Anita's Super Awesome Recitation Slides

15/18-213: Introduction to Computer Systems
Datalab and Floating Point, 28 Jan 2013

Anita Zhang, Section M

FAQ and Administrative Stuff

- Staff: 15-213-staff@cs.cmu.edu
- Data lab due this Thursday, 31 Jan 2013
- FAQ on the main site
- Questions?
- Progress?
- Autolab?
- Shark? > ssh shark.ics.cs.cmu.edu

Because Everyone Needs a Guide..

- Literature
- Bits and Bytes
- IEEE Floating Point
  - Review
  - Examples from past midterms
- Datalab Hints
- Special Notice
- More Question Time

Books I Like

- Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective, Second Edition*, Prentice Hall, 2011
- Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall, 1988
- Koenig, Andrew. *C Traps and Pitfalls*. Reading, MA: Addison-Wesley, 1988
- Kernighan, Brian W., and Rob Pike. *The Practice of Programming*. Reading, MA: Addison-Wesley, 1999

Random int Stuff

Random int Stuff

- Multiplication/ division by 2^k
  - Multiply => left shift by k
  - Division => right shift by k

Random int Stuff

- …I lied. Kind of.

Random int Stuff

- Division of a negative number by 2^k
  - Needs a bias
  - Looks like this: (x + (1 << k) - 1) >> k

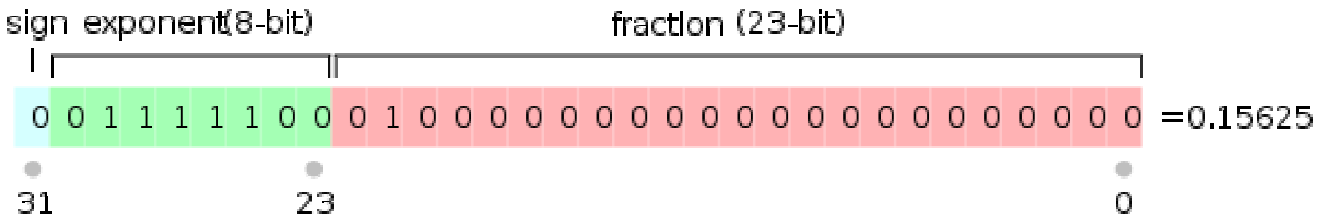
Random int Stuff

- Endianness is real
- But you won't see it in datalab

Endian	First byte (lowest address)	Middle bytes	Last byte (highest address)
big	*Most* significant	...	*least*significant
little	*Least* significant	...	*most* significant

- Random example: 0x59645322
  - Big: 59 64 53 22
  - Little: 22 53 64 59

(Quick and dirty) Floating Point



- For single precision (32 bit) floating point:
  - Fraction (frac): 23 bits
  - Exponent (exp): 8 bits
  - Sign (s): 1 bit
  - Bias = 127
- In general:
  - Value = $(-1)^s \times M \times 2^E$
  - Bias = $2^{k-1} - 1$

(Quick and dirty) Floating Point

Normalized

- exp =/= 00…0
- exp =/= 11…1
- E = exp – bias
- M = 1.xxxxxx
  - xxxxxx is the frac
  - Implied leading 1

Denormalized

- exp = 00..0
- E = 1 – bias
- M = 0.xxxxxx
  - xxxxxx is the frac
  - Leading 0
  - frac = 0 means ±0

Special cases

Infinity

- exp = 11….1
- frac = 00…0
  - Division by 0, ± ∞

Not a Number

- exp = 11….1
- frac =/= 00…0
  - sqrt(-1), ∞ - ∞, ∞ x 0

Special cases

- BTW, infinity and NaN are not the same.
  - Be aware of this for float_abs()

Legit Floating Point Rules

- Rounding
  - Rounds to even
    - To avoid statistical bias
    - 1.1011 => 1.11 (>1/2 up)
    - 1.1010 => 1.10 (1/2 down)
    - 1.0101 => 1.01 (>1/2 down)
    - 1.0110 => 1.10 (1/2 up)
- Addition and Multiplication…
  - Are lies
    - Associativity/ distributivity may not hold
    - 3.14 + (1e20 – 1e20) vs. (3.14 + 1e20) – 1e20

Floating Point on Exams

- Let's pretend we have a 5-bit floating point representation with no sign bit… (sadness)
  - k = 3 exponent bits (bias = 3)
  - n = 2 fraction bits

Value	Floating Point Bits	(Rounded) Value
9/32	001 00	1/4
3		
9		
3/16		
15/2		

Floating Point on Exams

- Let's pretend we have a 5-bit floating point representation with no sign bit… (sadness)
  - k = 3 exponent bits (bias = 3)
  - n = 2 fraction bits

Value	Floating Point Bits	(Rounded) Value
9/32	001 00	1/4
3	100 10	3
9	110 00	8
3/16	000 11	3/16
15/2	110 00	8

Floating Point on Exams

- Consider two 7 bit floating point representations based on the IEEE format. Neither has a sign bit.
- Format A
  - k = 3 exponent bits (bias = 3)
  - n = 4 fraction bits
- Format B
  - k = 4 exponent bits (bias = 7)
  - n = 3 fraction bits

Format A	Format B
011 0000	0111 000
101 1110	
010 1001	
110 1111	
000 0001	

Floating Point on Exams

- Consider two 7 bit floating point representations based on the IEEE format. Neither has a sign bit.
- Format A
  - k = 3 exponent bits (bias = 3)
  - n = 4 fraction bits
- Format B
  - k = 4 exponent bits (bias = 7)
  - n = 3 fraction bits

Format A	Format B
011 0000	0111 000
101 1110	1001 111
010 1001	0110 100
110 1111	1011 000
000 0001	0001 000

Datalab Other Stuff

- Use the tools
  - ./driver.pl
  - ./bddcheck/check.pl
    - Exhaustive
  - ./btest
    - Not exhaustive
  - ./dlc
    - This one will hate you if you're not writing C like it's 1989
    - Declare all your variables at the beginning of the function

Datalab Other Stuff

- Operator Precedence
  - There are charts. Google them.
- bang()
  - "What's so special about 0 and T_{min} anyways?"
- subOK()
  - When is it possible to overflow with subtraction? Addition?
- Undefined behavior
  - Shifting by 32… omg /* insert rant here */

Undefined Behavior (Adv. Topic)

"These instructions shift the bits in the first operand (destination operand) to the left or right by the number of bits specified in the second operand (count operand). Bits shifted beyond the destination operand boundary are first shifted into the CF flag, then discarded. At the end of the shift operation, the CF flag contains the last bit shifted out of the destination operand.

The destination operand can be a register or a memory location. The count operand can be an immediate value or register CL. The count is masked to five bits, which limits the count range to 0 to 31. A special opcode encoding is provided for a count of 1."

Warning

- TAs reserve the right to dock style points for code that is poorly commented, has no sense of variable names, lacks any form of coherent indentation, and of course, contains lines that are longer than 80 characters.

Questions & Credits Page

- http://www.superiorsilkscreen.com
- http://www.wikipedia.org/
- http://www.cs.cmu.edu/~213/
- Intel x86 Instruction Set Reference