "<", 2 is ">"  l = es\_head->length;  pc = es\_head->address;//값 초기화  run\_ptr pmove, pmove\_store;  int length\_address=0;  int buffer\_address=0;  while(1){//or pc counter is overflow  pmove = run\_head;  if(pc == es\_head->length){  end\_program = 1;  break;  }//만약 pc가 끝까지 간다면 exit하게 된다.  while(pmove != NULL){  if(pmove->location == pc){  break;  }  pmove=pmove->link;  }  if(pmove == NULL){  return;//error가 발생했을 경우 즉, location이 잘못 입력된 경우등을 의미함.  }  if(pmove->breakpoint == 1){  pmove->breakpoint = 0;  breakpoint = pmove->location;  break;  }//if it is breakpoint  if(pmove->format == 0){  pc += 3;  continue;  }//byte인 경우 "EOF"  else if(pmove->format == 5){//bytesize is just 1  pc += 1;  continue;  }  if(pmove->format == 2){  if(pmove->opcode==180){//CLEAR  reg1 = pmove->address;  reg1 = reg1/16;  if(reg1 == 0) a=0;  else if(reg1 == 1) x=0;  else if(reg1 == 2) l=0;  else if(reg1 == 3) b=0;  else if(reg1 == 4) s=0;  else if(reg1 == 5) t=0;  else if(reg1 == 8) pc=0;  pc += 2;  }  else if(pmove->opcode==160){//COMPR  cc = 0;  pc += 2;  }  else if(pmove->opcode==184){//TIXR  x += 1;  reg1 = pmove->address;  reg1 = reg1/16;  if(reg1 == 0) a=0;  else if(reg1 == 1) reg1 = x;  else if(reg1 == 2) reg1 = l;  else if(reg1 == 3) reg1 = b;  else if(reg1 == 4) reg1 = s;  else if(reg1 == 5) reg1 = t;  else if(reg1 == 8) reg1 = pc;  if(x > reg1) cc = 2;  else if( x < reg1) cc = 1;  else cc = 0;  pc += 2;  }  }  else if(pmove->format == 4){  if(pmove->opcode == 72){//JSUB  pc += 4;  l = pc;  pc = pmove->address;  }  else if(pmove->opcode == 116){  t = pmove->address;  pc += 4;  }//LDT  else{  pc += 4;  }  }//not pc relative so don't care about it  else if(pmove->format == 3){  if(pmove->opcode == 104){//LDB  b = pmove->address + pmove->location + 3;  pc += 3;  }  else if(pmove->opcode == 0){//LDA  a = pmove->address + pmove->location + 3;  if(pmove->address == 38){  a = length\_address;  }  else if(pmove->address == 16){  a = 4542278;  }//EOF의 값을 의미함  else if(pmove->address == 3){  a = 3;  }  pc += 3;  }  else if(pmove->opcode == 40){//COMP  if(a > pmove->address + pmove->location + 3) cc = 2;  else if(a < pmove->address) cc = 1;  else cc = 0;  pc += 3;  }  else if(pmove->opcode == 48){  if(cc==0){  pc = pmove->address + pmove->location + 3;  if(pc >= 8192){  pc -= 4096;  }//if it is negative  }  else{  pc += 3;  }  }//JEQ  else if(pmove->opcode == 60){  pc = pmove->address + pmove->location + 3;  if(pc >= 4096){  pc -= 4096;  }//if it is negative  if(pmove->address == 3){  pc = es\_head->length;  end\_program = 1;  break;  }//if the code is end. because i didn't use dump, so this case is just use pc change to length when j address have RETADR  }//J  else if(pmove->opcode == 224){  cc=1;  pc += 3;  }//TD(설명에 >로 설정하라고 나옴  else if(pmove->opcode == 56){  if(cc==1){  pc = pmove->address + pmove->location +3;  if(pc >= 8192){  pc -= 4096;  }//if it is negative  }  else{  pc += 3;  }  }//JLT  else if(pmove->opcode == 76){  pc = l;  }//RSUB  else if(pmove->opcode == 116){  t = pmove->address + pmove->location +3;  if(pmove->address == 0){  t = length\_address;  }  pc += 3;  }//LDT  else if(pmove->opcode == 80){  if(x==0){  a = buffer\_address/(16\*16\*16\*16);  }  else if(x==1){  a = buffer\_address%(16\*16\*16\*16)/(16\*16);  }  else if(x==2){  a = buffer\_address%(16\*16);  }  pc += 3;  }//LDCH  else if(pmove->opcode == 16){//STX  pmove\_store = run\_head;  if(pmove->check\_base\_relative == 1){  while(pmove\_store != NULL){  if(pmove\_store->location == b){  length\_address = x;  }  pmove\_store = pmove\_store->link;  }  }  pc += 3;  }  else if(pmove->opcode == 12){//STA  if(pmove->address == 22){//if buffer address  buffer\_address = a;  }  else if(pmove->address == 13){  length\_address = a;  }  else if(pmove->address == 35){  buffer\_address = a;  }  pc += 3;  }  else{  pc += 3;  }  }  }  printf("A : %06X\tX : %06X\nL : %06X\tPC : %06X\nB : %06X\tS : %06X\nT : %06X\n", a,x,l,pc,b,s,t);  if(end\_program == 1){//if the program is end  printf("\t\tEnd Program\n");  }  else{  printf("%d %d\n", pc, es\_head->length);  printf("\t\tStop at checkpoint[%X]\n",breakpoint);  }  return;  }  void make\_run\_table(){  FILE \*fp;  fp = fopen(input\_program1,"r");    char \*input;  input = (char \*)malloc(100\*sizeof(char));  char \*subinput;//for the input, Enter is Change NULL  int location;  char store\_loc[7]={0};//int와 char은 한 쌍으로, char형을 int형으로 바꿔주는 역할을 하게되는 변수  char store\_address[7]={0};  int address;  char store\_opcode[3]={0};  int opcode;  char check\_format\_store[2]={0};  int check\_format;  int check\_imm\_indr;  int check\_base\_relative;  int before\_location;//it use when push the run\_table  int breakpoint;  int check\_byte=0;//if 0, byte is not exist, 1 is exist  location = es\_head->length;//set location  while(fgets(input, 99, fp)!