

1 RISC-y Conversions

- 1.1 Convert the different RISC-V commands into their hex form or convert the hex form into RISC-V. The instructions are in order of when they would be executed.

(a) `0x005004B3`

`add s1, x0, t0`

(b) `lw t5, 17(t6)`

`0x011FAF03`

(c) `sll s9, x9, t0`

`0x005494B3`

(d) `0x03CE2283`

`lw t0, 60(t3)`

(e) `jalr a0, x11, 8`

`0x00858567`

(f) `ori t1, t2, 5`

`0x0053E313`

(g) `lui a7, 0xCF61C`

`0xCF61C8B7`

2 Instruction Format Design

Prof. Wawrzynek decides to design a new ISA for his ternary neural network accelerator. He only needs to perform 7 different operations with his ISA: xor, add, lw, sw, lui, addi, and blt. He decides that each instruction should be 17 bits wide, as he likes the number 17. There are no `funct7` or `funct3` fields in this new ISA.

- 2.1 What is the minimum number of bits required for the opcode field?

$$\lceil \log_2 7 \rceil = 3$$

Binary encoding, which requires least number of bits, is used here. In order to represent 7 operations, we need at least $\lceil \log_2 7 \rceil = 3$ bits.

- 2.2 Suppose Prof. Wawrzynek decides to make the `opcode` field 6 bits. If we would like to support instructions with 3 register fields, what is the maximum number of registers we could address?

The instruction is 17 bit wide, 6 bits are used for `opcode`, we have 11 bits left for register indexing. Given we need 3 register fields, we can have $\lfloor \frac{11}{3} \rfloor = 3$ bits per register field which means we could address 8 registers.

2.3 Given that the `opcode` field is 6 bits wide and each register field is 2 bits wide in the 17 bit instruction, answer the following questions:

- (a) Using the assumptions stated in the above description, how many bits are left for the immediate field for the instruction `blt` (Assume it takes `opcode`, `rs1`, `rs2`, and `imm` as inputs)?

$$17 - 6 - 2 - 2 = 7$$

`blt` has 1 `opcode` field (6), 2 register fields (2 + 2), we can use the rest $17 - 6 - 2 - 2 = 7$ bits for expressing jump offset.

- (b) Let n be your answer in part (a). Suppose that `blt`'s branch immediate is in units of instructions (i.e. an immediate of value 1 means branching 1 instruction away). What is the maximum number of bytes a `blt` instruction can jump forward from the current `pc` using these assumptions? Write your answer in terms of n .

$$(2^{n-1} - 1) * 17$$

In 2's complement, the range of an n -bit number is $[-2^{n-1}, 2^{n-1} - 1]$. jumping forward means that the offset is positive. With n -bit 2's complement offset, we can jump forward $2^{n-1} - 1$ instructions, which is $(2^{n-1} - 1) * 17$ bits since each instruction is 17 bits wide.

- (c) Using the assumptions stated in the description, what is the most negative immediate that could be used in the `addi` instruction (Assume it takes `opcode`, `rs1`, `rd`, and `imm` as inputs)?

$$-64$$

First, calculate the bit width of the immediate field, which is $17 - 6 - 2 - 2 = 7$ bits. The range of a 7-bit number in 2's complement is $[-2^6, 2^6 - 1]$ Thus, the most negative immediate is $-2^6 = -64$.

- (d) For `LUI`, we need `opcode`, `rd`, and `imm` as inputs. Using the assumptions stated in the description, how many bits can we use for the immediate value?

$$17 - 6 - 2 = 9$$

Given the `opcode` is 6 bits wide, register is 2 bits wide, we can use the rest of the bits for immediate. The width of immediate is therefore $17 - 6 - 2 = 9$.

3 Advanced RISC-V

3.1 You are given the following RISC-V code:

```
Loop:   andi t2 t1 1
        srli t3 t1 1
        bltu t1 a0 Loop
        jalr s0 s1 MAX_POS_IMM
```

- (a) What is the value of the byte offset that would be stored in the immediate field of the bltu instruction?

-8

- (b) What is the binary encoding of the bltu instruction? Please use hexadecimal to represent your answer.

0xFE A36CE3

3.2 As a curious 61C student, you question why there are so many possible opcodes, but only 47 instructions. Thus, you propose a revision to the standard 32-bit RISC-V instruction formats where each instruction has a unique opcode (which still is 7 bits). You believe this justifies taking out the funct3 field from the R, I, S, and SB instructions, allowing you to allocate bits to other instruction fields except the opcode field.

- (a) What is the largest number of registers that can now be supported in hardware?

64

- (b) With the new register size, how far can a jal instruction jump to (in halfwords)?

$[-2^{18}, 2^{18} - 1]$

- (c) Assume register `s0 = 0x1000 0000`, `s1 = 0x4000 0000`, `PC = 0xA000 0000`. Let's analyze the instruction `jalr s0, s1, MAX_POS_IMM` where `MAX_POS_IMM` is the maximum possible positive immediate for `jalr`. Using the register sizes defined above, what are the values in registers `s0`, `s1`, and `pc` after the instruction executes?

`s0 = 0xA000 0004`

`s1 = 0x4000 0000`

`pc = 0x4000 0FFF`