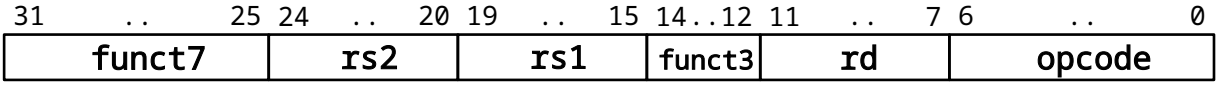
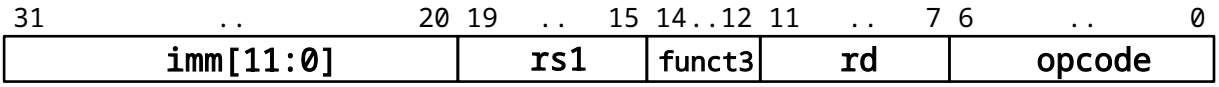
Worksheet: RISC-V Machine Language 2

In our previous lesson, we examined RISC-V ***register-to-register instructions***. These were called type-R instructions, and the format of the machine language instruction is given here.



Compare this to what is called ***immediate instructions***, which has the following format:



Rather than perform an operation on the values from two registers, an immediate instruction performs the operation on a single register, and a value stored in the instruction itself – specifically the value stored in the bits labeled `imm[11:0]`. As can be seen by the instruction format, the value can be up to 12 bits in length. For addition and subtraction, this immediate value is treated as a two's complement number.

1. What is the range of values that can be represented by a two's complement binary number of 12 bits in length?

If a programmer wishes to add or subtract a larger value, the value must be loaded into a register, then the operation can be performed with a register-to-register instruction. We will discuss instructions that load from memory later.

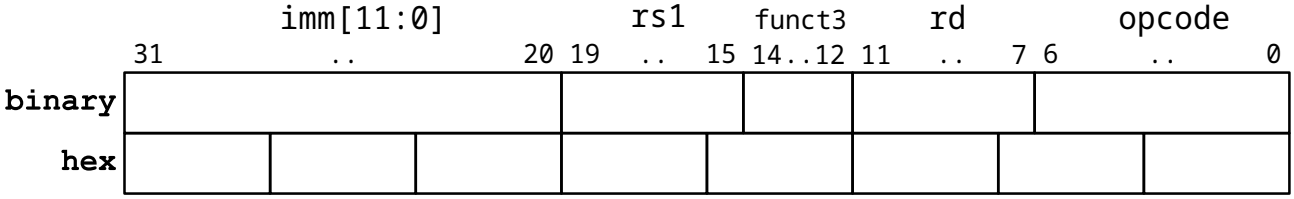
The `opcode` binary value `0010011` tells the processor that the instruction is an immediate (Type-I) instruction that involves the ALU.

The table to the right shows some of the RISC-V immediate operations that can be performed, and the values that `funct3` must be set to in order to perform these operations.

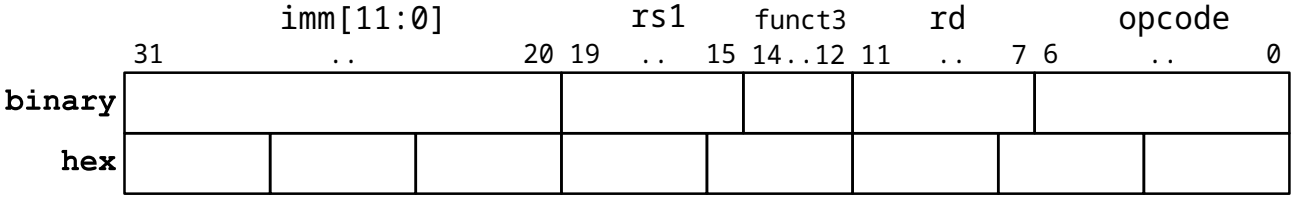
Operation	funct3	assembly
add immediate	`000`	`addi`
bitwise XOR immediate	`100`	`xori`
bitwise OR immediate	`110`	`ori`
bitwise AND immediate	`111`	`andi`

The format of a RISC-V assembly immediate instruction that involves calculation by the ALU is: `<`*`instruction`*`> <`*`rd`*`>, <`*`rs1`*`>, <`*`imm`*`>`. An example is given in question 2.

