1. (10 points) Define the term *combinational circuit*.
   A *circuit in which the output signals depend only on the current input signals.*

Let $x$ be a signal and $p$ be a purely asynchronous process within a VHDL architecture. Under what conditions is it necessary for $x$ to be included in the sensitivity list for $p$?

* $x$ must be included in the sensitivity list if it is used within $p$. That is, it appears on the right side of an assignment or in an if-statement or cast-statement condition, etc.

Write a short VHDL process involving `std_logic` signals $a$, $b$ and $c$ that will cause a circuit synthesizer to generate a latch.

```vhdl
process (a,b) is begin
  if a = b then c <= '0';
  elif a > b then c <= '1';
end if;
end process
```

How does a flip flop behave when it is metastable?

*The output oscillates between low and high, or remains stuck in between low and high.*

What is the maximum time period for which a flip flop can be metastable?

*There is no maximum time.*
2. (10 points). Use the Karnaugh map below to find a minimum sum-of-products expression for $\Sigma_m(0,1,3,4,5,8,9,12,14)$. How many simple gates of each type are needed to implement this expression (without further simplification)? How many LUT4s?

![Karnaugh Map 1](image1)

The sum-of-products expression is $A'C' + B'C' + A'B'D + ABD'$

6 AND gates, 3 OR gates and 4 inverters. 1 LUT.

Use the Karnaugh map below to find a minimum product-of-sums expression for $\Sigma_m(0,1,9,13,15), \Sigma_d(3,4,5,6,8)$. Make full use of the don’t cares.

![Karnaugh Map 2](image2)

\[ F' = AD' + A'C + B'C \]
\[ F = (A'+D)(A+C')(B+C') \]
3. (10 points) The diagram below shows a generic state machine. Assume that the state is encoded using 5 bits.

Does this circuit violate the hold time constraint?

\[ \text{hold time} + \text{clock skew} = 1.5 \text{ ns} > 1\text{ ns} = \text{min ff delay} + \text{min combinational delay}, \text{ so hold time violations} \]

If so, how could you eliminate the violations? If not, what is the smallest clock skew for which hold time violations are possible.

*Add delay to the feedback path from the outputs of the flip flops.*

What is the smallest safe clock period for this circuit (without any modifications to handle hold time violations)?

\[ \text{max ff delay} + \text{max combinational delay} + \text{setup} + \text{skew} = 4 + 5 + 1 + 1 = 11 \text{ ns} \]

If the clock goes from low to high at time \( t_0 \), during what time period must the inputs be stable?

*From \( t_0 - 4 \text{ ns} \) to \( t_0 + 0.5 - 1 = t_0 - 0.5 \text{ ns} \).*

Suppose that two copies of this circuit are instantiated together, with the output of one connected to the input of the other. For this combined circuit, what is the minimum safe clock period?

\[ \text{max ff delay} + \text{max combinational delay} + \text{setup} + \text{skew} = 4 + (4 + 3) + 1 + 1 = 13 \text{ ns} \]
4. (10 points) The processor simulation below includes several labeled blanks. Fill in the correct values in the spaces below.

A. _______ 80f1 _______  
B. _______ 002c _______  
C. _______ 00f1 _______  
D. _______ 0004 _______  
E. _______ 0004 _______

The instruction set for the WASHU-2 processor appears below.

<table>
<thead>
<tr>
<th>Code</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>halt</td>
<td>1xxx constant load</td>
</tr>
<tr>
<td>0001</td>
<td>negate</td>
<td>2xxx direct load</td>
</tr>
<tr>
<td>01xx</td>
<td>branch</td>
<td>3xxx indirect load</td>
</tr>
<tr>
<td>02xx</td>
<td>branch if zero</td>
<td>5xxx direct store</td>
</tr>
<tr>
<td>03xx</td>
<td>branch if positive</td>
<td>6xxx indirect store</td>
</tr>
<tr>
<td>04xx</td>
<td>branch if negative</td>
<td>8xxx add</td>
</tr>
<tr>
<td>05xx</td>
<td>indirect branch</td>
<td>cxxx and</td>
</tr>
</tbody>
</table>

- 4 -
5. (15 points) Write a program in the WASHU-2 assembly language that implements the following C-style pseudo-code. Note that the lines defining $i$, $j$ and $p$ are included below. For each of the branch instructions in your program, also show the actual machine instruction that is generated by the assembler for that instruction.

```c
while (i != 0) {
    if (j > i) *p = *p + j;
    i--;
}
```

```
loop:      location 0100
    dLoad i
    brZero end 0208
    negate
    add j
    brPos 2 0302
    branch skip 0104
    iLoad p
    add j
    iStore j
skip:      cLoad -1
    add i
    branch loop 01f5
```

```
i:  20
j:  7
p: 0123
```
6. (10 points) Consider an SRAM with 2K words of 32 bits each. How many address bits are needed to address these words? (Reminder $1024=2^{10}$, $4096=2^{12}$, $16,384=2^{14}$, $65,536=2^{16}$.)

