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 instruction are fetched and decoded as usual, but turned into a no-op if their predicate is false.

We extend the MIPS 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 MIPS arithmetic and load/store instructions to allow them to be predicated on the value of one of these four registers.

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

```
(p1) ADDI rt, rs, 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 rt, rs, 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, `SETPEQ`, to write to the predicate register file. `SETPEQ` sets a predicate register if two register values are equal:

```
SETPEQ pd, rs, rt ; Set PredReg[pd] to 1 if Reg[rs] == Reg[rt],
                                    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 `r1`, `r2`, and `r3` hold the values of `x`, `A`, and `B`, respectively, the following is the MIPS assembly for the code with a conditional branch:

```
    BNEZ    r1, L
    ADDI    r2, r3, 0
L:
```

With predication, we can rewrite the above assembly without the branch. The newly introduced `SETPEQ` 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:

```
     SETPEQ  p0, r1, r0
(p0) ADDI r2, r3, 0
```