INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

Adding a custom instruction to RISC-V.

Sparsh Mittal

Extending RISC-V

- In addition to supporting standard general-purpose software development, another goal of RISC-V is to provide a basis for more specialized instruction-set extensions or more customized accelerators.

- Any RISC-V processor implementation must support a base integer ISA (RV32I, RV32E, RV64I, or RV64E). In addition, an implementation may support one or more extensions.

Extending RISC-V

- Extensions are divided into two broad categories: *standard versus non-standard*.

- A *standard extension* is one that is generally useful and that is designed not to conflict with any other standard extension. Currently, "MAFDQCBTPV" is either a complete or planned standard extension.

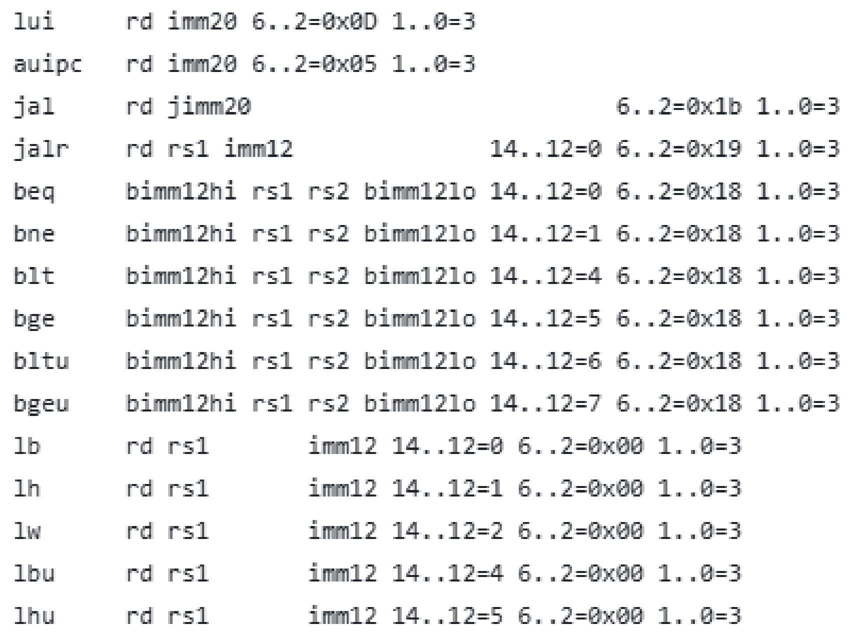
- A *non-standard extension* may be highly specialized and may conflict with other standard or non-standard extensions.

Adding a custom instruction.

- Go to the following link:

  https://github.com/riscv/riscv-opcodes/blob/master/extensions/rv_i

- Here you will be able to see the various opcodes and instruction bits assigned to various instructions (as shown below).
- If we want to add a custom instruction, we will need to provide the instruction encoding like the ones shown here.
- Note that the various fields like opcode, funct7, and funct3 specify the operation of a particular instruction.



```
lui     rd imm20 6..2=0x0D 1..0=3
auipc   rd imm20 6..2=0x05 1..0=3
jal     rd jimm20                           6..2=0x1b 1..0=3
jalr    rd rs1 imm12               14..12=0 6..2=0x19 1..0=3
beq     bimm12hi rs1 rs2 bimm12lo 14..12=0 6..2=0x18 1..0=3
bne     bimm12hi rs1 rs2 bimm12lo 14..12=1 6..2=0x18 1..0=3
blt     bimm12hi rs1 rs2 bimm12lo 14..12=4 6..2=0x18 1..0=3
bge     bimm12hi rs1 rs2 bimm12lo 14..12=5 6..2=0x18 1..0=3
bltu    bimm12hi rs1 rs2 bimm12lo 14..12=6 6..2=0x18 1..0=3
bgeu    bimm12hi rs1 rs2 bimm12lo 14..12=7 6..2=0x18 1..0=3
lb      rd rs1       imm12 14..12=0 6..2=0x00 1..0=3
lh      rd rs1       imm12 14..12=1 6..2=0x00 1..0=3
lw      rd rs1       imm12 14..12=2 6..2=0x00 1..0=3
lbu     rd rs1       imm12 14..12=4 6..2=0x00 1..0=3
lhu     rd rs1       imm12 14..12=5 6..2=0x00 1..0=3
```

Adding a custom instruction (Example).

- Now, to add an instruction like “modulo” to the ISA, we can specify the instruction and its various bits. The instruction and its semantics are given below:

Instruction:	mod r1, r2, r3
Semantics:	R[r1] = R[r2] % R[r3]

mod rd rs1 rs2 31..25=1 14..12=0 6..2=0x1A 1..0=3

Using a custom instruction in a program.

- Once the custom instruction is added to the assembler, it can be used in any program, as shown in the given example program.

```
#include <stdio.h>

int main(){
 int a,b,c;
 a = 5;
 b = 2;
 asm volatile
 (
  "mod   %[z], %[x], %[y]\n\t"
  : [z] "=r" (c)
  : [x] "r" (a), [y] "r" (b)
 )

 if ( c != 1 ){
   printf("\n[[FAILED]]\n");
   return -1;
 }

 printf("\n[[PASSED]]\n");

 return 0;
}
```

Adding the custom instruction to a particular simulator.

- Once the opcodes file is modified, other steps will be required to run the instruction in your own program. The procedure required depends on the simulator you are using. The following posts will be helpful:
    1. https://nitish2112.github.io/post/adding-instruction-riscv/
    2. https://medium.com/@viveksgt/adding-custom-instructions-compilation-support-to-riscv-toolchain-78ce1b6efcf4
    3. https://hsandid.github.io/posts/risc-v-custom-instruction/