이 코드에서 가장 많이 쓰이는 부분은 SubString함수이다

char\* SubString(char\* input, int begin, int end)

{

char\* substring = (char\*)malloc(sizeof(char) \* (end - begin + 2));

strncpy(substring, input + begin, end - begin + 1);

substring[end - begin + 1] = 0;

return substring;

}

이 함수는 입력값으로 들어온 문자열 input에서 시작과 끝을 매개변수로 받아 그 잘라진 문자열을 반환한다. 주어진 objfile에서 코드 길이를 가져올 때, 시작주소와 길이를 제외한 실질적인 코드부분을 가져올 때 주로 사용했다

첫번째는 memory storage capacity, start address , last address를 어떻게 구현하느냐이다

capacity는 처음 파일이 시작했을 때 읽어온줄. 즉 첫번째 줄에서 SubString함수를 이용해서 읽어오면 된다. 그 후 atoi함수로 문자열을 10진수 정수로 바꿔주면된다.

start address는 rand()를 이용해 랜덤하게 초기화하면 되고

last address를 구하려면 목적코드의 길이를 알아야한다. 다행히도 T로 시작하는 문자열의 7~8번째가 이 목적코드의 길이를 나타내는 16진수 정수이다. 따라서 T로 시작하는 문자열에 대해서 7~8번째를 SubString으로 자른 문자열을 16진수로 변환해서 다 더해준다. 이 값을 count라고 하자. last address는 start address + count - 1이다

두번째는 수정레코드(M으로 시작하는 줄)에서 가리키는 부분을 수정해줘야한다. 일단 수정레코드에서 가리키는 부분은 locations라는 부분에 저장해놓자. 그리고 locations 배열의 길이도 필요하다.

수정레코드M에서 가리키는 부분은 “전체 목적코드” 즉, 명령코드(17202D 이런거)들을 한줄로 작성했을 때 그 위치이므로 먼저 명령코드들을 하나로 합쳐줘야한다. 그게 all\_instruction이다.

all\_instruction에 대해서 M에 해당하는 주소의 명령어를 바꿔줘야한다.

4B101036 >> 4B10190a(시작주소를 2260이라 가정했을때)로 말이다. 이에 해당하는 부분이

int\* relocation\_addr\_int = (int\*)malloc(sizeof(int) \* location\_count);

char relocation\_addr\_string[3][7];

for (int i = 0; i < location\_count; i++)

{

relocation\_addr\_int[i] = strtol(SubString(all\_instruction, locations[i] \* 2, locations[i] \* 2 + 5), NULL, 16) + start\_addr;

sprintf(relocation\_addr\_string[i], "%X", relocation\_addr\_int[i]);

relocation\_addr\_string[i][6] = '\0';

}

이다. strtol을 이용해 해당부분의 문자열을 16진수 정수로 바꿔주었고 이에 시작주소 start\_addr을 더해서 101036 >> 10190a로 바꾸었다. M으로 시작한 횟수만큼 relocation이 이루어져야한다. 이제 relocation\_addr\_int에 16진수 정수가 저장되었으니 이를 다시 문자열로 바꿔야한다. 이것이 sprintf문이다. 바꾼 문자열은 relocation\_addr\_string에 저장해줘야하는데 여기서 주의할 점이 문자열의 마지막은 항상 ‘\0’(널문자)여야한다는 것이다.

이제 all\_instruction에서 변환된부분을 바꿔주자. 그게 밑의 코드다

for (int i = 0; i < all\_length; i++)

{

for (int j = 0; j < location\_count; j++)

{

if (i == locations[j] \* 2)

{

for (int k = 0; k < 6; k++)

{

all\_instruction[i + k] = relocation\_addr\_string[j][k];

}

}

}

}

여기서 i는 all\_instruction의 문자하나하나를 가리키는 인덱스라 볼 수 있다.

if( i == locations[j] \* 2) 부분은 변환되야하는 부분에 대한 조건이고

all\_instruction[i+k] = relocation\_addr\_string[j][k]부분이 실질적으로 변환이 이루어지는 곳이다.

해당부분의 문자가 하나하나씩 바뀐다. 예를 들어 4B101036에서 처음1부터 끝의 6까지가 변해야하는데, 이 처음1에 해당하는 위치가 locations[j] \* 2(14)이다. 이제 두번째 1이 바뀌려면 all\_instruction[14+2](k가 2)이 relocation\_addr\_string에 해당하는 문자로 바뀌게 되는 것이다.

이제 all\_instruction이 모두 변환되었다. 다음은 파일입력부분이다

for (int i = 0; i < all\_length; i++)

{

int instruction = strtol(SubString(all\_instruction, 0, 1), NULL, 16);

fprintf(after\_relocation, "%d %02X\n", address, instruction);

all\_instruction = SubString(all\_instruction, 2, all\_length);

all\_length = strlen(all\_instruction);

address += 1;

}

변환된 all\_instruction에 대해서 2글자씩 끊어서 만든 16진수 정수가 주소하나에 할당된다. 예를 들어 17202D는 17,20,2D로 나뉘는데 주소2260에 17, 2261에 20, 2262에 2D이런식으로 말이다.

all\_instruction은 문자열이므로 SubString을 이용해 두글자씩 끊어오고 이를 strtol로 16진수 정수로 바꿔주자. 그 후. fprintf로 파일에 입력해준다

그리고 나서 파일에 입력된 부분은 제외해서 남은 부분에 대해서 all\_instruction을 다시 만들어준다. 다시 만들어주려면 all\_length도 바뀌어야한다.

예를 들어 all\_instruction이 17202D4B10190A라면 위의 for문을 한번 돌면 instruction에 17이 저장되고 파일에 17과 그 address가 입력된다. 그 후 all\_instruction은 202D4B10190A가 된다. 다시 for문을 돌면 이제 instruction에는 20이 들어가고 파일에 20과 그 address가 입력된다. 그 후 all\_instruction은 남은 부분인 2D4B10190A가 된다. 이를 끝까지 반복해주면 된다.

남은 부분은 터미널창에 before, after를 출력하는 것인데

printf("before modification:\n");

printf("Address instruction\n");

for (int i = 0; i < location\_count; i++)

printf("%d %6X\n", start\_addr + locations[i], relocation\_addr\_int[i] - start\_addr);

printf("\n");

printf("after modification:\n");

printf("Address instruction\n");

for (int i = 0; i < location\_count; i++)

printf("%d %6X\n", start\_addr + locations[i], relocation\_addr\_int[i]);

before부분은 주소에 locations[i](즉, 변환되야 하는 부분)를 더했고, relcation\_addr\_int[i]에 대해서는 시작주소를 빼야한다(relocation\_addr\_int는 시작주소를 더해 변환된 부분이기 때문)

after부분에선 주소는 동일하고 relocation\_addr\_int는 그대로 써준다.