Lecture 11: Floating Point, Digital Design

- Today's topics:
  - FP formats, arithmetic
  - Intro to Boolean functions

- Exam reminders:
  - 1 sheet of notes, plus green sheet
  - No phones, simple calculators ok
  - 10:40 – 12:10, attempt every question
  - Practice exam posted tomorrow
  - Study tips
  - HW 1-4, until Lecture 8 / Slide 6

Value inf — 2 special cases up top that use the reserved exponent field of 255 — 0 255 00…0

Value NAN — 0 255 xx….x

Highest value ~2 x 2^{127} — 0 254 11….1

$\times 2^5$

Value 1 — 0 127 00…0

Exponent field < 127, i.e., after subtracting bias, they are negative exponents, representing numbers < 1

Smallest Norm ~2 x 2^{-126} — 0 0..01 00…0

Largest Denorm ~1 x 2^{-126} — 0 0..00 11…1

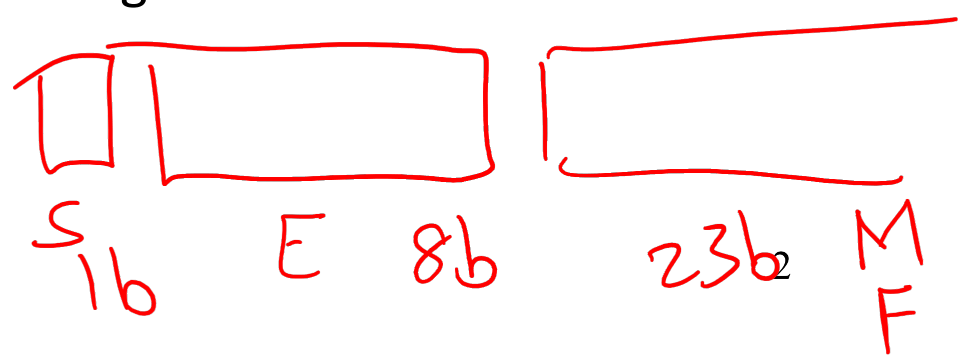
Special case with exponent field 0, used to represent denorms, that help us gradually approach 0

Smallest Denorm ~2^{-149} — 0 0..00 00…1

Value 0 — 0 00..0 00…0

Same rules as above, but the sign bit is 1

Same magnitudes as above, but negative numbers



$4.2\cancel{.}321 \times 10^{1} = 42.321$

leftmost

$42 \div 2 = 21$ rem 0

$21 \div 2 = 10$ rem 1

$10 \div 2 = 5$ rem 0

$5 \div 2 = 2$ rem 1

$2 \div 2 = 1$ rem 0

$1 \div 2 = 0$ rem 1

$0 \div 2 = 0$ rem 0

$.321 \times 2 = 0.642$

$0.642 \times 2 = 1.284$

$0.284 \times 2 = 0.568$

$0.568 \times 2 = 1.136$

$0.136 \times 2 = 0.272$

rightmost

128 4

1000 0100

Remember:

5	+127 →	132
True exponent	← -127	Exponent in register

$0101010.01010\ldots$

$1.0101001010\ldots \times 2^{5}$

$\boxed{0}\ \boxed{1000\ 0100}\ \boxed{0101001010\ldots}$

SIGN

Examples

Final representation: $(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

- Represent -0.75_{ten} in single and double-precision formats

Single: (1 + 8 + 23)

Double: (1 + 11 + 52)

Remember:

$$\text{True exponent} \underset{-127}{\overset{+127}{\rightleftarrows}} \text{Exponent in register}$$

- What decimal number is represented by the following single-precision number?

1 1000 0001 01000…0000

1 1000 0001 0100 0

FP Register

negative

128 + 1 = 129 Exp in Reg

-127

2 True exp

-1.01×2^2 = -101.00 (binary) = -5.0

implicit 1

Remember:

True exponent $\xrightarrow{+127}$ Exponent in register

True exponent $\xleftarrow{-127}$ Exponent in register

-1.01

1 1/2 1/4

$= -1.25 \times 4 = -5$

Examples

Final representation: $(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

- Represent -0.75_{ten} in single and double-precision formats

  Single: (1 + 8 + 23)
  1 0111 1110 1000…000

  Double: (1 + 11 + 52) 1023
  1 0111 1111 110 1000…000

- What decimal number is represented by the following single-precision number?
  1 1000 0001 01000…0000
  -5.0

FP Addition

- Consider the following decimal example (can maintain only 4 decimal digits and 2 exponent digits)

$9.999 \times 10^{1} \quad + \quad 1.610 \times 10^{-1}$

Convert to the larger exponent:

