

CS 590: Topics in Computer Science

Assignment 10: Floating Point and Number Representation

Goals

This lab should help you examine how computers store integer and floating point values.

Info Recall that the single precision floating point number is stored as:

SEEE **E**EEE **E**IIII **I**IIII **I**IIII **I**IIII **I**IIII **I**IIII

where:

S is the sign bit, 0 for positive, 1 for negative. **E** is the exponent, bias 127. **I** is the significant, with an implicit 1

For example, the floating point representation of 1.0 would be 0x3F800000. Verify to yourself that this is correct.

Exercise 1: Integers

Find the shortest sequence of MIPS instructions to determine if there is a carry out from the addition of two registers, say \$t3 and \$t4. Place a 0 or 1 in register \$t2 if the carry out is 0 or 1, respectively. (This can be done in just two instructions). Verify that your code works for the following values:

Operand	Operand	Carry out?
0x7fffffff	0x80000000	no
0xffffffff	0	no
0xffffffff	1	yes

Exercise 2: Floating Point

Find a positive floating point value x , for which $x+1.0=x$. Verify your result in a MIPS assembly language program, and determine the stored exponent and fraction for your x value (either on the computer or on paper).

Note: The provided MIPS program `pfloat2.s` will allow you to experiment with adding floating point values. It leaves the output in `$f12` and also `$s0`, so you can examine the hex representation of the floating point value by printing out `$s0`.

...

`num1: .word 0x3F800000`

`num2: .float 1.0`

`result: .word 0`

`string: .asciiz "\n"`

`.text`

`main:`

`la $t0, num1`

`lwc1 $f2, 0($t0)`

`lwc1 $f4, 4($t0)`

...

`# Do the actual addition`

`add.s $f12, $f2, $f4`

`# Transfer the value from the floating point reg to the integer reg`

`swc1 $f12, 8($t0)`

`lw $s0, 8($t0)`

`# At this point, $f12 holds the sum, and $s0 holds the sum which can
be read in hexadecimal`

...

Exercise 3: Floating Point

Next, find the smallest positive floating point value x for which $x+1.0=x$. Again, determine the stored exponent and fraction for x .

In Exc 2:

$x \Rightarrow \$f2$ ($0x3F800000 = 1.0$)

$0.1 \Rightarrow \$f4$ ($0x3F800000 = 1.0$)

Result $\Rightarrow \$f12, \$s0$ ($0x40000000 = 2.0$)

Let us look at the content of registers through video 10-3.mp4

Exercise 4: Floating Point Associativity

Finally, using what you have learned from the last two parts, determine a set of positive floating point numbers such that adding these numbers in a different order can yield a different value. You can do this using only three numbers. (Hint: Experiment with adding up different amounts of the x value you determined in part 3, and the value 1.0).

This shows that for three floating point numbers a , b , and c , $a+b+c$ does not necessarily equal $c+b+a$.

Write a program to add these three values in different orders. It should be a straightforward modification of the program from part 2-3.

EnAssembly/Lab 09_10/source/pfloat32.s - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0400000	0x3c011001	lui \$1,0x00001001	13: la \$t0, num1
	0x0400004	0x34280000	ori \$0,\$1,0x00000000	
	0x0400008	0xc5020000	lwc1 \$f2, 0(\$t0)	14: lwc1 \$f2, 0(\$t0)
	0x040000c	0xc5040004	lwc1 \$f4, 0(\$t0)	15: lwc1 \$f4, 4(\$t0)
	0x0400010	0xc5060008	lwc1 \$f6, 0x00000008...	16: lwc1 \$f6, 8(\$t0)
	0x0400014	0x24020002	addiu \$2,\$0,0x00000002	20: li \$v0, 2
	0x0400018	0x44001300	mov.s \$f13,\$f2	21: mov.s \$f13, \$f2
	0x040001c	0x0000000c	syscall	22: syscall
	0x0400020	0x24020004	addiu \$2,\$0,0x00000004	24: li \$v0, 4
	0x0400024	0x3c011001	lui \$1,0x00001001	25: la \$a0, string
	0x0400028	0x34240014	ori \$4,\$1,0x00000014	
	0x040002c	0x0000000c	syscall	26: syscall
	0x0400030	0x24020002	addiu \$2,\$0,0x00000002	28: li \$v0, 2
	0x0400034	0x44002300	mov.s \$f13,\$f4	29: mov.s \$f13, \$f4
	0x0400038	0x0000000c	syscall	30: syscall
	0x040003c	0x24020004	addiu \$2,\$0,0x00000004	32: li \$v0, 4
	0x0400040	0x3c011001	lui \$1,0x00001001	33: la \$a0, string
	0x0400044	0x34240014	ori \$4,\$1,0x00000014	
	0x0400048	0x0000000c	syscall	34: syscall
	0x040004c	0x24020002	addiu \$2,\$0,0x00000002	36: li \$v0, 2
	0x0400050	0x44003300	mov.s \$f13,\$f6	37: mov.s \$f13, \$f6
	0x0400054	0x0000000c	syscall	38: syscall
	0x0400058	0x24020004	addiu \$2,\$0,0x00000004	40: li \$v0, 4
	0x040005c	0x3c011001	lui \$1,0x00001001	41: la \$a0, string
	0x0400060	0x34240014	ori \$4,\$1,0x00000014	
	0x0400064	0x0000000c	syscall	42: syscall
	0x0400068	0x24020004	addiu \$2,\$0,0x00000004	44: li \$v0, 4
	0x040006c	0x3c011001	lui \$1,0x00001001	45: la \$a0, string
	0x0400070	0x34240014	ori \$4,\$1,0x00000014	
	0x0400074	0x0000000c	syscall	46: syscall
	0x0400078	0x44041300	add.s \$f13,\$f2,\$f4	50: add.s \$f13, \$f2, \$f4
	0x040007c	0x44040300	add.s \$f13,\$f13,\$f6	51: add.s \$f13, \$f13, \$f6
	0x0400080	0x44040340	add.s \$f13,\$f6,\$f4	52: add.s \$f13, \$f6, \$f4
	0x0400084	0x44020b40	add.s \$f13,\$f13,\$f2	54: add.s \$f13, \$f13, \$f2
	0x0400088	0xc500000c	swcl \$f12,0x00000000...	58: swcl \$f12, 12(\$t0)

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10110000
\$v0	2	0x00000004
\$v1	3	0x00000000
\$a0	4	0x10010014
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$a0	8	0x10010000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t0	16	0x00000000
\$t1	17	0x00000000
\$t2	18	0x00000000
\$t3	19	0x00000000
\$t4	20	0x00000000
\$t5	21	0x00000000
\$t6	22	0x00000000
\$t7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$k2	28	0x10000000
\$k3	29	0xfffffff0
\$k4	30	0x00000000
\$k5	31	0x00000000
\$pc		0x04000000
\$hi		0x00000000
\$lo		0x00000000

Mars Messages

Run IO

1.0
2.0
2.1
5.1
5.1

Clear

--- program is finished running (dropped off bottom) ---

Questions/Tasks:

1. How come that Out-of-Order execution is an indispensable performance feature? Explain.
2. Explain how Meltdown exploits the side effects of out-of-order execution.
3. Explain how come that the Meltdown attack is independent of the operating system, and it does not rely on any software vulnerabilities.
4. What's the main difference between Spectre and Meltdown exploits? Explain.