6.823 Computer System Architecture

Predication

http://csg.csail.mit.edu/6.823/

Predication is a technique to provide conditional execution without branches and control dependences. Predication allows associating each instruction with a Boolean flag, called a predicate. If the predicate is true, the instruction executes normally; if the predicate is false, the instruction is treated as a no-op. To avoid control dependences, predicated instructions are fetched and decoded as usual, but turned into a no-op if their predicate is false.

We extend the RISC-V ISA to add predication as follows. We add four 1-bit predicate registers, `p0` to `p3`, stored in a predicate register file. We also change the encoding of RISC-V arithmetic instructions to allow them to be predicated on the value of one of these four registers.

We denote predicated instructions in assembly code by prefixing them with the predicate in parenthesis. For example:

```
(p1) ADDI rd, rs1, imm
```

denotes that this `ADDI` instruction is predicated on the value of predicate register `p1`: if `p1` is True (i.e., 1), the instruction will execute as usual, and otherwise it will be turned into a no-op.

In our ISA extension, we also allow instructions to be predicated on the inverse of a predicate register. For example:

```
(!p1) ADDI rd, rs1, imm
```

denotes that this `ADDI` instruction is predicated on the inverse of the value of register `p1`: if `p1` is False (i.e., 0), the instruction will execute as usual, and otherwise it will be turned into a no-op.

Finally, we add a new instruction, `SETPGE`, to write to the predicate register file. `SETPGE` sets a predicate register if the first register value is greater than or equal to the other:

```
SETPGE pd, rs1, rs2 ; Set PredReg[pd] to 1 if Reg[rs1] >= Reg[rs2],
                                        or to 0 otherwise
```

With these changes, we can implement conditional execution without branches. For example, consider the code `if (x >= 0) { A = B; }`. Assuming that registers `x1`, `x2`, and `x3` hold the values of `x`, `A`, and `B`, respectively, the following is the RISC-V assembly for the code with a conditional branch:

```
      BLT     x1, x0, L
      ADDI    x2, x3, 0
 L:
```

With predication, we can rewrite the above assembly without the branch. The newly introduced `SETPGE` instruction sets predicate register `p0` based on the if-condition, and the `ADDI` instruction is predicated on `p0`, so it will have an effect only when `p0` is True:

```
      SETPGE  p0, x1, x0
(p0) ADDI r2, r3, 0
```