6.004 Tutorial Problems
L4 – Procedures and Stacks II

Problem 1.

Integer arrays **season1** and **season2** contain points Ben Bitdiddle had scored at each game over two seasons during his time at MIT Intramural Basketball Team. Please write a RISC-V assembly program which counts the number of games he scored more than 20 points. An equivalent C program is given below. Note that the base addresses for arrays **season1** and **season2** along with their size are passed down to function **greaterthan20**.

```
int main() {
      int season1[7] = {18, 28, 19, 33, 25, 11, 20};
      int season2[7] = {30, 12, 13, 33, 37, 19, 22};
      int result = greaterthan20(season1, season2, 7);
}

int greaterthan20(int a[], int b[], int size) {
      int count = 0;
      for (int i =0; i < size; i++) {
            if (a[i] > 20)
                count++;
            if (b[i] > 20)
                count++;
      }
      return count;
}
```

```
// Beginning of your assembly code
greaterthan20:
      li t0, 0 // t0 <= count
      li t1, 0 // t1 <= index
      li t2, 20
loop :
      ble a2, t1, endloop
      sll t1, t1, 2 // 4*index
checka:
      add t3, a0, t1
      lw t4, 0(t3)
      ble t4, t2, checkb // if a[i] =< 20, then check b[i]
      addi, t0, t0, 1 // increment counter
checkb:
      add t3, a1, t1
      lw t4, 0(t3)
      ble t4, t2, endcompare // if b[i] =< 20, then go to endcompare
      addi t0, t0, 1 // increment counter
endcompare:
      add t1, t1, 1 // increment index
      j loop
endloop:
      mv a0, t0
      ret
```

Problem 2.

The following C program computes the log base 2 of its argument. The assembly code for the procedure is shown on the right, along with a stack trace showing the execution of ilog2(10). The execution has been halted just as it's about to execute the instruction labeled "rtn:" The SP label on the stack shows where the SP is pointing to when execution halted.

```
/* compute log base 2 of arg */
int ilog2(unsigned x) {
    unsigned y;
    if (x == 0) return 0;
    else {
        /* shift x right by 1 bit */
        y = x >> 1;
        return ilog2(y) + 1;
   }
}
```

```
ilog2: beqz a0, rtn
       addi sp, sp, -8
       sw s0, 4(sp)
       sw ra, 0(sp)
       srli s0, a0, 1
       mv a0, s0
       jal ra, ilog2
       addi a0, a0, 1
       lw ra, 0(sp)
       lw s0, 4(sp)
       addi sp, sp, 8
rtn:   jr ra
```

(A) Please fill in the values for the two blank locations in the tack trace shown on the right. Please express the values in hex.

Fill in values (in hex!) for 2 blank locations

	0x93
	0x240
	0x1
	0x240
	0x2
SP→	0x240
	0x5
	0x1108
	0x37

(B) What are the values in a0, s0, sp, and pc at the time execution was halted? Please express the values in hex or write "CAN'T TELL".

Value in a0: 0x_____ in s0: 0x_____

Value in sp: 0x_____ in pc: 0x_____

a0 = 2, s0 = 2, sp = Can't tell, pc = 0x250

(C) What was the address of the original ilog2(10) function call?

Original ilog2(10) address: 0x_____

0x1104

Problem 3.

You are given an incomplete listing of a C program (shown below) and its translation to RISC-V assembly code (shown on the right):

```
int fn(int x) {
  int lowbit = x & 1;
  int rest = x >> 1;
  if (x == 0) return 0;
  else return ???;
}
```

```
fn: addi sp, sp, -12
    sw s0, 0(sp)
    sw s1, 4(sp)
    sw ra, 8(sp)
    andi s0, a0, 1
    srai s1, a0, 1
yy: beqz a0, rtn
    mv a0, s1
    jal ra, fn
    add a0, a0, s0

rtn: lw s0, 0(sp)
     lw s1, 4(sp)
     lw ra, 8(sp)
     addi sp, sp, 12
     jr ra
```

(A) What is the missing C source corresponding to ??? in the above program?

C source code: _____

fn(rest) + lowbit

The procedure **`fn`** is called from an external procedure and its execution is interrupted just prior to the execution of the instruction tagged '**`yy:`**'. The contents of a region of memory are shown on the left below. If the answer to any of the below problems cannot be deduced from the provided information, write "CAN'T TELL".

	0x1
1D0	???
	0x4C
SP→	0x1
rest	0x11
ra	0x4C
lowbit	0x1
rest	0x23
ra	0x4C
orig s0	0x3
orig s1	0x22
orig ra	0xC4

(B) What was the argument to the most recent call to **`fn`**?

Most recent argument (HEX): x=_____0x11

(C) What is the missing value marked **`???`** for the contents of location 1D0?

Contents of 1D0 (HEX): _____CAN'T TELL

(C) What is the hex address of the instruction tagged **rtn:?**

Address of rtn (HEX): _____0x50

(D) What was the argument to the *first recursive* call to **`fn`**?

First recursive call argument (HEX): x=_____0x23

(E) What is the hex address of the *jal* instruction that called **`fn`** *originally*?

Address of original call (HEX): _____0xC0

(F) What were the contents of s1 at the time of the *original* call?

Original s1 contents (HEX): _____0x22

(G) What value will be returned to the *original* caller if the value of *a0* at the time of the original call was 0x47?

Return value for original call (HEX): _____0x4
counts the number of 1's in original number