$9.999 \times 10^{1} \quad + \quad 0.016 \times 10^{1}$

Add

10.015×10^{1}

Normalize

1.0015×10^{2}

Check for overflow/underflow

Round

1.002×10^{2}

Re-normalize

shrink $\quad 001.61 \times 10^{-1}$

$0.016 \times 10^{+1}$ grown

$$\begin{array}{r}
9.999 \times 10^{1} \\
0.016 \times 10^{1} \\
\hline
10.015 \times 10^{1} \\
1.0015 \times 10^{2}
\end{array}$$

FP Addition

- Consider the following decimal example (can maintain only 4 decimal digits and 2 exponent digits)

$9.999 \times 10^1 \quad + \quad 1.610 \times 10^{-1}$

Convert to the larger exponent:

$9.999 \times 10^1 \quad + \quad 0.016 \times 10^1$

Add

10.015×10^1

Normalize

1.0015×10^2

Check for overflow/underflow

Round

1.002×10^2

Re-normalize

If we had more fraction bits, these errors would be minimized

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ } \\ 1\ 1\ 1 \\ 1\ 0\ 1 \\ \hline 1\,1\,0\,0 \end{array}$$

FP Addition – Binary Example

$0 \quad 0 \quad 1.01 \times 2^1$

- Consider the following binary example

0.0101×2^3

$1.010 \times 2^1 \quad + \quad 1.100 \times 2^3$

Convert to the larger exponent:

$0.0101 \times 2^3 \quad + \quad 1.1000 \times 2^3$

Add

1.1101×2^3

Normalize

1.1101×2^3

Check for overflow/underflow

Round

Re-normalize

IEEE 754 format: 0 10000010 11010000000000000000000

(room for 4 bits right of binary point)

1.1000×2^3

0.0101×2^3

1.1101×2^3

+127

130

$1.1101 \rightarrow$ round

1.111

FP Multiplication

$\times 2^3$

2^{10}

$\times 2^7$

- Similar steps: Addition (sub one of the biases)
  - Compute exponent (careful!)
  - Multiply significands (set the binary point correctly)
  - Normalize
  - Round (potentially re-normalize)
  - Assign sign

Exp + Bias

+ Bias

Exp

+Bias

1.23×10^3

4.71×10^4

$\times 10^7$

MIPS Instructions

single-prec 32b

double prec 64b

- add.s, add.d, and similarly for sub, mul, div

add.d $f0, $f2, $f4

- Comparison instructions: c.eq.s, c.neq.s, c.lt.s….
  These comparisons set an internal bit in hardware that is then inspected by branch instructions: bc1t, bc1f

C7

- Separate register file $f0 - $f31 : a double-precision value is stored in (say) $f4-$f5 and is referred to by $f4

c.eq.s

add

add

sub

bc1t

- Load/store instructions (lwc1, swc1) must still use integer registers for address computation

c.eq.s $f3, $f6

bc1t L1

Code Example

lwc1 $f2, 8($gp)

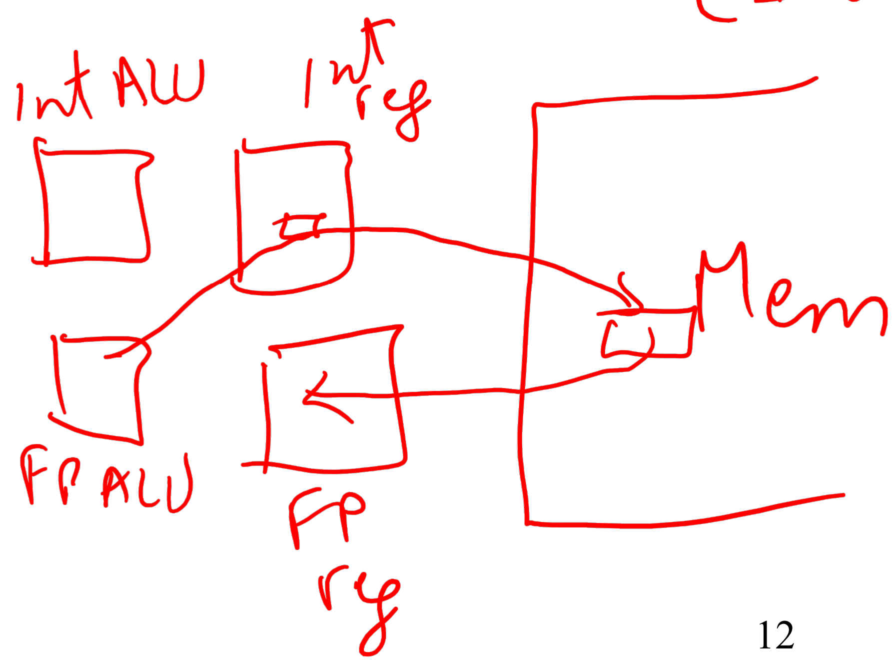
FP dest reg

Int src reg to calculate addr (Int)

```
float  f2c (float fahr)
{
    return ((5.0/9.0) * (fahr – 32.0));
}
```

