

Dr. Chase's lecture notes for CSC 281

Finishing up f.p.

- Assign to read: Section 3.8, and re-read Section 3.6.
- Using the transparency of the program `float.s`, show `.float` and `.double`
- Smallest normalized float difference is about 0.00000006, or 7 decimal digits precision (exactly 24 bits precision).
- To add f.p. numbers, adjust exponent of smaller: $9.99 \times 10^6 + 2.20 \times 10^4 = 1.00 \times 10^7$ (show). [Use example on p. 214] You don't need to know round bit, sticky bit, and guard bit. You do need to know that the result is that the rounding will come out the same as though the precision were infinite before it was rounded.
- To multiply, you might just think that you multiple mantissas and add exponents, and you'd be right: $3.1 \times 10^4 * 2.0 \times 10^3 = 6.1 \times 10^7$, but internally because the exponent is in biased notation, the hardware must subtract one bias (127).
- `l.s` is a pseudo-op that allows you to load a number in directly. Watch:

Number: `.float -3.2e15`

	<u>RAM/general registers</u>	<u>RAM/fp registers</u>	
	<code>lw \$t0, const(\$t1)</code>	<code>lwc1 \$f0, const(\$t1)</code>	similarly <code>ldc1</code>
	<code>sw \$t0, const(\$t1)</code>	<code>swc1 \$f0, const(\$t1)</code>	similarly <code>swc1</code>
Pseudo-ops:			
	<code>lw \$t1, Number</code>	<code>l.s \$f0, Number</code>	similarly <code>l.d, s.d</code>
=>	<code>la \$t1, Number</code>	=> <code>la \$at, Number</code>	
	<code>lw \$t1, 0(\$t1)</code>	<code>lwc1 \$f0, 0(\$at)</code>	

* Note that `lw` can be an instruction or a pseudo-op depending on what goes as the second address.

* Note also that `la` is itself a pseudo-op, so **might** use `$at`, and therefore mess things up, but in fact `la $t3, Number` expands to

```
lui $t3, Number16:31
ori $t3, Number0:15
```

which the assembler can do because it has access to the numerical value of `Number`, because the assembler decided where to put `Number`. Notice that `$at` is not used.

- f.p. addition is not associative! [See p. 221 for an example.]

Consequence: if you're going to add a long column of f.p. numbers, sort them and add them in order from smallest to largest, so the “big + small = big” problem doesn't swamp the little numbers.

Note: Print out sections B.1 through B.5 and B.7 through B.9. It's in an appendix because it's "review," but we don't have a Discrete Mathematics course (yet), so we're doing just-in-time use of the needed mathematics. I am lecturing on that material. You should begin to read it.

Boolean Algebra, an Introduction

Boole was a Unitarian back in the day when Unitarians believed that Christ was their Savior, although not that He was divine. Boole contemplated $x^2=x^1$. He was dualistic, seeing the battle of good and evil (as his wife Mary Everest Boole reports). This led to his book *Laws of Thought*, 1854, built on W. Gottfried Leibniz's dream for a "calculus" of rational thought.

Model Theory: use a universe. In our case, the universe is {T, F} for true and false. But in some Engineering applications, there are tri-state buffers, which might use {T, F, Unsure}. Things are provable if they are true in this small universe.

Do truth tables. Recall **not**

0	1	0	0	1	1	0	0
1	0	1	1	1	1	1	1

Boolean algebra describes a lot of things. Here is a dictionary.

	not		*		+		0		1
electricity	inverter		series		parallel		off		on
sets	complement		intersection		union		empty set		universe
logic	not		and		or		false		true
numbers	1-x		x*y		x+y-x*y		0		1

But besides model theory, there is proof theory, and proof theory requires axioms. Here they are (p. B-6)

Identity	$A + 0 = A$	$A * 1 = A$
Zero and one	$A + 1 = 1$	$A * 0 = 0$
Inverse	$A + A' = 1$	$A * A' = 0$
Commutative	$A + B = B + A$	$A * B = B * A$
Associative	$A + (B + C) = (A + B) + C$	$A * (B * C) = (A * B) * C$
Distributive	$A * (B + C) = A * B + A * C$	$A + (B * C) = (A + B) * (A + C)$
		^ surprise!

Do DeMorgan's laws two ways, first by model theory, then by proof theory.