- Work remainder into 64 bit register
- Work divisor into upper half of 64-bit register

Version 1

Hardware:

64 bit

Divisor

shift right

\[ \begin{array}{c}
\text{Alu} \\
64 \\
\text{Remainder} \\
64 bit
\end{array} \]

subtract/advance

32 bit

shift left

Quotient

\[ \begin{array}{c}
\text{Control} \\
\text{with} \\
\text{negative?}
\end{array} \]

Version 2

Instead of shifting divisor to the right, shift remainder to left.
- Same effect
- 32 bit divisor, 32-bit Alu

32 bit

Divisor

32 bit

\[ \begin{array}{c}
\text{Alu} \\
32 \\
\text{Remainder} \\
32 bit
\end{array} \]

shift left

Control

\[ \begin{array}{c}
\text{with} \\
\text{negative?}
\end{array} \]
Signed Division

Note: \(|\text{quotient} \times |\text{divisor}| \leq |\text{dividend}|\)

- Divide the absolute values of the numbers, then remember the quotient.
- What about the remainder?

If dividend is positive
- remainder \(\geq 0\)

If dividend is negative
- remainder \(\leq 0\)
Floating Point

Just superficial coverage

- Scientific notation: decimal to binary
  - "Normalized" - one digit to the left of the point
- Floating pt representation:

  sign-bit | exponent | significant

<table>
<thead>
<tr>
<th>IEEE</th>
<th>1</th>
<th>8</th>
<th>23</th>
</tr>
</thead>
</table>

32-bit floating pt number

| 64-bit | 1 | 11 | 82 |

- Leading 1 of significant is implicit
  - all numbers except 0 have it
- In order to represent negative exponents, a bias is used
  - bias is 127 in 32-bit numbers.
- Actual exponent = exponent-field - 127
- So, for $2^{-2}$, exponent field would be 129
- 0 is represented by all 0's in mantissa
  - there is no $1 \times 2^{-127}$ representable.
Note: Not using 2’s-complement in the exponent or for the significand allows comparison between floating pt numbers using the entire word (since sign is most significant, then exponent, then significand).

FP Addition (quick overview)

- Need the operands to have the same exponent.
  - If not, “de-normalize” the smaller number by shifting the significand to the right

- Add the significands (remember the implicit leading ones)

- Re-normalize *watch out for underflow or overflow*

FP Multiplication:

- Add the exponents
- Subtract 2’s bias (other wise summed)

- Multiply significands
- Re-normalize product

*Watch out for underflow or overflow*
The Processor - Data path & Control

Chaps. 5 in P+H

Clocking
- Clocking methodology
- Defines when times can be read or written
- Edge triggered is perhaps easiest to understand.

Remember this picture?

Need edge triggering to be sure that "old" value in the state element is the input to the combinational logic. Don't want "new value" to flow into state element and then into combinational logic.

That's why you want flip-flops rather than latches in a register.
Building a datapath

- i.e. mechanism for loading, storing, running instructions + data
- involves registers, memories, ALU, etc.

Portion for fetching instructions:
- PC holds address of instruction
- After fetch, increment PC by 4

Data path for "R-type Instructions"
- 2 register operands (source)
- 1 register (destination)
Data path for load/store instructions of the form:

1. \texttt{lw} $\text{reg}_1$, offset($\text{reg}_2$)
2. \texttt{sw} $\text{reg}_1$, offset($\text{reg}_2$)

Address is $\text{contents}(\text{reg}_2) + \text{offset}$

(can be negative)

Diagram:

- Instruction
- Read 1
- Read 2
- Write $\text{reg}$
- Write data
- Reg write
- Sign Extend
- 16
- 32
- Addr Memory
- Read Data
- Write Data
- Mem Read