(argument fahr is stored in $f12, return value in $f0)

```
lwc1   $f16, const5
lwc1   $f18, const9
div.s  $f16, $f16, $f18
lwc1   $f18, const32
sub.s  $f18, $f12, $f18
mul.s  $f0, $f16, $f18
jr     $ra
```



Fixed Point

Int - Add - 1 cycle
FP-Add - 4 cycles

- FP operations are much slower than integer ops
- Fixed point arithmetic uses integers, but assumes that every number is multiplied by the same factor
- Example: with a factor of 1/1000, the fixed-point representations for 1.46, 1.7198, and 5624 are respectively 1460, 1720, and 5624000

0.00234
2

- More programming effort and possibly lower precision for higher performance

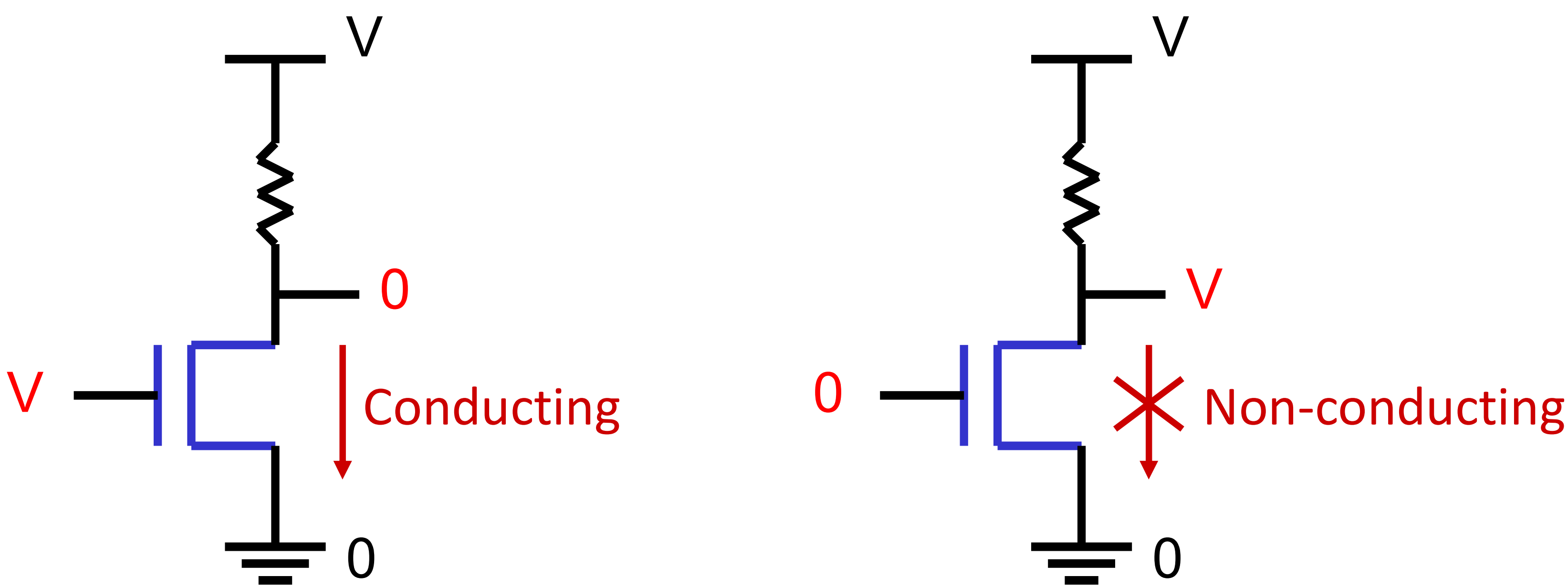
Subword Parallelism

- ALUs are typically designed to perform 64-bit or 128-bit arithmetic
- Some data types are much smaller, e.g., bytes for pixel RGB values, half-words for audio samples
- Partitioning the carry-chains within the ALU can convert the 64-bit adder into 4 16-bit adders or 8 8-bit adders
- A single load can fetch multiple values, and a single add instruction can perform multiple parallel additions, referred to as subword parallelism

Digital Design Basics

- Two voltage levels – high and low (1 and 0, true and false)
  Hence, the use of binary arithmetic/logic in all computers

- A transistor is a 3-terminal device that acts as a switch



Logic Blocks

- A logic block has a number of binary inputs and produces a number of binary outputs – the simplest logic block is composed of a few transistors

- A logic block is termed *combinational* if the output is only a function of the inputs

- A logic block is termed *sequential* if the block has some internal memory (state) that also influences the output

- A basic logic block is termed a *gate* (AND, OR, NOT, etc.)

We will only deal with combinational circuits today

Truth Table

- A truth table defines the outputs of a logic block for each set of inputs
- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

A	B	C	E

Truth Table

- A truth table defines the outputs of a logic block for each set of inputs

- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

A	B	C	E
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Can be compressed by only representing cases that have an output of 1