**고급시스템프로그래밍 5주차 실시간수업 과제**

|  |
| --- |
| 제출일: 2021.04.08  이름: 김현용 |

1. 다음 프로그램은 N과 M에 대한 #define 값을 제거한 C 프로그램과 이를 컴파일한 어셈블리 코드의 sum\_element 함수 부분이다. 어셈블리 코드를 참고하여 M과 N값을 결정하시오.

|  |
| --- |
| #define N  #define M  int sum\_element(int a[][N], long i, long j) {  return a[i][j]+a[j][i];  }  int main() {  int arr[M][N];  printf("%d\n", sum\_element(arr, 3,5));  } |
| sum\_element:  .LFB0:  pushq %rbp  movq %rsp, %rbp  movq %rdi, -8(%rbp)  movq %rsi, -16(%rbp)  movq %rdx, -24(%rbp)  movq -16(%rbp), %rax  salq $2, %rax  leaq 0(,%rax,8), %rdx  subq %rax, %rdx  movq -8(%rbp), %rax  addq %rax, %rdx  movq -24(%rbp), %rax  movl (%rdx,%rax,4), %edx  movq -24(%rbp), %rax  salq $2, %rax  leaq 0(,%rax,8), %rcx  subq %rax, %rcx  movq -8(%rbp), %rax  addq %rax, %rcx  movq -16(%rbp), %rax  movl (%rcx,%rax,4), %eax  addl %edx, %eax  popq %rbp  ret |

**A1-1) N = 7 M > 5**

**A1-2) 설명**  
먼저, c코드를 보면 sum\_element 함수는 배열과 i, j 값을 받아서 a[i][j]+a[j][i]를 return하는 함수이다. 그러므로 어셈블리 코드 안에서는 a[i][0]의 주소 값을 먼저 찾고, a[i][j]를 찾아가는 방식일 것이다. 어셈블리 코드를 보면 rsp는 스택 포인터이고, rdi = arr의 시작 주소, rsi = 3(i값), rdx = 5(j값) 인 것을 알 수 있고, 이 후  
 movq -16(%rbp), %rax ;rax = i

salq $2, %rax ;rax = 4i

leaq 0(,%rax,8), %rdx ;rdx = 8(4i)=32i

subq %rax, %rdx ;rdx = rdx-rax = 32i-4i = 28i

movq -8(%rbp), %rax ;rax = arr의 시작 주소

addq %rax, %rdx ;rdx = rdx+rax = arr[i][0] 주소 값

movq -24(%rbp), %rax ;rax = j

movl (%rdx,%rax,4), %edx ;edx = rdx + 4\*rax = arr[i][j]  
임을 알 수 있다. 이 때 28i에 주목을 해야 하는데, 이는 행이 한 개 증가하려면 4\*7만큼 증가해야 한다는 의미이므로 N은 7이라는 것을 알 수 있다. 이 후 , 코드도 똑같이 반복하여 arr[j][i]를 찾고, edx와 eax를 더하는 것을 알 수 있다.  
 그리고 M은 위의 어셈블리 코드만 봐서는 정확하게 몇인 지 알 수 없고, a[5][3]이 존재하는 값이기 때문에, 단순히 M>5라고만 유추할 수 있다.

2. 다음 C 파일과 이를 컴파일 한 첨부의 5-2.s 파일을 참고하여 물음에 답하시오.

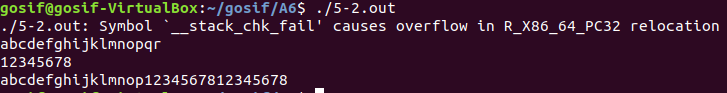
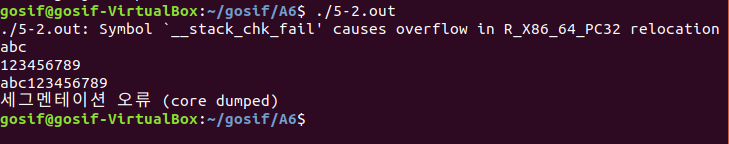
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  void f() {  char s1[4];  char s2[4];  char \*result;  gets(s1);  gets(s2);  result = malloc(strlen(s1)+strlen(s2));  strcat(result,s1);  strcat(result,s2);  printf("%s\n", result);  }  int main() {  f();  } |

1. 위 프로그램에서 함수 f에서 첫 gets 함수 호출 직전의 스택 프레임의 상태를 그림으로 그리시오.

**A2-1) 스택 상태**  
top rbp rbp-32 rbp-48 rsp

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| rbp | <ret> | rbp | rbx | UN | rax(canary) | UN | UN | UN | UN | UN |

(#UN = UNUSED)

1. 위 프로그램에서 입력으로 abcdefghijklmnopqr과 12345678을 입력한 결과의 원인을 설명하시오.  
   **A2-2) 결과 및 설명**  
     
   **설명** : 결과를 보면 입력한 qr은 없어지고 12345678은 두번 반복하여 출력된 것을 알 수 있다. 이는 먼저 어셈블리 코드를 보면 gets 함수를 call하기 전에,   
   leaq -48(%rbp), %rax 를 수행하는 것을 확인할 수 있는데, 이는 위의 스택 상태에서 rbp-48 구간의 UNUSED에 저장한다는 것을 알 수 있고, 두번 째 gets 함수를 call하기 전에는 rbp-32 구간의 UNUSED에 저장한다는 것을 알 수 있다. 그렇기 때문에 s1이 18비트를 입력해 s2의 영역까지 침범했고, 그래서 16비트는 보존이 되고 이후의 qr은 s2가 저장되면서 삭제되었다. 또한, s2가 두 번 출력된 이유는 string은 끝을 알리는 \0(널문자)도 s2 영역을 침범하면서 삭제되었기 때문에, s2의 끝의 널문자까지 인식 해버려서이다. 따라서 s1 = abcdefghijklmnopqr12345678, s2 = 12345678로 인식된 것이다.
2. 위 프로그램에서 입력으로 abc와 123456789를 입력하면 프로그램이 비정상적으로 종료한다. 그 원인을 설명하시오.  
   **A2-3) 결과 및 설명**  
     
   **설명 :** 이것도 위의 스택 상태를 보면 rbp-24 구간에 랜덤한 시스템 값 rax(canary bit)이 저장되어 있는 것을 확인할 수 있고, 어셈블리 코드 f 함수 마지막에 canary bit가 변했는 지 확인하고, 달라졌으면 call \_\_stack\_chk\_fail 을 하도록 되어 있다. 그렇기 때문에, s2를 저장하면서 8비트 이상 저장해버렸기 때문에, canary bit이 변해버려서 세그멘테이션 오류가 발생한 것이다.  
     
   \*\*2-2)에서 s2에 12345678만 해도 널문자 때문에 세그멘테이션 오류가 떠야하는데 그렇지 않았다. 이는 저장한 canary bit의 끝이 00으로 끝나기 때문이라고 유추할 수 있다.