**고급시스템프로그래밍 4주차 실시간수업 과제**

|  |
| --- |
| 제출일: 2021.04.01  이름: 김현용 |

1. 프로그램과 어셈블리 코드에 대하여 물음에 답하시오.

|  |
| --- |
| #include <stdio.h>  int inc(int x, int y) {  register int s = 0;  while (\_\_\_\_\_\_\_\_) {  if (\_\_\_\_\_\_\_\_)  s = \_\_\_\_\_\_\_\_\_;  else  s = s+1;  x++;  }  return s;  }  main() {  printf("%d\n\n", inc(1, 100));  } |
| inc:  .LFB0:  pushq %rbp  movq %rsp, %rbp  pushq %rbx  movl %edi, -12(%rbp)  movl %esi, -16(%rbp)  movl $0, %ebx  jmp .L2  .L5:  movl -12(%rbp), %eax  andl $1, %eax  testl %eax, %eax  jne .L3  addl $2, %ebx  jmp .L4  .L3:  addl $1, %ebx  .L4:  addl $1, -12(%rbp)  .L2:  movl -12(%rbp), %eax  imull -12(%rbp), %eax  cmpl -16(%rbp), %eax //x^2-y (1,2, …10)  jl .L5  movl %ebx, %eax  popq %rbx  popq %rbp  ret |

1. 어셈블리 코드를 참고하여 C 프로그램을 완성하시오.

**A11-1) 소스 코드**

#include <stdio.h>

int inc(int x, int y) {

register int s = 0;

while (x\*x<y) {

if ( x % 2 == 0)

s = s+2;

else

s = s+1;

x++;

}

return s;

}

void main() {

printf("%d\n\n", inc(1, 100));

}

**A11-2) 결과 및 코드 설명**

결과 : 150 출력

1) while 조건 : 가장 아래의 .L2 에서  
 movl -12(%rbp), %eax

imull -12(%rbp), %eax

cmpl -16(%rbp), %eax

jl .L5  
를 보고 eax과 (rbp-16) 값을 비교하는데 이 때, 코드를 읽다 보면 rbp-16은 y임을 알 수 있고, eax는 (rbp-12) 이고 이는 인자 x임을 알 수 있다. 그리고 cmpl b a \n jl 은 a<b 일 때 점프를 하는 것이므로 x\*x <y 일 때 jump가 되는 것이다. 따라서 while 조건문의 내용도 x\*x<y인 것이다.  
2)if 조건 :   
andl $1, %eax

testl %eax, %eax  
위를 보면 1과 eax(=(rbp-12) = 인자 x) and 연산을 하는 것인데 여기서 1은 bit로 일의 자리만 1인 숫자이므로 eax의 bit에서 일의 자리가 1이면 1이 저장되고, 0이면 0이 저장된다. 이는 짝수, 홀수 판단인 것을 알 수 있다. 따라서 if 조건문은 x의 짝수 판단을 하는 x % 2 == 0 인 것이다.  
3) if문 참 수행문 :  
jne .L3

addl $2, %ebx  
위를 보면 jne는 ~ZF가 1이면 점프를 하는 것인데 이는 이전의 testl로 set이 된다. testl에서는 a&b ==0일 경우, ZF가 1로 set되는데 이는 짝수일 경우이고, ~ZF가 1이려면 홀수여야 하는 것이다. 따라서 jne문에서는 eax가 홀수일 경우 점프를 한다.  
addl 문은 단순히 ebx = ebx + 2인 것이므로 .L3와 비교하면 s = s+2인 것이다.

1. 어셈블리 코드의 inc 함수를 conditional move 명령어를 사용하여 동일한 결과를 내는 함수로 고치시오. (붉은 점프 문 제거)

**A12-1) 소스 코드**

.L5:

movl -12(%rbp), %eax

movl %ebx, %ecx

addl $2, %ebx

addl $1, %ecx

andl $1, %eax

testl %eax, %eax

cmovne %ecx, %ebx

addl $1, -12(%rbp)

.L2: ;(다른 코드는 변동 사항 없음)

**A12-2) 코드 설명**

conditional move 명령어는 미리 조건문 이후의 계산의 결과 값을 저장해 놓고 조건 문으로 판단 후 적용하는 것이므로 위와 같이 새로운 레지스터 ecx를 추가시켜 else의 계산 결과를 ecx에 저장하고 if의 계산 결과를 ebx에 저장한 뒤, cmovne를 통해 위의 testl로 짝수 홀수 판단 결과를 이용해 ~ZF가 1이면 ebx를 ecx로 저장하고, 아닌 경우는 그대로 ebx를 사용하는 것이다.

