This exam is open book and open notes. Credit for problems requiring calculation will be given only if you show your work.

1. Floating Point

Translate the IEEE single-precision floating point numbers shown below to their decimal equivalent and write them in the space provided below.

**Value of number:**

1. $-1.101 \times 2^{(135-127)} = -1.625 \times 2^8 = -416$

2. $1.00111 \times 2^{(7-127)} = -1.109375 \times 2^{-120} = 8.346 \times 10^{-37}$

3. NaN (Not a Number)

4. $0.000111 \times 2^{-126} = 0.109375 \times 2^{-126} = 1.286 \times 10^{-39}$
2. Verilog / Logic & Arithmetic  

We would like to design a shifter module in Verilog with the inputs, outputs, and function described below. When DIR=0, it shifts the input left by the number of bits specified by AMT. When DIR=1, it shifts the input right by the number of bits specified by AMT. All shifts are logical shifts, i.e., the values that are “shifted in” are all zero.

![Shifter Diagram]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMT</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>SHIN</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>SHOUT</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>SHIN &lt;&lt; AMT</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>SHIN &gt;&gt; AMT</td>
<td>1</td>
</tr>
</tbody>
</table>

Fill in the Verilog module declaration below to show all input and output ports and the behavioral code necessary to perform the desired function (you may use either an assign statement or an always block).

```verilog
module Shifter(DIR, AMT, SHIN, SHOUT);
  input DIR;
  input [4:0] AMT;
  input [31:0] SHIN;
  output [31:0] SHOUT;

  reg [31:0] SHOUT;

  always @(DIR or AMT or SHIN)
    if (DIR) SHOUT = SHIN >> AMT;
    else SHOUT = SHIN << AMT;
endmodule
```

Alternative solution using assign

```verilog
module Shifter(DIR, AMT, SHIN, SHOUT);
  input DIR;
  input [4:0] AMT;
  input [31:0] SHIN;
  output [31:0] SHOUT;

  assign SHOUT = ( DIR ? SHIN >> AMT : SHIN << AMT );
endmodule
```
3. Single-Cycle Processor Design 30 Points

Modify the single-cycle processor design in the book to implement the shifting instruction “srl” (shift-right-logical) using the shifter designed in problem 2. The srl instruction is used in assembly language like this:

\[ \text{srl } \text{rd, rt, shamt} \]

The register transfer for this instruction is as follows:

\[ \text{Reg[rd] } \leftarrow \text{Reg[rt]} \gg \text{shamt}; \]

The srl instruction is encoded as an R-Type instruction, with \( \text{opcode}=0 \) and \( \text{funct}=2 \). Show all changes to the datapath along with any changes to the Control Unit or ALU Control modules in the space provided below.

There are two ways to approach the control for this problem:

1) Add a row to the truth table of the control unit (requires that Control Unit looks at both \( \text{opcode} \) and \( \text{funct} \)) – in this case, modify only the Control Unit table – ALU Control stays the same.

2) Modify the ALU Control but not the control unit (Use existing control unit signals for R-Type instructions)

Modified tables are shown on the next page.
### Approach 1 Control Unit (Must add `funct` input) – No Change to ALU Control

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RegDst</th>
<th>ALUSrc</th>
<th>MemtoReg</th>
<th>RegWrite</th>
<th>MemRead</th>
<th>MemWrite</th>
<th>Branch</th>
<th>ALUOp1</th>
<th>ALUOp0</th>
<th>ShDir</th>
<th>ShOrALU</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-format</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>beq</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>srl</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

### Approach 2 ALU Control – No Change to Control Unit

<table>
<thead>
<tr>
<th>ALUOp1</th>
<th>ALUOp0</th>
<th>F5</th>
<th>F4</th>
<th>F3</th>
<th>F2</th>
<th>F1</th>
<th>F0</th>
<th>Operation</th>
<th>ShDir</th>
<th>ShOrALU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>010</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>110</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>010</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>000</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>001</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>111</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>XXX</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### 3. Single-Cycle Processor Timing 15 Points

Assume that the shifter module has a delay of 3 ns.

(a) Calculate delay of the longest path in your modified circuit when performing the `srl` instruction.

- Mem Fetch: 2 ns
- Reg. Read: 1 ns
- Shift: 3 ns
- Reg. Write: 1 ns
- Total: 7 ns

(b) Calculate the longest path for all instructions in your modified circuit.

The longest path is still for the load word instruction.

- Mem Fetch: 2 ns
- Reg. Read: 1 ns
- ALU Operation: 2 ns
- Mem Read: 2 ns
- Reg. Write: 1 ns
- Total: 8 ns
4. Multicycle Processor Design (Datapath) 15 Points

We wish to modify the multicycle processor design to implement the \texttt{srl} instruction described in Problem 3. Show all changes on the diagram on the next page.