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) The range of numbers that can be present by signed int and unsigned int are the same.
   a. True   b. False   c. depends operating systems

(B) We can have two caches of the same size but different associativity
   a. The above statement is true.
   b. The above statement is false.
   c. It depends on whether we have TLB or not.

(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) The dynamic memory allocator deals with virtual addresses only:
   a. True   b. False   e. Depends on the size of the cache

(E) Assume the register rax stores value x. What will be the content of rax after executing the instruction: leaq 0(%rax, %rax, 2), %rax
   a. 3x   b. 4x   c. 5x   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 $2, %eax
    movl $32, %ebx
    xorl %ecx, %ecx
L0: cmpl %ebx, %ecx
    jg L1
    cmpl %edi, %eax
    jle L2
    addl %esi, %eax
    jmp L3
L2: movl %edi, %edx
    addl %esi, %edx
    addl %edx, %eax
L3: addl $2, %ecx
    jmp L0
L1: Ret
```

```c
int do(int x, int y){
    int j = __32___;
    int data = __2____;
    int i;
    i = __0________;
    while(____i <= j________){
        if( data <= x)
            data += x+y____;
        else
            data += y____;
        ___i += 2_______;
    }
    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 e version of registers (e.g. eax, ebx, …) instead of the q version (like rax, rbx, …)?

Because all the variables are integers, which are 4 bytes (32 bits), and hence the e version.

4. [2 points] Write two different instructions to multiply register rax by 4 without using multiplication instructions. Each one must be one assembly instruction only. Assume rax contains unsigned long

- leal 0(, %rax, 4), %rax
- shl $2, %rax
5. For the following piece of code:

```c
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.        par();
12.        printf("hello\n");
13.        exit(0);
14.     }
```

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

6 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. [3 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.

Yes, it will be different because with exit(0) the child process (and its child too) won’t execute the last printf in the main function. In which case, “Hello” will be printed 4 times only.
6. Suppose that we have a system with main memory access time of 100 cycles. We added to that system a cache with 50 cycles access time. The resulting average memory access time becomes 150.

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

No, because without cache each memory access would have been 100 cycles instead of 150.

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

avg mem access = m + (1-p)M  
Filling the blanks: m = 50  avg mem access = 150  M = 100  
This makes (1-p) = 1 = 100% miss rate.

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

Physical

d. [2 points] Suppose the above cache has 6 bits for offset, 4 bits for set, and 22 bits for TAG. What is the total cache size in bytes assuming the cache is 2-way set associative? You can leave the size as power of two to make it easier. To get full-credit, show all the steps.

#bits on offset gives us block size: 6 bits → block size = 2^6  
#bits in set gives us the total number of sets: 4 bits → we have 2^4 sets  
Total cache size = #sets * #blocks per set * block size  
= #sets * associativity * block size  
= 2^4 * 2 * 2^6 = 2^{11} \text{ bytes}