2. In high-level languages, we can check if the value in variable `x` is even by testing: "`if(x&1 == 0)`". Show the RISC-V machine code for the & operation ("`x&1`"), using the registers as given in the equivalent assembly code: "`andi x30, x31, 1`".



3a. Fill out the table below for the RISC-V machine language instruction that decrements register `x1`.



Worksheet: RISC-V Machine Language 2

3b. Write the assembly code for the instruction given in 3a.

3c Write a high-level language for the instruction in 3a.

RISC-V instructions that load from external memory are also considered Type-I instructions, but the `opcode` binary value is `0000011`.

The table below right shows the memory load instructions for RISC-V. A ***word*** in RISC-V is 32-bits, so the load doubleword instruction (`ld`) and load word unsigned instruction (`lwu`) will only be implemented in RV64 (for 64-bit processors).

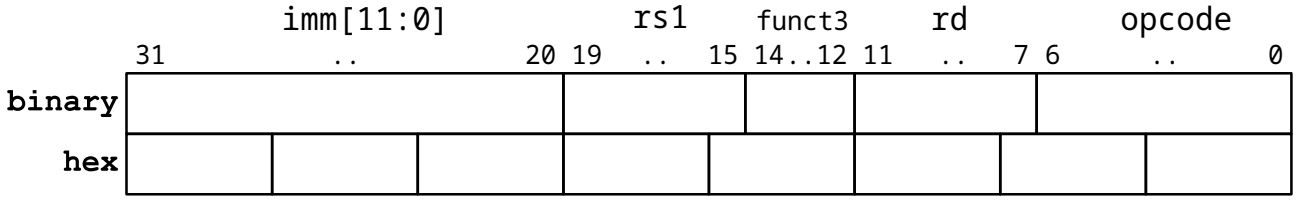
The format of a RISC-V assembly load instruction is: `<`*`instruction`*`> <`*`rd`*`>, <`*`imm`*`>(rs1)`. The value of register `rs1` is the base address of memory to load, and the `imm` value is added to this address to obtain the final address to load.

Unsigned load operations leave the upper bits zero, while the others are signed operations, so will fill the upper bits with the sign bit (the most significant bit) of the value loaded from memory.

For example, if the memory location at the address given by `x2` contains the value `0x80`, then the instruction "`lb x1, (x2)`" will result in the value `0xFFFFFF80` being loaded into register `x1`. The instruction "`lbu x1, (x2)`" will result in the value `0x00000080` being loaded into register `x1`.

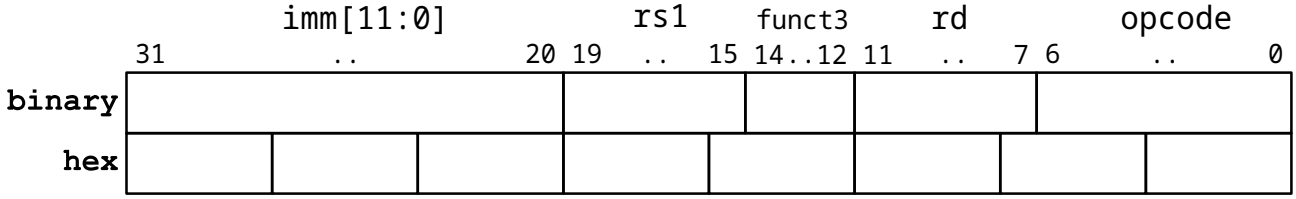
Operation	funct3	assembly
load byte	`000`	`lb`
load halfword	`001`	`lh`
load word	`010`	`lw`
load doubleword	`011`	`ld`
load byte unsigned	`100`	`lbu`
load halfword unsigned	`101`	`lhu`
load word unsigned	`110`	`lwu`

4a. Write the machine language instruction that will load the word that is four memory locations (4 bytes) before the address given by the address in register `x2`, and stores the loaded value in register `x3`.



4b. Write the assembly code for the instruction given above

5a. Write the machine language instruction that will load the half word that is two memory locations (2 bytes) after the address given by the address in register `x20`, and stores the loaded value in register `x15` without sign extension.



5b. Write the assembly code for the instruction given above.