1. [5 points] Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

(A) Any algorithm can be implemented in any programming language.  
   a. This statement is true.  
   b. This statement is false.  
   c. It depends on the underlying hardware.  
   d. It depends on the underlying OS.

(B) If we want to design a computer system, what is the correct order?  
   a. develop the ISA first, then the control unit, then the datapath  
   b. develop the datapath, then the ISA, then the control unit  
   c. develop the control unit, then the ISA, then the datapath  
   d. develop the ISA first, then the datapath, then the control unit

(C) For the data path given in question 3, which of the following statements is a valid microinstruction?  
   a. IR → MDR   b. PC → AOR   c. MDR → MAR  
   d. SE(imm) → AOR   e. MAR → MDR

(D) If we want to add an instruction from the instruction set of a processor, which of the following MUST be updated?  
   a. control unit  b. datapath  c. the compiler and assembler  
   d. (a and b)  e. (b and c)  f. (a and c)

(E) The main difference between MIPS ISA and x86 ISA is:  
   a. MIPS ISA CISC while x86 is RISC  
   b. MIPS ISA is RISC while x86 is CISC  
   c. x86 uses way more registers  
   d. x86 has less instructions
2. Suppose we have the following Boolean function:
   \[ F = A'BC' + AB + B'C \]
   a. [2 pts] Draw the truth table of this function

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

   b. [1 pt] Draw the logic diagram

3. For the datapath in next page:

   a. [1 pt] What problems may arise if we remove AOR and connect ALU directly to the bus?
      - Data cannot be buffered and may be lost after ALU generates it.
      - We will have to slow ALU down till destination gets the output of ALU

   b. [1 pts] Is it good or bad to add another AIR as the second input to ALU (instead of getting the second input directly from the bus)? Justify your answer.
      - Bad
      - We will need an extra cycle to fill this extra AIR
c. [2 pts] Write the microinstructions required to implement: \textit{sw Rd, 5(Rt)} (That is, write the content of Rd to memory location \(Rt+5\)). No need to write the fetch phase.

\begin{align*}
\text{SE(imm)} & \rightarrow \text{AIR} \\
\text{R}[Rt] + \text{AIR} & \rightarrow \text{AOR} \\
\text{AOR} & \rightarrow \text{MAR} \\
\text{R}[Rd] & \rightarrow \text{MDR} \\
\text{MDR} & \rightarrow \text{M}[\text{MAR}] \\
\end{align*}

4. a. [2 pts] Write an algorithm that, given an array of \(n\) elements, determine whether there are any repeated elements. The algorithm returns TRUE if there is at least one repeated element, or FALSE otherwise.

\begin{equation*}
\text{for}(\ i = 0, \ i < \text{length}(\text{Array A}) - 1; \ i++) \\
\quad \text{for}(\ j = i+1; \ j < n; \ j++) \\
\quad \quad \text{if}(\text{A}[i] == \text{a}[j]) \\
\quad \quad \quad \text{return TRUE} \\
\text{Return FALSE}
\end{equation*}

b. [1 pt] What is the complexity of the worst case scenario (i.e. the Big-Oh notation)?

\(O(n^2)\)

c. [1 pt] Will it help if the array is sorted? Justify.
Yes, because the second loop will need to stop after the first un-equal element.
5. For the following two functions (assume \( n \geq 0 \)):

```c
int fact(int n)
{
    int i;
    int result = 1;
    for (i = n; i > 0; i--)
        result = result * i;
    return result;
}
```

```c
int fact_u2(int n)
{
    int i;
    int result = 1;
    for (i = n; i > 0; i-=2)
        result = (result * i) * (i-1);
    return result;
}
```

(a) [1 pt] What does the function `fact` do? (Do not explain the program to me, just 2-3 words are more than enough!)

\( n! \)

(b) [1 pt] For which values of \( n \) do `fact` and `fact_u2` produce similar results?

even values of \( n \)

(c) [2 pts] What is the minimum modification that we can do to `fact_u2` such that it always produces same result as `fact`?

make the for-loop: `for( i = n; i>1; i-=2)`