2. 다음 C 프로그램 2.c와 이에 대하여 gcc -S -o1 2.c로 컴파일한 어셈블리 코드에 대하여 물음에 답하시오. (일부 디렉티브는 삭제함)

|  |
| --- |
| #include <stdio.h>  int f(int x1, int x2, int x3, int x4, int x5) {  int temp;  temp = x1+x2+x3+x4+x5;  if (x1>x2)  return temp;  else  return temp + (f(x1+1, x2-1, x3, x4, x5) + x1);  }  int main() {  printf("%d\n", f(1,2,3,4,5));  } |
| .file "2.c"  .text  .globl f  .type f, @function  f:  .LFB23:  pushq %rbp  pushq %rbx  subq $8, %rsp  leal (%rdi,%rsi), %ebx  addl %edx, %ebx  addl %ecx, %ebx  addl %r8d, %ebx  movl %ebx, %eax  cmpl %esi, %edi  jg .L2  movl %edi, %ebp  subl $1, %esi  leal 1(%rdi), %edi  **call f**  addl %ebp, %eax  addl %ebx, %eax  .L2:  addq $8, %rsp  popq %rbx  popq %rbp  ret  .LFE23:  .size f, .-f  .section .rodata.str1.1,"aMS",@progbits,1  .LC0:  .string "%d\n"  .text  .globl main  .type main, @function  main:  .LFB24:  subq $8, %rsp  movl $5, %r8d  movl $4, %ecx  movl $3, %edx  movl $2, %esi  movl $1, %edi  call f  movl %eax, %edx  movl $.LC0, %esi  movl $1, %edi  movl $0, %eax  call \_\_printf\_chk  movl $0, %eax  addq $8, %rsp  ret  .LFE24:  .size main, .-main  .ident "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"  .section .note.GNU-stack,"",@progbits |

1) 밑줄 그은 함수 호출 수행 시의 스택의 상태와 레지스터 값을 나타내시오. (스택은 main 함수로의 리턴 주소부터 스택 탑까지의 내용을 표시한다. 레지스터 값 중 주소는 화살표로 표시하고, 리턴 주소는 <ret addr>로 표시)

**A21) 스택의 상태  
코드 흐름** : main에서 rsp를 8만큼 빼서 스택공간 1칸을 확보했고, 그 이후 call로 인해 return address가 스택에 저장되었다. 그 후, pushq 명령어로 callee saved 레지스터 rbp, rbx를 스택에 저장했고, 이후 또 subq로 rsp를 8만큼 빼서 스택공간 1칸을 확보했다. 이후에는 add 연산을 진행 후, cmpl에서 edi-esi = -1이기 때문에 jg는 무시되었고, call까지 진행된다.

<ret addr>

Saved %rbx

Saved %rbp

UNUSEDd

……

%rsp

2) f 함수에서 Callee Save 레지스터이기 때문에 메모리에 저장한 레지스터는 무엇인가?

**A22-1)** 어셈블리 코드를 보면 f 함수 시작하자마자 pushq %rbp, pushq %rbx를 하는 것을 보아 이는 callee-saved 레지스터이기 때문에 이 후 f 함수 내에서 값이 변하기 때문에 미리 값을 스택에 저장하는 것을 알 수 있다.

f함수에서 재귀 호출 전에 caller save 레지스터들을 메모리에 저장하지 않은 이유 무엇인가?   
**A22-2)** main 함수에서나 f 함수에서나 f 함수를 호출하기 전에 계산이나 저장한 값을 호출 후 복귀했을 때 사용하는 caller save 레지스터가 없기 때문에 굳이 caller save 레지스터들을 메모리에 저장하지 않았다.