CSE3666

Homework 3

Due Date: By the end of Friday, 09/29/2023.

Total points: 100

1. Translate function foo() in the following C code to RISC-V assembly code. Assume function bar() has already been implemented. The constraints/tips are:
   1) Allocate register `s1` to `sum`, and register `s2` to `i`.
   2) There are no load or store instructions in the loop. If we want to preserve values across function calls, place the value in a saved register before the loop. For example, we keep variable `i` in register s2.
   3) Identify the registers that are changed in function foo() but should be preserved. Note that the callee, bar(), may change any temporary and argument registers.
   4) Save registers at the beginning of the function and restore them before the exit.

   Your code should follow the flow of the C code. Write concise comments.

   Clearly mark instructions for saving registers, loop, function calls, restoring register, etc.

```
// prototype of bar
// the first argument is an address of an integer
int  bar(int a[], int i);

int foo(int d[], int n)
{
    int sum = 0;
    for (int i = 0; i < n; i += 1) {
        sum += bar(&d[i], n - i);    // &d[i] means d[i]'s address
    }
    return sum;
}
```

2. Translate function msort() in the following C code to RISC-V assembly code. Assume merge() and copy() are already implemented. The array passed to msort() has at most 256 elements.

   Your code should follow the flow of the C code. Write concise comments. Clearly mark instructions for saving registers, function calls, restoring register, and so on.

   To make the code easier to read, we change sp twice at the beginning of the function: once for saving registers and once for allocating memory for array c.

   The function should have only one exit. There is only one return instruction.

   Another reminder: callees may change any temporary and argument registers.

```
void merge(int c[], int d1[], int n1, int d2[], int n2);
void copy(int d[], int c[], int n);

void  msort(int d[], int n)
{
    int c[256];

    if (n <= 1)
        return;

    int n1 = n / 2;

    msort(d, n1);
    msort(&d[n1], n - n1);    // &d[n1] means the address of d[n1]
    merge(c, d, n1, &d[n1], n - n1);
    copy(d, c, n);
}
```

Always explain your code with concise comments. The code should clearly show how arguments are set in each function calls.

3. Find the machine code for the following instruction. Assume all instructions are labeled sequentially, for example, I1, I2, I3, …, I150.

```
…
I10 :        BGE  x10, x20, I100
I11 :        BEQ  x10, x0, I1
…
I140:        JAL  x0, I100
```

4. Decode the following instructions in machine code. Find the offset in decimal and then the target address in hexadecimal. The hexadecimal number before the colon is the instruction's address.

```
0x0400366C:          0xDB5A04E3

0x04208888:          0xFA9FF0EF
```