CMPT 295 RISC-V Basics

1 RISC-V with Arrays and Lists

Comment what each code block does. Each block runs in isolation. Assume that there is an array, `int arr[6] = {3, 1, 4, 1, 5, 9}`, which starts at memory address `0xBFFFFF00`, and a linked list struct (as defined below), `struct ll* lst`, whose first element is located at address `0xABCD0000`. Let `s0` contain `arr`'s address `0xBFFFFF00`, and let `s1` contain `lst`'s address `0xABCD0000`. You may assume integers and pointers are 4 bytes and that structs are tightly packed. Assume that `lst`'s last node's `next` is a NULL pointer to memory address `0x00000000`.

```
struct ll {
    int val;
    struct ll* next;
}
```

1.1
```
lw  t0, 0(s0) // t0 = arr[0]
lw  t1, 8(s0) // t1 = arr[1]
add t2, t0, t1 // add numbers
sw  t2, 4(s0) // arr[2] = t2
```

Sets `arr[1]` to `arr[0]` + `arr[2]`

1.2
```
loop: beq  s1, x0, end
      lw   t0, 0(s1)
      addi t0, t0, 1
      sw   t0, 0(s1)
      lw   s1, 4(s1)
      jal  x0, loop
 end:
```

Increments all values in the linked list by 1.

1.3
```
       add  t0, x0, x0
loop:  slti t1, t0, 6
       beq  t1, x0, end
       slli t2, t0, 2
       add  t3, s0, t2
       lw   t4, 0(t3)
       sub  t4, x0, t4
       sw   t4, 0(t3)
       addi t0, t0, 1
       jal  x0, loop
 end:
```

Negates all elements in `arr`

2 RISC-V Calling Conventions

2.1 How do we pass arguments into functions?

Use the 8 arguments registers `a0` - `a7`

2.2 How are values returned by functions?

Use `a0` and `a1` as the return value registers as well

2.3 What is `sp` and how should it be used in the context of RISC-V functions?

`sp` stands for stack pointer. We subtract from `sp` to create more space and add to free space. The stack is mainly used to save (and later restore) the value of registers that may be overwritten.

2.4 Which values need to saved by the caller, before jumping to a function using `jal`?

Registers `a0` - `a7`, `t0` - `t6`, and `ra`

2.5 Which values need to be restored by the callee, before returning from a function?

Registers `sp`, `gp` (global pointer), `tp` (thread pointer), and `s0` - `s11`. Important to note that we don't really touch `gp` and `tp`

3 More Translating between C and RISC-V

3.1 Translate between the RISC-V code to C. What is this RISC-V function computing? Assume no stack or memory-related issues, and assume no negative inputs.

C	RISC-V
`// a0 -> x, a1 -> y,`	`Func: addi t0 x0 1`
`// t0 -> result`	`Loop: beq a1 x0 Done`
`// Function computes pow(x,y)`	`      mul t0 t0 a0`
`// Direct translation:`	`      addi a1 a1 -1`
`int power(int x, int y) {`	`      jal x0 Loop`
`  int result = 1;`	`Done: add a0 t0 x0`
`  while (y != 0) {`	`      jr ra`
`    result *= x;`	
`    y--;`	
`  }`	
`  return result;`	
`}`	