- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Homework 18

NOTE: You are not required to hand in the following exercises, but you are strongly encouraged to complete them to strengthen your understanding of the concepts covered in class.

1. A C function `fun` has the following code body:

```
*p = d;
return x-c;
```

The IA32 code implementing this body is as follows:

```
1 movsbl 12(%ebp),%edx
2 movl   16(%ebp), %eax
3 movl   %edx, (%eax)
4 movswl 8(%ebp),%eax
5 movl   20(%ebp), %edx
6 subl   %eax, %edx
7 movl   %edx, %eax
```

Write a prototype for function fun, showing the types and ordering of the arguments `p`, `d`, `x`, and `c`. Here, `movsbl` moves a sign-extended byte (8-bit) to double word (32-bit), and `movswl` moves a sign-extended word (16-bit) to double word (32-bit).

2. Given the C function

```
int proc(void)
{
   int x,y;
   scanf("%x %x", &y, &x);
   return x-y;
}
```

`gcc` generates the following assembly code:

```
1  proc:
2     pushl %ebp
3     movl  %esp, %ebp
4     subl  $40, %esp
5     leal  -4(%ebp), %eax
6     movl  %eax, 8(%esp)
7     leal  -8(%ebp), %eax
8     movl  %eax, 4(%esp)
9     movl  $.LC0, (%esp)     # Pointer to string "%x %x"
10    call  scanf             # Draw a diagram of stack frame at this point
```

```
11    movl -4(%ebp), %eax
12    subl -8(%ebp), %eax
13    leave
14    ret
```

Assume that procedure `proc` starts executing with the following register values:

Register	Value
%esp	0x800040
%ebp	0x800060

Suppose `proc` calls `scanf` (line 10), and that it reads values `0x46` and `0x53` from the standard input. Assume that the string "`%x %x`" is stored at memory location `0x300070`.

A. What value does `%ebp` get set to on line 3?
B. What value does `%esp` get set to on line 4?
C. At what addresses are local variables `x` and `y` stored?
D. Draw a diagram of the stack frame for `proc` right after `scanf` returns. Include as much information as you can about the addresses and the contents of the stack frame elements.
E. Indicate the regions of the stack frame that are not used by `proc`.

3. The following code sequence occurs right near the beginning of the assembly code generated by `gcc` for a C procedure:

```
1  subl $12, %esp
2  movl %ebx, (%esp)
3  movl %esi, 4(%esp)
4  movl %edi, 8(%esp)
5  movl 8(%ebp), %ebx
6  movl 12(%ebp), %edi
7  movl (%ebx), %esi
8  movl (%edi), %eax
9  movl 16(%ebp), %edx
10 movl (%edx), %ecx
```

We see that just three registers (`%ebx`, `%esi`, and `%edi`) are saved on the stack (lines 2–4). The program modifies these and three other registers (`%eax`, `%ecx`, and `%edx`). At the end of the procedure, the values of registers `%edi`, `%esi`, and `%ebx` are restored (not shown), while the other three are left in their modified states. Explain this apparent inconsistency in the saving and restoring of register states.