11

Assuming that the central memory array has the same number of rows as it has columns, how many rows are there?

256

How many words are stored in each row?

8

How many of the address bits are used by the row decoder?

8

How many are used by the column decoder?

3

Draw a diagram of the basic storage cell that is typically used in an SRAM.
7. (10 points) Consider a version of the WASHU-2 processor that is equipped with a 2-way set-associative cache with 256 rows (so, 512 words altogether) for both instructions and data, with the partial contents shown below. The least recently used (LRU) bit in the center column is equal to 0 if the left-hand entry has been least recently used and is equal to 1 if the right-hand entry has been least recently used.

<table>
<thead>
<tr>
<th>tag</th>
<th>value</th>
<th>LRU</th>
<th>tag</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>22</td>
<td>0</td>
<td>82</td>
<td>0012</td>
</tr>
<tr>
<td>4a</td>
<td>34</td>
<td>0</td>
<td>24</td>
<td>8249</td>
</tr>
<tr>
<td>4b</td>
<td>23</td>
<td>0</td>
<td>24</td>
<td>2021</td>
</tr>
<tr>
<td>4c</td>
<td>20</td>
<td>0</td>
<td>50</td>
<td>0202</td>
</tr>
</tbody>
</table>

If PC=244a, what instruction is fetched next?

8249

Assume ACC=5 initially. What value is in the ACC after the next instruction executes?

ACC=5+3=8

Show how the fetching and execution of this instruction modifies the cache by marking the changes in the table above.

Suppose that the instruction at location 244b is 504c. Show how the fetching and execution of this instruction changes the cache contents, by marking your changes on the diagram above (be sure to update the LRU bits, if appropriate).

What value is in the ACC after this instruction executes?

Since this is a direct store instruction, the ACC does not change. So, it is still 8.
8. (10 points) On a modern processor used in a laptop, approximately how long does it take to retrieve an instruction from the L1 instruction cache? How long does it take to retrieve an instruction from main memory, if it’s not present in any of the caches?

*It takes about 1 ns to retrieve an instruction from L1 cache and about 100 ns to retrieve it from main memory.*

What is the advantage of having multiple general purpose registers in processor (as opposed to the single accumulator used in the WASHU-2)?

*It allows programs to avoid many of the load and store operations that would otherwise be required. When a subprogram is entered, registers can be assigned to each parameter and local variable, allowing most uses of those parameters and variables to be done using register-to-register instructions that do not require expensive loads and stores.*

List two instructions that could be added to the WASHU-2 instruction set that would improve the performance of typical programs. For each instruction give an instruction summary similar to the one shown below for the brZero instruction. Your new instructions should be natural additions to the existing instruction set. In particular, do not re-use any instruction codes already assigned.

```
02xx    brZero    if ACC=0 then PC=PC+ssxx
9xxx    multiply   ACC = ACC * M[pxxx]
axxx    divide     ACC = ACC/M[pxxx]
```

Explain how these instructions improve the performance of programs running on the processor. Estimate the amount by which they would improve the performance of programs using them.

*In programs that do integer arithmetic, these instructions can eliminate the need to call special subprograms. For example, the multiply subprogram used in labs requires over 100 instructions, taking an average of about 100 ns each. So that’s 10 microseconds per multiply. A multiply instruction could perform this operation in about 100 ns. So for each multiply instruction in a program, the improvement is a factor of more than 100. Of course, the overall impact on the program will be smaller, since most instructions are not multiplications.*
9. (10 points) Consider the VHDL process shown below. Assume that all signals are 16 bits.

```vhdl
[1] process (clk, a, b, c) begin
[2] x <= a + b;
[3] if rising_edge(clk) then
[4] if a = c then
[5] y <= b + c; z <= c;
[6] elsif b > c then
[7] y <= a - x;
[8] end if; end if; end process;
```

Suppose that a circuit is synthesized for this process for the FPGA on our prototype board. How many flip flops are used by this circuit? How many latches?

32 flip flops, no latches

Approximately how many LUT4s are used to evaluate the condition in the if-statement on line 4?

*This is an equality comparison on 16 bit values, so about n/2=8 LUTs.*

Approximately how many LUT4s are used to evaluate the condition in the elsif-statement on line 6?

*This is a greater-than comparison on 16 bit signals, so about 16 LUTs.*

Approximately how many LUT4s are used by all the assignments to y?

*There are two possible values assigned to y, b+c and a-x. Since the operands are all different, we’ll require a separate adder and subtracter, each requiring about 16 LUTs. In addition, we’ll need a mux to select one of these two values, adding another 16 LUTs. So, about 48 LUTs, altogether.*