

# Cykor 1주차 과제 탐구 보고서

# 목차

가. 사용된 헤더파일.....	3
나. 초기 선언된 배열들.....	3
다. push().....	4
라. pop().....	4
마. func_prologue().....	5
바. func_epilogue().....	6

## 가. 사용된 헤더파일(23~24)

작성 코드	<pre> 23  #include &lt;stdio.h&gt; 24  #include &lt;stdarg.h&gt; </pre>
설명	<p>1. #include &lt;stdio.h&gt; : 표준 입출력 함수들, 예: printf, scanf, fgets 등</p> <p>2. #include &lt;stdarg.h&gt; : 가변 인자 함수를 작성할 때 사용하는 C 표준 라이브러리, 예: va_list 등</p>

## 나. 초기 선언된 배열들(27~32)

작성 코드	<pre> 27  int    call_stack[STACK_SIZE];           // Call Stack을 저장하는 배열 28  char    stack_info[STACK_SIZE][20];      // Call Stack 요소에 대한 설명을 저장하는 배열 29  int     func_values_info[4][3] = {{0,0},{3,1,1},{2,1,2}, {1,2,3}}; // 함수들의 초기, 지역 30  char     beginning_values_name_set[4][20] = { "Return Address", "arg1", "arg2", "arg3" }; 31  char     func_name_set[4][10] = { "null", "func1 SFP", "func2 SFP", "func3 SFP" }; 32  char     local_value_name_set[5][20] = { "null", "var_1", "var_2", "var_3", "var_4" }; 33 </pre>
설명	<p>29: func_values_info[4][3], 함수들의 초기, 지역 변수들의 개수, local_value_name_set 처음 위치 저장, object file의 symbol table 같은 역할을 한다. ex) func_values_info[2][1] = fun2의 지역변수 개수</p> <p>30. beginning_values_name_set[4][20], 함수들의 초기변수들 이름을 저장</p> <p>31. func_name_set[4][10], 함수들의 이름에 SFP를 붙여서 각 함수의 고유번호에 맞는 위치에 저장. func1의 고유번호 1, func2의 고유번호 2, func3의 고유번호 3이다.</p> <p>32. local_value_name_set[5][20], 함수들의 지역변수들 이름을 저장</p>

## 다. push()(81~87)

작성 코드	<pre> 81      void push(char name[20], int value) { 82          SP += 1; 83          for (int i = 0; name[i] != 0; i++) { 84              stack_info[SP][i] = name[i]; 85          } 86          call_stack[SP] = value; 87      }</pre>
설명	<p>82: SP 위치를 1 증가시킨다.</p> <p>83~85: stack_info에 value의 정보를 넣는다.</p> <p>87: call_stack에 value값을 저장한다.</p>

## 라. pop()(89~97)

작성 코드	<pre> 89      void pop() { 90          for (int i = 0; stack_info[SP][i] != 0; i++) { 91              stack_info[SP][i] = 0; 92          } 93          call_stack[SP] = 0; 94          SP -= 1; 95      }</pre>
설명	<p>91~93: stack_info에서 있는 가장 끝에 저장되어있는 정보 제거.</p> <p>94: call_stack에 있는 가장 끝에 존재하는 값 제거.</p> <p>96: SP의 위치를 1 감소시킨다.</p>

## 마. func\_prologue()(99~118)

<p><b>작성 코드</b></p>	<pre> 99  void func_prologue(int fun_num, ...) { 100 101      va_list values; 102      va_start(values, fun_num); 103 104      for (int i = func_values_info[fun_num][0]; i &gt;= 0; i--) { 105          int beginning_value = (i != 0) ? va_arg(values, int) : -1; 106          push(beginning_value_name_set[i], beginning_value); 107      } 108 109      push(func_name_set[fun_num], (FP == -1) ? -1 : FP); 110      FP = SP; 111 112      for (int i = 0; i &lt; func_values_info[fun_num][1]; i++) { 113          int loc_num = i + func_values_info[fun_num][2]; 114          push(local_value_name_set[loc_num], va_arg(values, int)); 115      } 116 117      va_end(values); 118  }</pre>
<p><b>설명</b></p>	<p>101~102: 가변 인자 목록 포인터를 선언, 설정한다.</p> <p>104~107: 함수 고유번호에 맞는 초기변수 개수를 불러와 개수만큼 반복문 실행, 이렇게 되면 함수가 받아온 값에서 초기변수들만 받을 수 있다. 함수가 받아온 초기변수들의 값을 va_arg()를 통해 불러오고, 이 값과 변수에 대한 정보를 push()를 통해 call_stack[], stack_info[]에 넣는다.</p> <p>109~110: 함수의 SFP를 push()를 통해 stack에 저장하고, FP 위치를 현재 SP 위치로 바꾼다.</p> <p>112~115: 함수 고유번호에 맞는 지역변수 개수를 불러와 개수만큼 반복문 실행, 이렇게 되면 함수가 받아온 값에서 지역 변수들만 받을 수 있다. 함수가 받아온 지역 변수들의 값을 va_arg()를 통해 불러오고, 이 값과 정보를 push()를 통해 call_stack[], stack_info[]에 넣는다.</p> <p>117: 가변 인자 목록 포인터를 NULL로 초기화한다.</p>

## 바. func\_epilogue()(120~127)

<p><b>작성 코드</b></p>	<pre> 120  void func_epilogue(int fun_num) 121  { 122      int total = func_values_info[fun_num][0] + func_values_info[fun_num][1] + 2; 123      for (int i = 1; i &lt;= total; i++) { 124          pop(); 125          if (i == func_values_info[fun_num][1]) FP = call_stack[SP]; 126      } 127  } 128 </pre>
<p><b>설명</b></p>	<p>122: 고유번호가 fun_num이랑 같은 함수의 stack 수를 계산한다. 이때 +2는 변수들의 stack 개수에 Return Adress, SFP의 스택 개수를 더한 것이다.</p> <p>123~126: pop()을 이용해서 고유번호가 fun_num이랑 같은 함수의 stack을 제거한다. 또한 SFP에 저장된 위치로 FP 위치를 이동시킨다.</p>