Points [ ] Update and Regrade | Hide Question Details

□ **1. Multiple Choice: 1: Analyze the following block of RISC-V...** Points: 0.5

Question

Analyze the following block of RISC-V assembly code and calculate the value stored in register t1, t2, t3 after all lines are executed. All numbers are decimal.

```
li t1,  200        //t1=200
li t2, 300         //t2=300
add t3, t1, t2     //t3=t1+t2=500
sub t2, t3, t2     //t2=t3-t2=200
```

Answer

✓ a. t1= 200, t2=200,   t3=500

b. t1= 300, t2=200,   t3=500

c. t1= 200, t2=500,   t3=300

d. t1= 200, t2=100,   t3=500

□ **2. Multiple Answer: 2: Which one of the following is a Pseud...** Points: 1

Question

Which one of the following is a Pseudo-Instruction in RISC-V ISA? Select all that apply. For help, look at RISC-V user manual and documents uploaded in the reading section.

Answer

✓ a. nop   no operation, (implemented with "addi x0, x0, 0")

b. sub rd, x0, rs Not a pseudo instruction

✓ c. not rd, rs rd= NOT rs

✓ d. ret Return from function

□ 3. Multiple Answer: 3: Which one of the following instructio...

Points: 0.5

Question Which one of the following instructions or pseudo-Instructions is not valid in RISC-V ISA? Select all that apply. Look at the RISC-V user manual for help.

Answer

✓ addi x1,x2,x3 format not correct, addi should contain an immediate value

li x5, 200 Correct format

bnez x5, label Correct format

✓ lw x1,x2,x3 format not correct, should contain an immediate value. (LW rd, rs1, imm)

□ 4. Multiple Answer: 4: Analyze the following block of code a...

Points: 1

Question

Analyze the following block of code and calculate the number of times the instruction “add t2, t2, t1” will execute? All numbers are decimal.

```
       li t1,5

       li t2,100

       loop:
                                 iteration 1--iteration 2--iteration 3--iteration 4--iteration 5
line 3 add t2, t2, t1      After line 3: t1=5   --t1=4    --t1=3    --t1=2    --t1=1
line 4 addi t1, t1, -1     After line 4: t1=4   --t1=3    --t1=2    --t1=1    --t1=0
line 5 bnez t1, loop       After line 5: branch --branch  --branch  --branch  --go to line 6
line 6 addi sp,sp,-48
```

Answer

✓ a. 5

b. 100

c. 105

d. 6

☐ **5. Multiple Answer: 5: What is the content in memory address...**

Points: 0.5

Question

What is the content in memory address 100 and 104 after executing the following code? All numbers are decimal.

```
li t0,50             //t0=50
addi a0, x0, 100     //a0=0+100          --x0 is always 0
addi t1, t0, 2       //t1=50+2=52
sw t0, 0(a0)         //store to  mem address (0+a0) from 100 from reg t0. t0 contains 50
sw t1, 4(a0)         //store to  mem address (4+a0) from 104 from  reg t1. t1 contains 52
```

Answer

✓ a. Value in address 100 is 50.

Value in address 104 is 52

b. Value in address 100 is 100.

Value in address 104 is 102

c. Value in address 100 is 50.

Value in address 104 is 100

d. Value in address 100 is 52.

Value in address 104 is 56

□ 6. Multiple Answer: 6: In the base RISCV 32 bit integer ISA,...

Points: 0.5

Question

In the base RISCV 32 bit integer ISA, there are six instruction formats (look at the RISC-V Green card uploaded as reading materials on Blackboard).

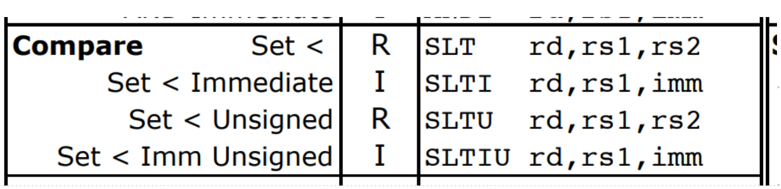
Which one of the following instructions follows the R-format?