=NULL){  //TODO location break  if((subinput = strchr(input,'\n'))!=NULL)\*subinput = '\0';//last array is NULL  check\_byte = 0;  if(input[0]=='T'){  input++;  strncpy(store\_loc,input,6);  location = change\_to\_dec(store\_loc);  input+=8;  while(1){  before\_location = location;  breakpoint = 0;  check\_imm\_indr = 0;  check\_base\_relative = 0;  if(input[0] == '\0'){  break;  }  if(strncmp(input,"454F46",6)==0){  strncpy(store\_address,input,6);  address=change\_to\_dec(store\_address);  check\_format = 0;  run\_push(location, opcode, check\_format, address, breakpoint,check\_imm\_indr,check\_base\_relative);  input+=6;  location+=3;  continue;  }//except EOF byte  else if(check\_byte == 1){  strncpy(store\_address,input,2);  address=change\_to\_dec(store\_address);  check\_format = 5;  run\_push(location, opcode, check\_format, address, breakpoint,check\_imm\_indr,check\_base\_relative);  input+=2;  location+=1;  check\_byte=0;  continue;  }//except the byte;  strncpy(store\_opcode,input,2);  opcode = change\_to\_dec(store\_opcode);  check\_imm\_indr = opcode%4;  opcode = opcode - check\_imm\_indr;  input+=2;  if(opcode == 180){//CLEAR  strncpy(store\_address,input,2);  store\_address[2]='\0';  address = change\_to\_dec(store\_address);  check\_format = 2;  input += 2;  location += 2;  }  else if(opcode == 160){//COMPR  strncpy(store\_address,input,2);  store\_address[2]='\0';  address = change\_to\_dec(store\_address);  check\_format = 2;  input += 2;  location += 2;  }  else if(opcode == 184){//TIXR  strncpy(store\_address,input,2);  store\_address[2]='\0';  address = change\_to\_dec(store\_address);  check\_format = 2;  input += 2;  location += 2;  }//format 2  else{//format 3,4 or byte  check\_format\_store[0] = input[0];  check\_format = change\_to\_dec(check\_format\_store);  check\_base\_relative = check\_format/4;  input++;  if(check\_format%2==1){  strncpy(store\_address, input, 5);  store\_address[5]='\0';  address = change\_to\_dec(store\_address);  check\_format = 4;  location += 4;  input += 5;  }  else{  strncpy(store\_address,input,3);  store\_address[3]='\0';  address = change\_to\_dec(store\_address);  check\_format = 3;  location += 3;  input += 3;  if(opcode == 76){  check\_byte = 1;  }//RSUB because byte exist after the RSUB  }  }  run\_push(before\_location, opcode, check\_format, address,breakpoint,check\_imm\_indr,check\_base\_relative);  }  }  }  return;  }  void run\_push(int location, int opcode, int format, int address, int breakpoint, int check\_imm\_indr, int check\_base\_relative){  run\_ptr padd, pmove;  padd = (run\_table \*)malloc(sizeof(run\_table));  padd->location = location;  padd->opcode = opcode;  padd->format = format;  padd->address = address;  padd->breakpoint = breakpoint;  padd->check\_imm\_indr = check\_imm\_indr;  padd->check\_base\_relative = check\_base\_relative;  padd->link = NULL;  if(run\_head == NULL){  run\_head = padd;  }  else{  pmove = run\_head;  while(pmove->link != NULL){  pmove = pmove->link;  }  pmove->link = padd;  }  }  void push\_bp\_in\_run(int breakpoint){  run\_ptr pmove;  pmove = run\_head;  while(pmove != NULL){  if(pmove->location == breakpoint){//find location of breakpoint  pmove->breakpoint = 1;  }  pmove = pmove->link;  }  return;  } |

|  |
| --- |
| free.c |
| #include "20181703.h"  void sym\_free(){//free symbol table memory  sym\_ptr pfree;  if(sym\_head == NULL){  return;  }  else{  while(sym\_head!=NULL){  pfree = sym\_head;  sym\_head = sym\_head->link;  free(pfree);  }  }  return;  }  void opt\_free(){//free opcode table memory  opt\_ptr pfree;  if(opt\_head == NULL){  return;  }  else{  while(opt\_head!=NULL){  pfree = opt\_head;  opt\_head = opt\_head->link;  free(pfree);  }  }  return ;  }  void t\_free(){  t\_ptr pfree;  if(t\_head == NULL){  return;  }  else{  while(t\_head!=NULL){  pfree = t\_head;  t\_head = t\_head->link;  free(pfree);  }  }  return;  }  void r\_free(){  r\_ptr pfree;  for(int i=0;i<3;i++){  if(r\_head[i] == NULL){  continue;  }  else{  while(r\_head[i]!=NULL){  pfree = r\_head[i];  r\_head[i] = r\_head[i]->link;  free(pfree);  }  }  }  return;  }  void es\_free(){  es\_ptr pfree;  if(es\_head == NULL){  return;  }  else{  while(es\_head!=NULL){  pfree = es\_head;  es\_head = es\_head->link;  free(pfree);  }  }  return;  } |