Building an Arithmetic Machine

Second chance exam (verilog only)
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
Complex operations require multiple assembly instructions


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

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>addi $1, 1, 2</td>
<td>addi $1, $0, 1</td>
<td>addi $1, $0, 1</td>
</tr>
<tr>
<td></td>
<td>addi $2, 3, 4</td>
<td>addi $1, $1, 2</td>
<td>addi $2, $0, 2</td>
</tr>
<tr>
<td></td>
<td>add $1, $1, $2</td>
<td>addi $1, $1, 3</td>
<td>addi $3, $0, 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>addi $1, $1, 4</td>
<td>addi $4, $0, 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>add $1, $1, $2</td>
<td>add $1, $1, $2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>add $3, $3, $4</td>
<td>add $3, $3, $4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>add $1, $1, $3</td>
<td>add $1, $1, $3</td>
</tr>
</tbody>
</table>

A: none of the above  
B: I and II  
C: I and III  
D: II and III  
E: all of the above
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

Which two MIPS assembly instructions can be used to negate a two’s complement represented number?

a) not     b)  and    c)  nor    d)  xor
add i      add i    add i      add i

\[
\text{nor} (0, \$2, \$2)
\]