Answer

a. Load Byte (LB)

from risc v cheat sheet in reading material

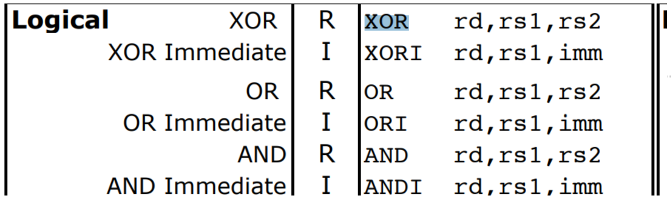
b. Add Immediate (ADDI)



Compare	Set <	R	SLT rd,rs1,rs2
	Set < Immediate	I	SLTI rd,rs1,imm
	Set < Unsigned	R	SLTU rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU rd,rs1,imm

✓ c. Set less than (SLT)

✓ d. XOR



Logical	XOR	R	XOR rd,rs1,rs2
	XOR Immediate	I	XORI rd,rs1,imm
	OR	R	OR rd,rs1,rs2
	OR Immediate	I	ORI rd,rs1,imm
	AND	R	AND rd,rs1,rs2
	AND Immediate	I	ANDI rd,rs1,imm

e. Jump and link (JAL)

□ 7. Multiple Choice: 7: In the base RISCV 32 bit integer ISA,...

Points: 0.5

Question

In the base RISCV 32 bit integer ISA, there are six instruction formats (look at the RISCV Green card uploaded as reading materials on Blackboard).

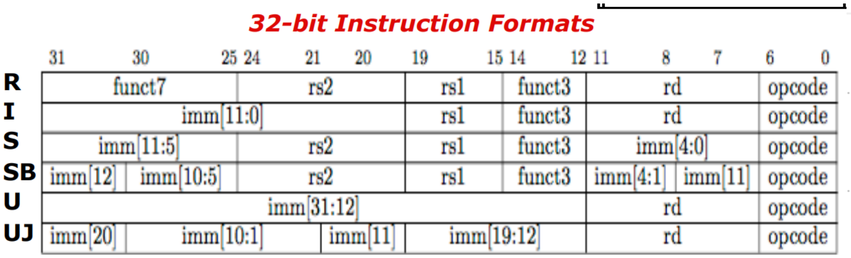
In the I-format, how many bits are reserved for encoding the 'immediate' value?

Answer

✓ a. 12

b. 10

c. 1



32-bit Instruction Formats

	31	30 25	24 21	20	19 15	14 12	11 8	7	6 0
R	funct7		rs2		rs1	funct3	rd		opcode
I	imm[11:0]				rs1	funct3	rd		opcode
S	imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode
SB	imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode
U	imm[31:12]						rd		opcode
UJ	imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode

from risc v cheat sheet in reading material

d. 20

□ 8. **Multiple Answer: 8: Which one of the following opcodes is...**

Points: 0.5

Question Which one of the following opcodes is not a valid control flow instruction or pseudo-instruction in the RISCV ISA? Select all that apply.

Refer to the RISC-V user manual (https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf).

Answer

a. ble

b. beqz

✓ c. bgr

d. blt

e. jalr

✓ f. jrl

□ 9. **Multiple Answer: 9: Which ones are ‘caller saved’ registe...**

Points: 1

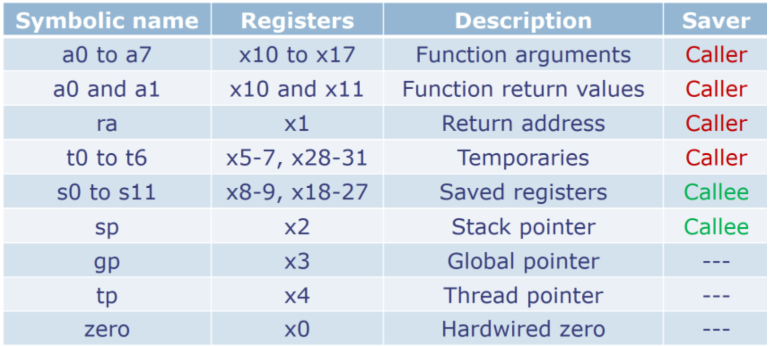
Question Which ones are ‘caller saved’ register in a procedure call? Select all that apply.

