计算机组成小测2答案

1. **Choose true or false**
   a. In order to use the saved registers (s0-s11) in a function, we must store their values before using them and restore their values before returning. (T)
   b. jalr is a shorthand expression for a jal that jumps to the specified label and does not store a return address anywhere. (F)
   c. Assuming no compiler or operating system protections, it is possible to have the code jump to data stored at 0(a0) (offset 0 from the value in register a0) and execute instructions from there. (T)
   d. In the multiplication and division extension of RISC-V, the division instruction ensures that an exception will not be thrown even if the divisor is zero. (F)
   e. Each instruction in the RISC-V instruction set has two operating modes, 32-bit and 64-bit, corresponding to the processor's bit width. (F)
2. **Answer the following questions briefly**
   a. Translate between C and RISC-V

C	RISC-V

e.g.
// s0 -> a, s1 -> b
// s2 -> c
int a = 1, b = 2, c;
c = a + b;
	e.g.
	addi s0, x0, 1
	addi s1, x0, 2
	add s2, s0, s1

// s0 -> a, s1-> b
int a = 5, b = 10;
if (a + a == b) a = 0;
else b = a - 1;
	addi s0, x0, 5
	addi s1, x0, 10
	addi t0, s0, s0
	beq t0, s1, if
	addi s1, s0, -1
	jal x0, exit
	if:
	addi s0, x0, 0
	exit:

// a0 -> a, a1 -> b, result -> a0
int f (int a, int b) {
if (a <= 0 || b <= 0) return 0;
else return f(b-2, a-b);
}
	func:
	bge x0, a0, done
	bge x0, a1, done
	addi sp, sp, -4
	sw ra, 0(sp)
	addi t0, a1, -2
	sub a1, a0, a1
	addi a0, t0, 0
	jal ra, func
	lw ra, 0(sp)
	jalr x0, 0(ra)
	done:
	addi a0, x0, 0
	jalr x0, 0(ra)

// s0 -> int * p =intArr
// s1 -> a
 	
*p = 0;
int a = 2;
p[1] = p[a] = a;
	sw x0, 0(s0)
	addi s1, x0, 2
	sw s1, 4(s0)
	slli t0, s1, 2
	addi t0, s0, t0
	sw s1, 0(t0)

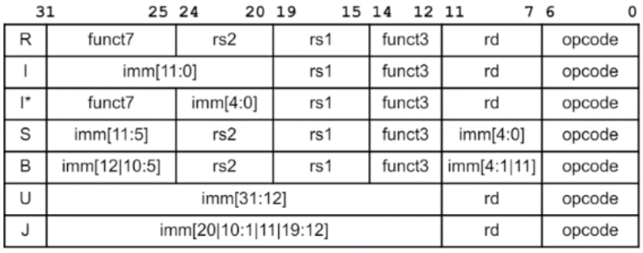
b. Convert the following single-precision floating point numbers from hexadecimal to decimal or from decimal to hexadecimal. (Use IEEE 754)

1) -∞   2) 65.3125   3) -0.3   4) 0x7FABCDEF

1) 0xFF800000   2) 0x4282A000   3) 0xBE99999A   4) NaN

3. Consider the following loop in RISC-V:

```
LOOP:
        blt x6, x0, DONE
        addi x6, x6, -1
        addi x5, x5, 3
        jal x0, LOOP
DONE:
        addi x0, x0, 0
```



	31 25	24 20	19 15	14 12	11 7	6 0
R	funct7	rs2	rs1	funct3	rd	opcode
I	imm[11:0]		rs1	funct3	rd	opcode
I*	funct7	imm[4:0]	rs1	funct3	rd	opcode
S	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
B	imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]				rd	opcode
J	imm[20|10:1|11|19:12]				rd	opcode

a. Assume that the register x6 is initialized to the value 100. What is the final value in register x5 assuming the x5 is initially zero ?

303

b. For the loop written in RISC-V assembly above, assume that the register x6 is initialized to the value N. How many RISC-V instructions are executed ?

4N+6

c. Assume that the PC value to label "LOOP" is 0x04, write the machine binary code of the first and second instruction of this LOOP.

J-Type opcode: 0x6F; I-Type opcode: 0x13, addi-funct3: 0; B-Type opcode: 0x63, blt-funct3: 0x4

e.g. addi, x0, x0, 0 -> 0x00000013

blt x6, x0, DONE --> 0x00034863

addi x6, x6, -1 --> 0xFFF30313