CS61C Spring 2026

Precheck: RISC-V

Discussion 4

1 RISC-V Instructions

RISC-V is an assembly language composed of simple instructions that each perform a single task such as addition of two numbers or storing data to memory. Below is a comparison between RISC-V code and its equivalent C code:

```
// x in s0, &y in s1
addi s0, x0, 5          int x = 5;
sw s0, 0(s1)            y[0] = x;
mul t0, s0, s0
sw t0, 4(s1)            y[1] = x * x;
```

For your reference, here are some of the basic instructions for arithmetic/bitwise operations and memory access, which can also be found on the 61C Reference Card.

The below are abbreviations that will be used in the table:

- `rs1`: Argument register 1
- `rs2`: Argument register 2
- `rd`: Destination register
- `imm`: Immediate value (integer literal constant)
- `R[register]`: Value contained in `register`
- `inst`: One of the instructions in the table

Register-to-register operations (R-type): `inst rd rs1 rs2`	
`add`	Adds `R[rs1]` and `R[rs2]` and stores the result in `rd`
`xor`	Exclusive ORs `R[rs1]` and `R[rs2]` and stores the result in `rd`
`mul`	Multiplies `R[rs1]` by `R[rs2]` and stores the result in `rd`
`sll`	Logical left shifts `R[rs1]` by `R[rs2]` and stores the result in `rd`
`srl`	Logical right shifts `R[rs1]` by `R[rs2]` and stores the result in `rd`
`sra`	Arithmetic right shifts `R[rs1]` by `R[rs2]` and stores the result in `rd`
`slt(u)`	If `R[rs1] < R[rs2]`, puts 1 in `rd`, else puts 0 (`u` compares unsigned)

Memory operations	
`sw rs2 imm(rs1)`	Stores `R[rs2]` *to* the address `R[rs1] + imm` in memory
`lw rd imm(rs1)`	Loads address `R[rs1] + imm` *from* memory into `rs2`

Branch operations (B-type): `inst rs1 rs2 label`	
`bne`	If `rs1 != rs2`, jump to `label`

2 *Precheck: RISC-V*

Branch operations (B-type): `inst rs1 rs2 label`	
`beq`	If `rs1 == rs2`, jump to `label`

Jump operations (J-type): `inst rd label`	
`jal`	Stores the next instruction's address into `rd` and jumps to `label`

A RISC-V "immediate" is any numeric constant. For example, `addi t0, t0, 20`, `sw a4, -8(sp)`, and `lw a1, 0x44(t2)` have immediates `20`, `-8`, and `0x44` respectively. Note that there is a limit to the size (number of bits) of an immediate in any given instruction (depends on what type of instruction, more on this soon!).

You may also see that there is an "i" at the end of certain instructions, such as `addi`, `slli`, etc. This means that `rs2` becomes an "immediate" or an integer instead of a register. There are immediates in instructions which use an offset such as `sw` and `lw`. When coding in RISC-V, always use the 61C reference card for the details of each instruction (the reference card is your friend)!