Machine-Level Representation of Programs #3 **4190.308 Computer Architecture**

Due Date: Thursday, October 2, 2014, 23:59 **Solution**

Submission: in paper form.
There will be a drop off box in class and in front of the CSAP Lab in building 301, room 419.

Question 1
switch statements

For the following C function `switcher` GCC generates assembly code and a jump table as shown below:

```
int switcher(int a, int b, int c)
{
  int answer;
  switch(a) {

    case 4: /* Case A */

      c = b & 15;

      /* Fall through */

    case 7: /* Case B */

      answer = c - 138;
      break;

    case 5: /* Case C */

    case 0: /* Case D */

      answer = (c + 3) * b;
      break;

    case 2: /* Case E */

      answer = 4;
      break;
    default:

      answer = b * a;
  }
  return answer;
}
```

```
a at %ebp+8, b at %ebp+12, c at %ebp+16

switcher:
  movl  8(%ebp), %edx
  movl  12(%ebp), %ecx
  movl  16(%ebp), %eax
  cmpl  $7, %edx
  jbe .L10
.L2:
  movl  %ecx, %eax
  imull %edx, %eax
  ret
.L10:
  jmp *.L7(,%edx,4)
.L5:
  movl  %ecx, %eax
  andl  $15, %eax
.L6:
  subl  $138, %eax
  ret
.L8:
  movl  $4, %eax
  ret
.L3:
  addl  $3, %eax
  imull %ecx, %eax
  ret
.L7:
  .long .L3
  .long .L2
  .long .L8
  .long .L2
  .long .L5
  .long .L3
  .long .L2
  .long .L6
```

Fill in the missing parts of the C code. Except for the ordering of case labels `C` and `D`, there is only one way to fit the different cases into the template.

Question 1

Stack frame structure

A C function `fun` has the following function body and the IA32 code implementing this body is as follows:

```c
int fun(char c, unsigned short d, int *p, int x)
{
  *p = (int) d;
  return x-c;
}
```

```
fun:
    pushl   %ebp
    movl    %esp, %ebp
    movzwl  12(%ebp), %edx
    movl    16(%ebp), %eax
    movl    %edx, (%eax)
    movsbl  8(%ebp), %edx
    movl    20(%ebp), %eax
    subl    %edx, %eax
    popl    %ebp
    ret
```

Write a prototype for function `fun`, showing the types and ordering of the arguments `p`, `d`, `c`, and `x`.

Question 2

Recursive Procedures

For the C function *rfun* with the following general structure GCC generates the assembly code as shown below:

```
int rfun(unsigned x, unsigned y)
{

  if (y == x) return 1;

  if (y == 1) return x;

  return rfun(x-1, y-1) + rfun(x-1, y);
}
```

```
rfun:
  pushl %ebp
  movl  %esp, %ebp
  subl  $40, %esp
  movl  %ebx, -12(%ebp)
  movl  %esi, -8(%ebp)
  movl  %edi, -4(%ebp)
  movl  8(%ebp), %esi
  movl  12(%ebp), %ebx
  movl  $1, %eax
  cmpl  %esi, %ebx
  je  .L2
  movl  %esi, %eax
  cmpl  $1, %ebx
  je  .L2
  subl  $1, %esi
  leal  -1(%ebx), %eax
  movl  %eax, 4(%esp)
  movl  %esi, (%esp)
  call  rfun
  movl  %eax, %edi
  movl  %ebx, 4(%esp)
  movl  %esi, (%esp)
  call  rfun
  addl  %edi, %eax
.L2:
  movl  -12(%ebp), %ebx
  movl  -8(%ebp), %esi
  movl  -4(%ebp), %edi
  movl  %ebp, %esp
  popl  %ebp
  ret
```

a) Fill in the missing expressions in the C code shown above.

b) (Assume x > y > 0) Describe what function this code computes.

rfun computes the number of y-combinations from a set S of x elements,

$$\binom{x}{y} = \binom{x-1}{y-1} + \binom{x-1}{y}$$