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) Every call to malloc() requires OS intervention:
   a. True       b. False      c. depends on how much physical memory we have

(B) Whenever there is a cache miss, a replacement policy must be used.
   a. The above statement is true.
   b. The above statement is false.
   c. It depends on the design of the cache.

(C) What is the correct order for memory access
   a. TLB-page table- cache   b. cache-page table -TLB
   c. page table- cache - TLB  d. page table - TLB - cache

(D) Assume a signed int x, what does the following expression present:
   \((1 + (x<<2) + \sim x)\)
   a. 3x       b. 4x       c. 5x       d. 6x       e. none of them

(E) Assume the register rax stores value x. What will be the content of rax after executing the instruction: \texttt{leaq 0(%rax, %rax, 4), %rax}
   a. 5x       b. 6x       c. 7x       d. depends on what is stored in memory
2. [12 points] Given the following x86_64 assembly code and its corresponding C code, fill in the blanks in the corresponding C code. Also on the far right, fill in the correspondence between each register and its corresponding variable in C. (Hint: you can neglect edx because it does not correspond to any variable in the C code and is used only by the compiler for the translation).

```assembly
do: movl $1, %eax
    movl $16, %ebx
    xorl %ecx, %ecx
L0: cmpl %ebx, %ecx
    jge L1
    cmpl %edi, %eax
    jle L2
    subl %esi, %eax
    jmp L3
L2: movl %edi, %edx
    addl %esi, %edx
    addl %edx, %eax
L3: addl $3, %ecx
    jmp L0
L1: Ret
```

```c
int do(int x, int y){
    int j = __16____;
    int data = __1____;
    int i;

    i = ___0_______;

    while(__j > i____){
        if( data > x)
            data -= y____;
        else
            data += x + y____;

        i += 3;
    }

    return data;
}
```

<table>
<thead>
<tr>
<th>Register</th>
<th>Corresponding Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>edi</td>
<td>x</td>
</tr>
<tr>
<td>esi</td>
<td>y</td>
</tr>
<tr>
<td>ebx</td>
<td>j</td>
</tr>
<tr>
<td>eax</td>
<td>data</td>
</tr>
<tr>
<td>ecx</td>
<td>i</td>
</tr>
</tbody>
</table>

3. [1 point] From the assembly code above, in one sentence explain why we used the l version of instructions (e.g. movl) instead of the q version (like movq)?

Because the variables in the C code are all integers, hence, 4 bytes in length.

4. [3 points] Write three different ways to put 0 in register rax. Each one must be one assembly instruction only.

- movl $0, %rax
- xorl %rax, %rax
- subl %rax, %rax
5. For the following piece of code:

1. void par() {
2.     if (fork() == 0) {
3.         fork();
4.         printf("hello\n");
5.     }
6.     return;
7. }
8. 
9. int main() {
10.    printf("hello\n");
11.    printf("hello\n");
12.    par();
13.    exit(0);
14. }

a. [1 point] How many times will “Hello” be printed?

4 times

b. [1 point] Before executing line 13 by any process; how many page tables exist (assuming the system uses 1-level page table)?

3 page tables, one per process.

c. [2 points] If instead of return at the end of par() function we use exit(0). Will your answer on question (a) above differ? If yes, what is the new answer? If no, explain why not.

No, the answer will be the same because there are no printf to be executed by any process after that point.
6. Suppose that we have a system with main memory access time of 200 cycles. We added to that system a cache with 20 cycles access time. The resulting average memory access time becomes 40.

   a. [1 points] What is the cache miss rate?

   avg mem = m + (1-p) M   Filling the blanks: m = 20, M = 200, and avg mem = 40
   (1-p) = miss rate = 20/200 = 1/10 = 10%

   b. [1 points] Did we benefit from having a cache in that case? Why?

   Yes, because without cache any memory access will be 200 cycles.

   c. [1 point] Is the cache mentioned above accessed with virtual address? or physical address?

   Physical address.

   d. [2 points] Suppose the above cache has block size of 64 bytes, associativity of 8, and 32 sets. What is the total size of the cache in bytes (you can leave it as power of two if you want).

   Total cache size = number of sets * number of blocks per set * block size
   = number of sets * associativity * block size
   = 32 * 8 * 64
   = 2^5 * 2^3 * 2^6 = 2^{14} = 16384 = 16 KB