Answer

a. Saved register, s1

✓ b. Function argument register, a0

✓ c. Return address register, ra



Symbolic name	Registers	Description	Saver
a0 to a7	x10 to x17	Function arguments	Caller
a0 and a1	x10 and x11	Function return values	Caller
ra	x1	Return address	Caller
t0 to t6	x5-7, x28-31	Temporaries	Caller
s0 to s11	x8-9, x18-27	Saved registers	Callee
sp	x2	Stack pointer	Callee
gp	x3	Global pointer	---
tp	x4	Thread pointer	---
zero	x0	Hardwired zero	---

lecture 6

d. Stack pointer register, sp

□ 10. Multiple Choice: 10: Let us assume that we have an instruc...

Points: 0.5

Question

Let us assume that we have an instruction “jal ra, func” in memory location 104. Register ‘ra’ holds the return address and ‘func’ is the label of a procedure.

What is the address stored in ‘ra’ after executing this instruction? All numbers are in decimal.

Answer

✓ a. 108

b. 100

c. 104

d. 400

during procedure call, before we make the jump we must store the return address to reg ra. The return address is the next address from which we are calling the procedure. Next address for 32 bit RISC-V is current add.+ 4.

In the example to the right, function foo() has a procedure bar(). When we jump to bar() we have store the return address because after bar() is complete we want to continue from z=2 in address 12. So ret add is 12

```
def foo ():
    x = 1
    bar ()
    z = 2

def bar ():
    y = 7
```

Address	32 bit data
0	foo()
4	X=1
8	Jump to bar()
12	Z=2
16	
24	bar()
28	Y=7

□ 11. Multiple Choice: 11: Following the RISC-V calling conven...

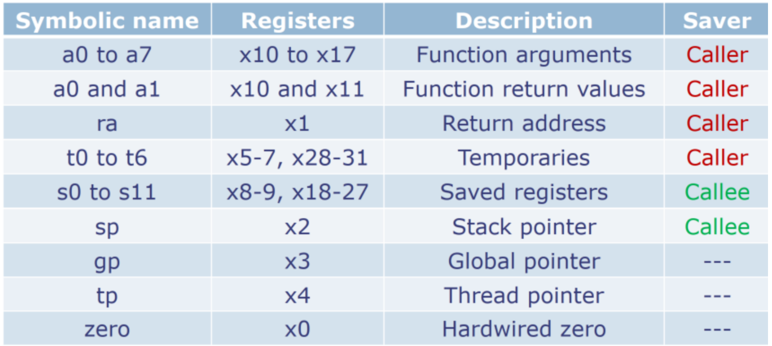
Points: 1

Question

Following the RISC-V calling convention (slide 21 of lecture 6), the return value a+b of function sum() in the following code should be stored using which register?

```
int sum(int a, int b){
    return a+b;
}
void main() {
     int x=10;
     int y=20;
     int z= sum(x,y);
     z++;
     x=x+2;
}
```

the function return values are stored using reg a0



Symbolic name	Registers	Description	Saver
a0 to a7	x10 to x17	Function arguments	Caller
a0 and a1	x10 and x11	Function return values	Caller
ra	x1	Return address	Caller
t0 to t6	x5-7, x28-31	Temporaries	Caller
s0 to s11	x8-9, x18-27	Saved registers	Callee
sp	x2	Stack pointer	Callee
gp	x3	Global pointer	---
tp	x4	Thread pointer	---
zero	x0	Hardwired zero	---

Answer

a. ra

b. sp

c. t0

✓ d. ao

□ 12. Multiple Choice: 12: Consider the following C code. Let us...

Points: 1

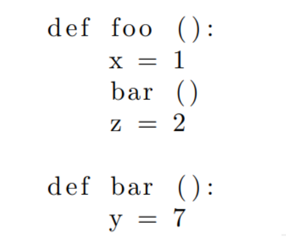
Question

Consider the following C code. Let us assume that the function sum() is currently executing in the processor. Which main memory address is currently available in the return address register ra?

```
int sum(int a, int b){
        return a+b;
}
void main() {
       int x=10;
      int y=20;
       int z= sum(x,y);
       z++;
       x=x+2;
}
```

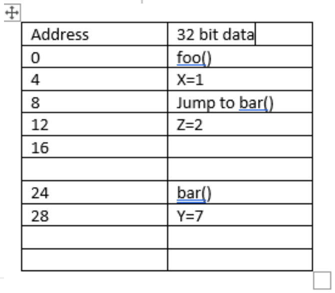
during procedure call, before we make the jump we must store the return address to reg ra. The return address is the next address from which we are calling the procedure. Next address for 32 bit RISC-V is current add.+ 4.

