Chapter 6: Architecture

Floating-Point Instructions

RISC-V Floating-Point Extensions

- RISC-V offers three floating point extensions:
  - **RVF:** single-precision (32-bit)
    - 8 exponent bits, 23 fraction bits
  - **RVD:** double-precision (64-bit)
    - 11 exponent bits, 52 fraction bits
  - **RVQ:** quad-precision (128-bit)
    - 15 exponent bits, 112 fraction bits

Floating-Point Registers

- **32** Floating point registers
- **Width** is highest precision – for example, if RVQ is implemented, registers are 128 bits wide
- When multiple floating point extensions are implemented, the lower-precision values occupy the lower bits of the register

Floating-Point Registers

Name	Register Number	Usage
`ft0-7`	f0-7	Temporary variables
`fs0-1`	f8-9	Saved variables
`fa0-1`	f10-11	Function arguments/Return values
`fa2-7`	f12-17	Function arguments
`fs2-11`	f18-27	Saved variables
`ft8-11`	f28-31	Temporary variables

Floating-Point Instructions

- Append .s (single), .d (double), .q (quad) for precision. I.e., `fadd.s`, `fadd.d`, and `fadd.q`
- **Arithmetic operations**:

  `fadd`, `fsub`, `fdiv`, `fsqrt`, `fmin`, `fmax`, multiply-add (`fmadd`, `fmsub`, `fnmadd`, `fnmsub`)
- **Other instructions:**

  move (`fmv.x.w`, `fmv.w.x`)

  convert (`fcvt.w.s`, `fcvt.s.w`, etc.)

  comparison (`feq`, `flt`, `fle`)

  classify (`fclass`)

  sign injection (`fsgnj`, `fsgnjn`, `fsgnjx`)

See Appendix B for additional RISC-V floating-point instructions.

Floating-Point Multiply-Add

- `fmadd` is the most critical instruction for signal processing programs.
- Requires four registers.

```
fmadd.f f1, f2, f3, f4      # f1 = f2 x f3 + f4
```

Floating-Point Example

C Code

```
int i;
float scores[200];



for (i=0; i<200; i=i+1)



  scores[i]=scores[i]+10;
```

RISC-V assembly code

```
# s0 = scores base address, s1 = i
  addi s1, zero, 0          # i = 0
  addi t2, zero, 200        # t2 = 200
  addi t0, zero, 10         # ft0 = 10.0
  fcvt.s.w ft0,  t0
for:
  bge    s1, t2, done       # i>=200? done
  slli   t0, s1, 2          # t0 = i*4
  add    t0, t0, s0         # scores[i] address
  flw    ft1, 0(t0)         # ft1=scores[i]
  fadd.s ft1, ft1, ft0      # ft1=scores[i]+10
  fsw    ft1, 0(t0)         # scores[i] = t1
  addi   s1, s1, 1          # i = i+1
  j      for                # repeat
done:
```

Floating-Point Instruction Formats

- Use R-, I-, and S-type formats
- Introduce another format for multiply-add instructions that have 4 register operands: R4-type

R4-Type