Recitation Exercises #2

CSCI 2400 Systems
Fall 2014
~~25~~ 29
September ~~25~~, 2014

1

Below is the assembly code for a C main function as well as the factorial function which it calls.

```
0804841d <factorial>:
804841d:        55                      push   %ebp
804841e:        89 e5                   mov    %esp,%ebp
8048420:        83 ec 18                sub    $0x18,%esp      Subtract stack pointer 24
8048423:        83 7d 08 00             cmpl   $0x0,0x8(%ebp)  compare n:0
8048427:        75 07                   jne    8048430 <factorial+0x13>  jump n!=0
8048429:        b8 01 00 00 00          mov    $0x1,%eax       n=1
804842e:        eb 12                   jmp    8048442 <factorial+0x25>
n!=0 ⇒8048430:  8b 45 08                mov    0x8(%ebp),%eax  eax=n
8048433:        83 e8 01                sub    $0x1,%eax       n-1
8048436:        89 04 24                mov    %eax,(%esp)     esp→,eax=n
8048439:        e8 df ff ff ff          call   804841d <factorial>  call factorial
804843e:        0f af 45 08             imul   0x8(%ebp),%eax  n.return
→8048442:       c9                      leave
8048443:        c3                      ret

08048444 <main>:
8048444:        55                      push   %ebp
8048445:        89 e5                   mov    %esp,%ebp
8048447:        83 e4 f0                and    $0xfffffff0,%esp  aligns stack to 16 bytes
804844a:        83 ec 20                sub    $0x20,%esp      subtract stack pointer 32
804844d:        c7 04 24 07 00 00 00    movl   $0x7,(%esp)     esp=7
8048454:        e8 c4 ff ff ff          call   804841d <factorial>  call factorial
8048459:        89 44 24 1c             mov    %eax,0x1c(%esp)
804845d:        8b 44 24 1c             mov    0x1c(%esp),%eax
8048461:        89 44 24 04             mov    %eax,0x4(%esp)
8048465:        c7 04 24 10 85 04 08    movl   $0x8048510,(%esp)
804846c:        e8 7f fe ff ff          call   80482f0 <printf@plt>
8048471:        b8 00 00 00 00          mov    $0x0,%eax       eax=0
8048476:        c9                      leave
8048477:        c3                      ret
```

ebp→ n
0 –
-4 –
-8 –
-12 –
-16 –
-20 –
-24 – 0x8(ebp) [n]
-28 –
ebp→ 0 -32 – ←esp, [-4]
4 –
8 –
12 –
16 –
20 –
-24 – ←esp [→eax=n]

1.1

How large is the stack frame created for each call to *factorial*? 24 bytes

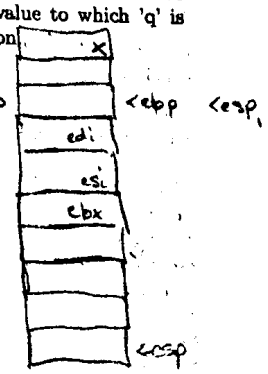
1.2

What is the total "distance" covered by the stack pointer from the first call to *factorial* until the end of the last recursive call?

2

N - 24 bytes

Look at the assembly code below and fill out the blanks in the corresponding C code. You need to indicate the data types (short, int, char, etc as well as unsigned if appropriate) for 'a', 'x', 'y', 'q' and 'z', the value to which 'q' is initialized, what is returned and the equivalent computation.




```
function1:
pushl %ebp
movl %esp, %ebp
pushl %edi
pushl %esi
pushl %ebx
subl $16, %esp
movl $43, %esi
movl $0, %ecx  Q
movsbl 8(%ebp),%ebx  sign extended to dword
                     byte

.L2:
movzbl %cl, %eax  zero extended to dword
subw -24(%ebp,%eax,4), %si  byte
leal (%eax,%ebx), %edx
movl %edx, -24(%ebp,%eax,4)
addl $1, %ecx  q+=1
movzbl %cl, %eax  zero extended to dword
cmpl 12(%ebp), %eax  right
jl .L2
movswl %si,%eax  Sign extended word to dword
addl %ebx, %eax
addl $16, %esp
popl %ebx
popl %esi
popl %edi
popl %ebp
ret
```

C Code:

```
function1(char a, signed int x)
{

signed int z[3];
signed short y = 43;
unsgined int q = 0;

do {

a = a - z[8];
z[q] = y + a;
q = 8 + 1;
} while ( a < x ); //condition for the "do loop"
return a + y;
}
```