Problem 1 (6 points)

Draw a block diagram of the datapath unit of a circuit capable of executing the pseudocode given below.

In this code, MEM_LT and MEM_GE represent single-port memories of the size of 64 x 8. The input median represents a median of a set of 128 8-bit numbers fed to the circuit through the input din.

begin:
wait for s=1;
Done = 0

i=0
j=0
for k=0 to 127 do
if (din < median)
   MEM_LT[i] = din
   i++;
else
   MEM_GE[j] = din
   j++;
end if
end for

SUM_LT = 0
PROD_GE = 1
MIN_LT = MEM_LT[0] ;
MAX_GE = MEM_GE[0] ;

for n=1 to 63 do
   SUM_LT = SUM_LT + MEM_LT[n] ;
   if MEM_LT[n] < MIN_LT then
      MIN_LT = MEM_LT[n]
   end if
   PROD_GE = (PROD_GE * MEM_GE[n])/256 ;
   if MEM_GE[n] > MAX_GE then
      MAX_GE = MEM_GE[n]
   end if
end for

AVR_LT = SUM_LT/64.

done = 1
wait for s=0
go to begin
Please clearly mark the widths of all buses in your circuit.

Assume the following interface to your circuit:

<table>
<thead>
<tr>
<th>Port</th>
<th>Width</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>1</td>
<td>System clock</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>System reset – clears internal registers. Active high.</td>
</tr>
<tr>
<td>din</td>
<td>8</td>
<td>Input data bus</td>
</tr>
<tr>
<td>median</td>
<td>8</td>
<td>Median of a set of 128 8-bit numbers fed to the circuit through the input din</td>
</tr>
<tr>
<td>s</td>
<td>1</td>
<td>Operating mode: 0 = waiting for data, 1 = processing.</td>
</tr>
<tr>
<td>rd</td>
<td>1</td>
<td>Read enable. 0 = high impedance on the output bus dout, 1 = valid output dout</td>
</tr>
<tr>
<td>dout</td>
<td>8</td>
<td>One of the 4 results calculated by the circuit</td>
</tr>
<tr>
<td>sel_out</td>
<td>2</td>
<td>Selection between 4 calculated results: 00 = AVR_LT, 01 = MIN_LT, 10 = MAX GE, 11 = PROD GE.</td>
</tr>
<tr>
<td>done</td>
<td>1</td>
<td>Asserted when all results are ready, zero otherwise</td>
</tr>
</tbody>
</table>

Problem 2 (6 points)

1. Draw an interface to the circuit from Problem 1, with the division between the Datapath and the Controller.
2. Draw an ASM chart corresponding to the pseudocode from Problem 1.
3. Express all operations in your ASM chart in terms of active values of control signals generated as outputs of the Control unit and used as inputs in the Datapath.

Problem 3 (8 points)

Task 1 (4 pints)

Draw a detailed block diagram of the digital system including
1. the PicoBlaze core, KCPSM3
2. all external memories required for the basic operation of the PicoBlaze core
3. four input registers with the virtual addresses C5, D5, E5, and F5
4. four output registers with the virtual addresses C5, D5, E5, and F5
5. a D flip-flop with the output Q connected to the interrupt input of the PicoBlaze core, input SET connected to the external port INT, and input CLR connected to an appropriate output of the PicoBlaze core.

Assume that
- all input registers are identical to the output registers with the same addresses
- the input and output registers specified above are the only registers that the PicoBlaze core is communicating with
- your system needs to be able to allow the PicoBlaze core to write to all aforementioned output registers, and read from all aforementioned input registers using instructions OUTPUT and INPUT, respectively
- you need to provide all details of the address decoder, and build it out of basic logic components you are familiar with
- all registers and flip-flops have a reset input connected to the external port RESET.
Please clearly mark on your schematic:

- sizes of all memories and registers
- sizes and directions of all buses.

**Task 2 (4 points)**

Determine the contents of

1. internal registers s0-s4
2. flags C, Z, I, preserved C, preserved Z
3. all input/output registers
4. stack
5. PC

at the time of the execution of the instruction **RETURNI DISABLE**

before this instruction takes effect assuming that

- at the time = 0 a short pulse is generated at the input RESET
- at the time = 10 seconds, a short pulse is generated at the input INT
- the contents of the instruction memory is given by the following program:

```plaintext
CONSTANT BIT4, 10
CONSTANT BIT3, 08
CONSTANT BIT0_COMP, FE

ADDRESS 000
LOAD s0, FE
LOAD s1, C5
LOAD s2, 04

INIT:
OUTPUT s0, (s1)
ADD s0, 02
ADD s1, 10
SUB s2, 01
JUMP NZ, INIT
ENABLE INTERRUPT

LOOP:
JUMP LOOP

ADDRESS 100

ISR:
INPUT s3, F5
LOAD s4, CD
OR s4, BIT4
AND s4, BIT0_COMP
XOR s4, BIT3
SLA s4
TEST s3, s4
RETURNI DISABLE

ADDRESS 3FF
JUMP ISR
```