Building an Arithmetic Machine
Today’s lecture

- MIPS assembly basics
- SPIM demo
MIPS is a register-to-register architecture: Arithmetic/logical state manipulations read from registers (or constants) and write to registers.
MIPS assembly instructions generally perform one manipulation at a time

<table>
<thead>
<tr>
<th>ADD</th>
<th>ADDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB</td>
<td>SUBI</td>
</tr>
<tr>
<td>AND</td>
<td>ANDI</td>
</tr>
<tr>
<td>OR</td>
<td>ORI</td>
</tr>
<tr>
<td>NOR</td>
<td>NORI</td>
</tr>
<tr>
<td>XOR</td>
<td>XORI</td>
</tr>
</tbody>
</table>
Complex operations require multiple assembly instructions


add $6, $1, $2   # $6 contains $1 + $2
sub $5, $3, $4   # Temporary value $5 = $3 - $4
mul $4, $6, $5   # $4 contains the final product

Use temporary registers as needed
  - Be careful reusing registers (e.g., $1 and $3)
What if we wanted to compute the following?

\[ 1 + 2 + 3 + 4 \]
To be continued elsewhere…

- We will write code some assembly code (add.s)
- Execute code in MIPS simulator called SPM
How can we write MIPS code to compute the following expression?

\[ z = 4 + x \times y - z; \]

- Assume the following register allocation:
  - $13 = x$, $20 = y$, $15 = z$
Negate a two’s complement represented number