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 ISA.
   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) The algorithm of calculating the prime numbers between 1 and one million is:
   a. unsolvable    b. intractable
   c. undecidable   d. none of them

(C) The number of registers that we see in an ISA (e.g. 16 in x86 and 32 in MIPS) is exactly the same number of registers that we see in the datapath.
   a. This statement is true.   b. This statement is false.
   c. We cannot tell because it depends on each situation.

(D) In the same program, the size of a pointer to an array of 10 integers is bigger than the size of a pointer to a float.
   a. This statement is true.   b. This statement is false.
   c. Cannot tell from the information given.

(E) We need to use heuristics if the algorithm of the problem at hand is:
   a. unsolvable   b. intractable   c. undecidable
2. [2 pts] Using as many 2 x 1 multiplexers as you want and as many gates as you want, design a 4x1 multiplexer. Just draw it and use muxes as black boxes.

A 4x1 multiplexer has 4 inputs, 2 control lines, and one output.

3. Suppose we want to design a logic circuit that outputs 1 if there is an overflow after an addition and 0 otherwise for signed numbers. The inputs to that circuit are: the sign bit of operand 1, the sign bit of operand 2, and the sign bit of the result.

(a) [8 points] Fill-in the Truth Table (3 inputs: A, B, and C; and 1 output F)

<table>
<thead>
<tr>
<th>Sign op 1: A</th>
<th>Sign op 2: B</th>
<th>Sign result: C</th>
<th>Overflow?: 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>0</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>0</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>0</td>
</tr>
</tbody>
</table>

(b) [1 point] Write the Boolean equation from the truth table

F = A’B’C + ABC’

(c)[2 points] Draw the logic circuit
4. [5 point] Suppose you have a string name[50] that contains a specific sentence. Write a function output() that prints this string 10 times without using any loops or goto statements. No need to write a full program, just the function, showing its arguments and the implementation of that function. Assume name[50] has already been filled up with the correct string and is declared globally. Also show how this function will be called from main().

void output(int count){
    if(count < 10){
        printf("%s\n", name);
        output(count+1);
    }
}

The function will be called from main() like this:  output(0);
5. [3 points] Write x86_64 assembly instructions required to multiply the value in %rax by 67 without using the multiplication operation. The value in rax is long.

leaq (%rax, %rax, 2), %rbx  # now rbx = 3*rax
salq  %rax, $6  # rax = 64rax
addq %rbx, %rax  # r64rax + 3rax

6. [1 point] Why is sign-extension needed in the datapath? (Note: You will lose points if you give me the definition of sign-extension! The question did not ask for definitions).

Because in some operations the two operands are not of the same length. The ALU requires the two operands to be of the same length.

7. Given the following C code (hint: if x = 13 then x/=10 will make x = 1):

```
int sum(int x) {
    int s;
    for(s=0; x>0; s+=x%10, x/=10);
    return s;
}
```

a. [2 point] What will the above function return if x = 43?

7

b. [3 points] What does this function do? Do NOT explain the code but explain what the function does in plain English. No credit will be given if you tell me “This function executions a for-loop that …”.

It calculates the sum of the digits of a number. So 43 → 4+3 = 7
8. Given the following piece of code:

```c
void process(int *x){
    int y[10];
    x = (int *)malloc(10*sizeof(int));
}
```

a. [1 point] How many bytes does the array of y[] consume?

40

b. [1 point] How many bytes does the array allocated by malloc consume?

40

c. [1 point] Where is the array y[] stored?

Stack

d. [1 point] Where is the array pointed to by x[] stored?

Heap

e. [2 points] What happens to array x[] and y[] when the function process() returns?

Array Y[] is gone and array X[] remains in the heap.

f. [2 points] Write the C statement that makes pointer x points to the array y[].

```c
x = y;
```
or

```c
x = &y[0];
```