LAB 09: Floating-Point

Saleh AlSaleh

salehs@kfupm.edu.sa

King Fahd University of Petroleum and Minerals
College of Computing and Mathematics
Computer Engineering Department

COE301: Computer Architecture
Term 222

Agenda

IEEE 754 Standard

- S = Sign Bit (0 positive, 1 negative)

IEEE 754 Standard

- S = Sign Bit (0 positive, 1 negative)
- E = Exponent Bits (8 Single, 11 Double Precession)

IEEE 754 Standard

- S = Sign Bit (0 positive, 1 negative)
- E = Exponent Bits (8 Single, 11 Double Precession)
- F = Fraction Bits (23 Single, 52 Double Precision)

IEEE 754 Standard

- S = Sign Bit (0 positive, 1 negative)
- E = Exponent Bits (8 Single, 11 Double Precession)
- F = Fraction Bits (23 Single, 52 Double Precision)
- Bias (127 Single, 1023 Double)

IEEE 754 Standard

- S = Sign Bit (0 positive, 1 negative)
- E = Exponent Bits (8 Single, 11 Double Precession)
- F = Fraction Bits (23 Single, 52 Double Precision)
- Bias (127 Single, 1023 Double)
- Normalized Value = $\pm(1.F)_2 \times 2^{E-Bias}$

S	Exponent	Fraction

Single-Precession Floating Point

Coprocessor 1

- Coprocessor 1 has 32 floating-point registers (32 bit each).

Coprocessor 1

- Coprocessor 1 has 32 floating-point registers (32 bit each).
- These registers are numbered as \$f0-\$f31.

Coprocessor 1

- Coprocessor 1 has 32 floating-point registers (32 bit each).
- These registers are numbered as \$f0-\$f31.
- Each register can hold one single-precision floating-point number.

Coprocessor 1

- Coprocessor 1 has 32 floating-point registers (32 bit each).
- These registers are numbered as \$f0-\$f31.
- Each register can hold one single-precision floating-point number.
- The double-precision number uses two registers and is stored in an even-odd pair of registers, but we only refer to the even-numbered register.

Coprocessor 1

- Coprocessor 1 has 32 floating-point registers (32 bit each).
- These registers are numbered as $f0-$f31.
- Each register can hold one single-precision floating-point number.
- The double-precision number uses two registers and is stored in an even-odd pair of registers, but we only refer to the even-numbered register.
- There are 8 condition flags, numbered from 0 to 7 used by floating-point compare and branch instructions.

Floating Point Instructions

Instruction	Description
lwc1 or l.s	Load a word from memory to a single-precision floating-point register
ldc1 or l.d	Load a double word from memory to a double-precision register
swc1 or s.s	Store a single-precision floating-point register in memory
sdc1 or s.d	Store a double-precision floating-point register in memory
add.s, add.d	Floating Point Addition (Single, Double)
sub.s, sub.d	Floating Point Subtraction (Single, Double)
mul.s, mul.d	Floating Point Multiplication (Single, Double)
div.s, div.d	Floating Point Division (Single, Double)
sqrt.s, sqrt.d	Floating Point Square Root (Single, Double)
abs.s, abs.d	Floating Point Absolute Value (Single, Double)

Floating Point Instructions

Instruction	Description
neg.s, neg.d	Floating Point Negative Value (Single, Double)
mov.s, mov.d	Copy floating point value from one register to another (Single, Double)
cvt.s.w	Convert from word (integer) to single precision floating point
cvt.s.d	Convert from double precision to single precision floating point
cvt.d.w	Convert from word (integer) to double precision floating point
cvt.d.s	Convert from single precision to double precision floating point
cvt.w.s	Convert from single precision to word (integer)
cvt.w.d	Convert from double precision to word (integer)
ceil.w.s, ceil.w.d	Integer ceiling (Single, Double)
floor.w.s, floor.w.d	Integer floor(Single, Double)
trunc.w.s, trunc.w.d	Truncate (Single, Double)

Floating Point Conditional Instructions

Instruction	Example	Description
c.eq.s c.eq.d	c.eq.s $f0, $f1 c.eq.d 3, $f2, $f4	If ($f0== $f1), set flag 0 to true, else false If ($f2== $f4), set flag 3 to true, else false
c.lt.s c.lt.d	c.lt.s $f0, $f1 c.lt.d 4, $f2, $f4	If ($f0< $f1), set flag 0 to true, else false If ($f2< $f4), set flag 4 to true, else false
c.le.s c.le.d	c.le.s $f0, $f1 c.le.d 5, $f2, $f4	If ($f0<= $f1), set flag 0 to true, else false If ($f2<= $f4), set flag 5 to true, else false
bc1t	bc1t **loop** bc1t 6, **while**	Branch to **loop** if condition flag 0 is true Branch to **while** if condition flag 6 is true
bc1f	bc1f **loop** bc1f 7, **while**	Branch to **loop** if condition flag 0 is false Branch to **while** if condition flag 7 is false

FP Register Convention

Registers	Usage
$f0 - $f3	Floating-point procedure results
$f4 - $f11	Temporary floating-point registers, NOT preserved across procedure calls
$f12 - $f15	Floating-point parameters, NOT preserved across procedure calls. Additional floating-point parameters should be pushed on the stack.
$f16 - $f19	More temporary registers, NOT preserved across procedure calls.
$f20 - $f31	Saved floating-point registers. Should be preserved across procedure calls.

Live Examples

Task #1

Write a MIPS assembly program that reads two double-precision Floating-Point numbers from the user **x** & **y**. Then, perform the operation $\frac{x}{y}$. If the result of the division is less than 0, perform $3.14\sqrt{-\frac{x}{y}}$. Otherwise, perform $\sqrt{8\frac{x}{y}}$. Finally, print the result.

Sample Run 1

```
Enter double x: 4
Enter double y: 2
The result is 4.0
```

Sample Run 2

```
Enter double x: -1
Enter double y: 1
The result is 3.14
```

Task #2

Write a MIPS assembly program that reads 12 single-precision Floating-Point numbers from the user representing the grades of a quiz taken by 12 student and report back the average.

Sample Run

```
Enter grade 0: 7.25
Enter grade 1: 6.5
Enter grade 2: 10
Enter grade 3: 9
Enter grade 4: 2.75
Enter grade 5: 8.5
Enter grade 6: 7.75
Enter grade 7: 10
Enter grade 8: 9.5
Enter grade 9: 9.75
Enter grade 10: 8.25
Enter grade 11: 8.75
The average of the 12 grades is: 8.166667
```