In the example to the right, function foo() has a procedure bar(). When we jump to bar() we have store the return address because after bar() is complete we want to continue from z=2 in address 12. So ret add is 12



```
def foo ():
    x = 1
    bar ()
    z = 2

def bar ():
    y = 7
```



Address	32 bit data
0	foo()
4	X=1
8	Jump to bar()
12	Z=2
16	
24	bar()
28	Y=7

Answer

✓ a. Memory address that stores line “z++;”

b. Memory address that stores line “int x=10;”

c. Memory address that stores line “return a+b;”

d. Memory address that stores line “int z= sum(x,y);”

☐ **13. Multiple Choice: 13: What operation is always needed befor...** Points: 0.5

Question What operation is always needed before storing (push operation) something to the stack:

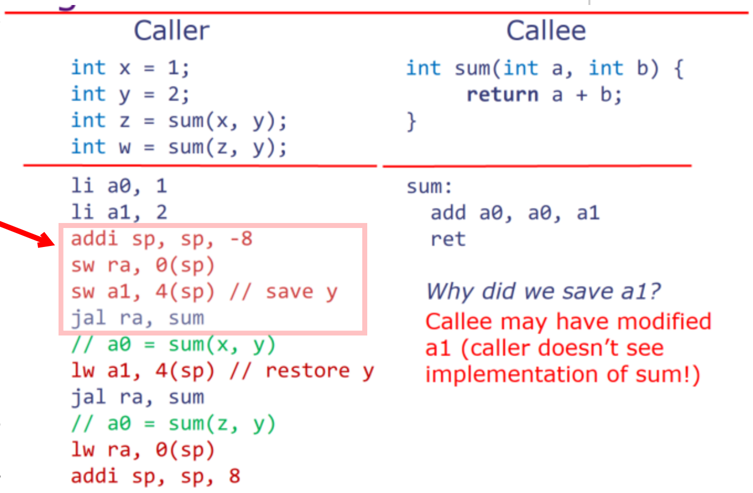
Answer

a. Clearing the existing data on the stack

✓ b. Decreasing the stack pointer value

c. Increasing the stack pointer value

d. Storing the return address to the stack pointer register



Caller

```
int x = 1;
int y = 2;
int z = sum(x, y);
int w = sum(z, y);
```

```
li a0, 1
li a1, 2
addi sp, sp, -8
sw ra, 0(sp)
sw a1, 4(sp) // save y
jal ra, sum
// a0 = sum(x, y)
lw a1, 4(sp) // restore y
jal ra, sum
// a0 = sum(z, y)
lw ra, 0(sp)
addi sp, sp, 8
```

Callee

```
int sum(int a, int b) {
    return a + b;
}
```

```
sum:
  add a0, a0, a1
  ret
```

Why did we save a1?

Callee may have modified a1 (caller doesn't see implementation of sum!)

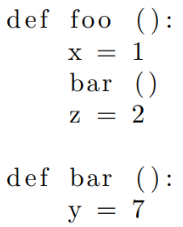
☐ **14. Multiple Choice: 15: State true or false: For the followin...** Points: 1

Question

State true or false: For the following C code, since function sum() contains no other function call inside its code, it is not necessary to store the return address to the stack before executing sum().

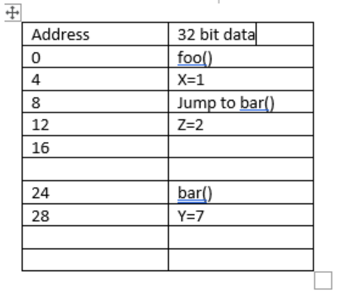
```
int sum(int a, int b){
          return a+b;
}
void main() {
          int x=10;
         int y=20;
          int z= sum(x,y);
          z++;
          x=x+2;
}
```

For the same example, we know that we are storing the return address to register ra.
But what if function bar() has another function call inside, say bar2().
If we jump to bar2() and store a new return address to reg ra,
our return address to go back to foo() will get deleted.
So before we make a jump to bar2(), we store the current return address register's value to stack. After that we can use reg ra.



```
def foo ():
    x = 1
    bar ()
    z = 2

def bar ():
    y = 7
```



Address	32 bit data
0	foo()
4	X=1
8	Jump to bar()
12	Z=2
16	
24	bar()
28	Y=7

Answer

✓ a. True

b. False