CS61C Fall 2026 — RISC-V Discussion 4

1 RISC-V Instructions

1.1 Assume we have an array in memory that contains `int *arr = {1,2,3,4,5,6,0}`. Let register `s0` hold the address of the element at index 0 in `arr`. You may assume integers are four bytes and our values are word-aligned. What do the following snippets of RISC-V code do? Assume that all the instructions are run one after the other in the same context.

(a)
```
lw t0, 12(s0)
```

(b)
```
sw t0 16(s0)
```

(c)
```
slli t1, t0, 2
add t2, s0, t1
lw t3, 0(t2)
addi t3, t3, 1
sw t3, 0(t2)
```

(d)
```
lw t0, 0(s0)
xori t0, t0, 0xFFF
addi t0, t0, 1
```

2 Lost in Translation

2.1 Translate the code verbatim between C and RISC-V. The comments above the code indicate which registers to store the variables.

C	RISC-V

`// s0 -> a`
`// s1 -> b`
`// s2 -> c`
`// s3 -> z`
`int a = 4, b = 5, c = 6;`
`int z = a + b + c + 10;`	

`// int *p = intArr;`
`// s0 -> p;`
`// s1 -> a;`
`*p = 0;`
`int a = 2;`
`p[1] = p[a] = a;`	

`// s0 -> a,`
`// s1 -> b`
`int a = 5;`
`int b = 10;`
`if (a + a == b) {`
`  a = 0;`
`} else {`
`  b = a - 1;`
`}`	

`// Compute s1 = 2^30`
`int s0 = 0;`
`int s1 = 1;`
`for (; s0 != 30; s0 += 1) {`
`  s1 *= 2;`
`}`	

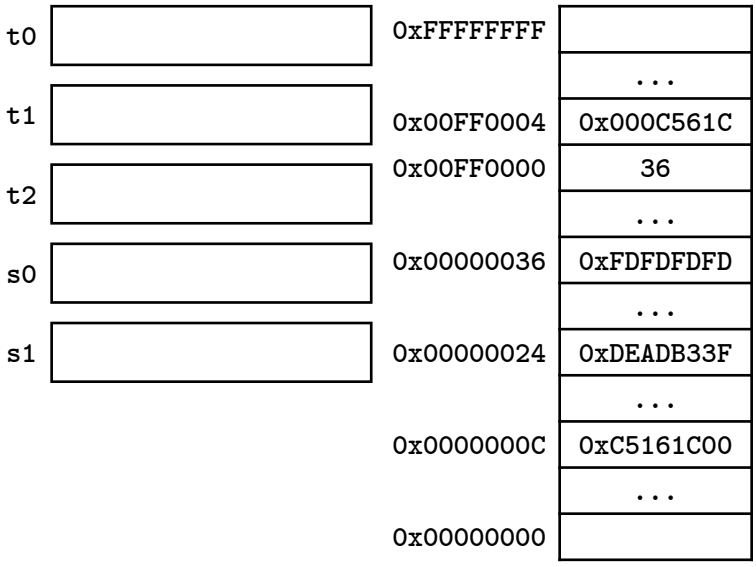
`// s0 -> n`
`// s1 -> sum`
`for (int sum = 0; n > 0; n--) {`
`  sum += n;`
`}`	

3 RISC-V Memory Access

For Q3.1 – Q3.2, use the instructions and memory to figure out what the code does. Recall that RISC-V is little-endian and byte addressable. For any unknown instructions, use the CS 61C reference card!

3.1 Fill in the registers with the values they contain after the code finishes executing.

```
li t0 0x00FF0000
lw t1 0(t0)
addi t0 t0 4
lh t2 2(t0)
lw s0 0(t1)
lb s1 3(t2)
```



3.2 Fill in the memory diagram and `t3` register with the values contained in them after the code finishes executing. The values in the `t0`, `t1`, and `t2` registers at the start of program execution have been provided to you. Assume that all memory starts out initialized to zeros.

```
sw t0 0(t1)
addi t0 t0 4
sh t1 2(t0)
sh t2 0(t0)
lw t3 0(t1)
sb t1 1(t3)
sb t2 3(t3)
```