1. Write MIPS code for the following expression. Assume the following register allocation:

\[ m = \$12, \ n = \$5, \text{ and } p = \$9. \]

\[ m = n + (p \ join 0x7f); \]

2. What MIPS instruction is encoded by the following machine code?

\[ 0x00687826 = 32'b0000_0000_0110_1000_0111_1000_0010_0110 \]
3. Here’s some new Verilog syntax:

1. Replicate some wires:
   ```verilog
   ```

2. Concatenate some wires:
   ```verilog
   // notice that the order is MSB to LSB, from left to right
   ```

Use it to connect the buses as described in the Verilog code below:

```verilog
module thingy(B, A);
    output [7:0] B;
    input [3:0] A;

    assign B = {A[3], {3{A[2]}}, {2{A[1:0]}}};
endmodule // thingy
```

```
A[0] ________        ________ B[0]
         ________ B[4]
         ________ B[5]
         ________ B[6]
         ________ B[7]
```
For each of the following instructions, highlight the buses of the datapath being used and specify the values of the control signals and/or datapath values. *Recall that the ALU operations are encoded as: ADD=3'b010, SUB=3'b011, AND=3'b100, OR=3'b101, NOR=3'b110, and XOR=3'b111.*

```
xor $18, $9, $13
```

```
addi $7, $14, -1
```
Decoder

The outputs correspond to control signals on the datapath. The `except` signal should be 1 when the opcode/funct field pair is not recognized. `wr_enable` should be 0 in case of an unrecognized instruction.

<table>
<thead>
<tr>
<th>instruction</th>
<th>opcode</th>
<th>funct</th>
<th>alu_op</th>
<th>rd_src</th>
<th>alu_src2</th>
<th>wr_enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>addi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>andi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ori</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xori</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We’ll talk about converting this truth table to expressions in lab.