

Solution of Discussion04

DrinkLessMilkTea

2025 年 3 月 10 日

1 Pre-Check

1.1

False, a 寄存器也有可能会改变, 只有 s 寄存器不会

1.2

False, 不一定, 这条指令是将 a0 后 4 个字节为首地址的字加载到 s0 中, 如果 x 数组的元素不是 4 个字节, 则不是 x[1] 被加载到 s0

1.3

True

1.4

True

1.5

False, **jalr** 是跳转到某个地址并将返回地址存入某个寄存器, 不是直接忽略返回地址, 除非将寄存器设置为 x0 才会忽略返回地址

1.6

False, **j label** 是指直接跳转到某个 label, 忽略返回地址; 而 **jal label** 是指跳转到某个 label, 同时将返回地址保存到 ra 寄存器

2 RISC-V: A Rundown

2.1

```

addi s0 x0 5    // int x = 5
sw s0 0(s1)     // y[0] = x0
mul t0 s0 s0    // int temp = x * x
sw t0 4(s1)     // y[1] = temp

```

3 Registers

3.1

```

add x8 x0 x11
or s2 ra t5

```

4 Basic Instructions

4.1

(a)

```
t0 = arr[3] = 4
```

(b)

```
arr[4] = t0 = 4
```

(b)

```

t1 = t0 * 2 = 8
t2 = s0 + t1 = &arr[2]
t3 = arr[2] = 3
t3 = t3 + 1 = 4
arr[2] = t3 = 4

```

(c)

```

t0 = arr[0] = 1
t0 = t0 ^ -1 = -2
t0 = t0 + 1 = -1

```

5 C to RISC-V

5.1

5.1.1

```
li s0 4
li s1 5
li s2 6
li s3 0
add s3 s0 s1
add s3 s3 s2
addi s3 s3 10
```

5.1.2

```
sw x0 0(s0)
li s1 2
sw s1 4(s0)
slli s2 s1 2
add s0 s0 s2
sw s1 0(s0)
```

5.1.3

```
li s0 5
li s1 10
add t0 s0 s0
bne t0 s1 ELSE
li s1 0
j NEXT
ELSE:
sub s1 s0 1
NEXT:
```

5.1.4

```
for(int i = 0, j = 1; i < 30; i++) {
    j <<= 1;
}
```

5.1.5

```

    li  s1  0
loop:
    beq  s0  x0  exit
    add  s1  s1  s0
    addi s0  s0  -1
    j    loop
exit:

```

6 RISC-V with Arrays and Lists**6.1**

```
arr[2] = arr[0] + arr[1];
```

6.2

遍历链表, 将每个元素的值自增 1

6.3

遍历数组, 对每个元素取相反数

7 RISC-V Calling Conventions**7.1**

通过设定 `a` 寄存器的值来传递参数, `a0`, `a1`, ... 分别代表第 1 个, 第 2 个... 参数

7.2

函数的返回值保存在寄存器 `a0`

7.3

`sp` 寄存器是栈顶指针, 在函数中, 栈顶指针用来标记当前栈顶的位置, 通过移动栈顶指针可以增加或减少栈空间, 可以用于保存局部变量, 返回地址等

7.4

返回地址, 以及所有后续需要用到的存在 a 和 t 寄存器中的值

7.5

所有后续需要用到的存在 s 寄存器中的值, 以及 sp, gp, tp 寄存器

7.6

s 寄存器和 sp, gp, tp 寄存器保持不变, a 和 t 寄存器还有 ra 寄存器可能改变

8 Writing RISC-V Functions

8.1

参数 n 保存在寄存器 a0, **square** 函数的参数和返回值也保存在 a0, 返回值也存在 a0

8.2

```
# Prologue
addi sp sp -16
sw ra 0(sp)
sw s0 4(sp)
sw s1 8(sp)
sw s2 12(sp)

# start
mv s0 a0
bge x0 s0 exit
j next

exit:
addi sp sp
li a0 0
ret

next:
li s1 1
li s2 0

loop:
blt s0 s1 end
```

```
    mv a0 s1
    jal square
    add s2 s2 a0
    addi s1 s1 1
    j loop
end:
    mv a0 s2
    # Epilogue
    addi sp sp 16
    lw ra 0(sp)
    lw s0 4(sp)
    lw s1 8(sp)
    lw s2 12(sp)
    ret
```

9 More Translating between C and RISC-V

9.1

```
int function(int x, int y) {
    for(int result = 1; y != 0; y --) {
        result *= x;
    }
    return result;
}
```