Today

- Division (wrap up)
- Data and Control Path (intro)

Assignments

- HW3 out (due Thurs)
- Exam 1 on Thurs
Review: Binary Division

- “Long-hand” decimal division

```
   508 quotient

  _____
divisor 12 | 6102 dividend
    60
    --
    10
  102
  96
  ---
  6   remainder
```

- same process using binary numbers ...

```
  101 quotient (5)

  ______
divisor (4) 100 | 10110 dividend (22)
    100
    --
    11
    110
    100
    ---
    10   remainder (2)
```
Basic Algorithm: Assume subtracting 4-bit positive numbers

- Initialization:
  - represent quotient as 4-bit value 0000
  - represent remainder as 8-bit number, initialized to dividend
  - represent divisor as 8-bit number in “high” 4-bits (e.g., 0100 0000)

- Repeat $n + 1$ times for $n =$ # of quotient digits (e.g., 4 + 1 times)
  1. remainder = remainder $-$ divisor \[\ldots\] using 2’s complement
  2. shift quotient left by 1
     - if (remainder $\geqslant$ 0) then quotient = quotient $+$ 1
     - else remainder = remainder $+$ divisor \[\ldots\] restore remainder
  3. Shift divisor right by 1
Exercise: 0010 \text{ )} 0111

<table>
<thead>
<tr>
<th>Step</th>
<th>Remainder</th>
<th>Quotient</th>
<th>Divisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>0000 0111</td>
<td>0 \phantom{1}</td>
<td>0010 0000</td>
</tr>
<tr>
<td>(1)</td>
<td>0000 0111</td>
<td>00 \phantom{1}</td>
<td>0001 0000</td>
</tr>
<tr>
<td></td>
<td>1110 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1110 0111 (neg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0010 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 0111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>0000 0111</td>
<td>000 \phantom{1}</td>
<td>0000 1000</td>
</tr>
<tr>
<td></td>
<td>1111 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1111 0111 (neg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0001 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 0111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>0000 0111</td>
<td>000 \phantom{1}</td>
<td>0000 0100</td>
</tr>
<tr>
<td></td>
<td>1111 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1111 1111 (neg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 0111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td>0000 0111</td>
<td>0001 \phantom{1}</td>
<td>0000 0010</td>
</tr>
<tr>
<td></td>
<td>1111 1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 0011 (pos)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>0000 0011</td>
<td>0011 \phantom{1}</td>
<td>0000 0001</td>
</tr>
<tr>
<td></td>
<td>1111 1110</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 0001 (pos)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hardware support for division

- Square boxes are **registers** (Divisor, Remainder, Quotient)
- Another example of a “control” component
- The basic idea ... more efficient ways to implement in hardware
More CPU: Datapath and Control

Three general types of operations:

- Arithmetic-logical (ALU)
- Memory reference (e.g., store and load)
- Branching

Overview of components:

For each instruction type, first two steps identical:

1. Send program counter (PC) to memory to fetch instruction
2. Read registers using fields of instruction (e.g., addl $5 %eax)