1. For the following questions circle True or False

The operating system has the power to suspend your program when it is running.	TRUE	FALSE
The only reason the operating system is “special” is because it is the first program loaded.	TRUE	FALSE
In a time sharing system multiple process take turns running on the CPU.	TRUE	FALSE
Caches would work well for a program that randomly accesses locations in memory.	TRUE	FALSE
An an **Interrupt Drive I/O** system the **hardware** is responsible for doing all of the work necessary to service an interrupt.	TRUE	FALSE
An a **I/O Polling** system the **software** is responsible for doing all of the work necessary to service I/O.	TRUE	FALSE
Intel is a little endian machine	TRUE	FALSE

2. On a **little** endian machine the **least** significant byte of a **word** appears at the lowest address but on a big endian machine the **most** significant byte of a **word** appears at the lowest address.

3. Explain the difference between I/O mapped I/O and Memory Mapped I/O. Which does Intel use?
   In I/O mapped memory we partition the I/O and memory address space into two separate spaces. The hardware has to support addintional instructions so as to disambiguate wether we are writing to an I/O device or memory. Intel is uses I/O mapped approach and used IN and OUT for I/O but MOV for memory. In memory mapped I/O there is 1 unified address space for both memory and I/O devices but we reserve certain addresses for I/O devices. The remaining addresses are used for memory

4. Convert 0x17ADB to binary. : **1 0111 1010 1101 1011**

5. Convert 101 1101 1011 1010 to hex.: **5DBA**

6. Convert 7456_8 to base 5.: **111021**

7. Write down the IEEE floating point format for the following number 63.25.

   111111.01 * 2^0
   1.111101 * 2^5
   exponent = 127 + 5 = 132
   mantissa = 111101 followed by 17 zeros

Sign	Exponent	Mantissa
0	'10000100'	'11110100000000000000000'

8. Assume that the label x contains the value 500. The following table shows the current contents of memory

Address	500	501	502	503	504	505	506	507
Value	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7

   After executing this instruction: `movl $0x12A96B7C, x.` Show what the contents of memory will be

Address	500	501	502	503	504	505	506	507
Value	**0x7C**	**0x6B**	**0xA9**	**0x12**	0x4	0x5	0x6	0x7

9. What is wrong with the following assembly code? Write code that correctly implements the user's intent. You may assume that all registers besides EAX are free to use.

```
movl x, (%eax)
```

2 operands are in memory, but instructions only support 1

```
1. movl x, %ebx
2. movl %ebx, (%eax)
```

10. Given that array is defined as `int array[100][50][25]` translate the following line of C to assembly code: `X = Array[I][J][K]`. Assume the following
    Array is stored in ESI
    X is EDI
    I is stored in EAX
    J is stored in EBX
    K is stored in ECX
    That all registers besides the ones mentioned are available to use.
    That I, J, K, and Array are not need after this line completes.

    #first calculate I displacement
    Imull $50
    Imull $25

    movl %eax, %edi

    #then calculate J displacement
    movl %ebx, %eax
    Imull $25
    addl %eax, %edi

    #then calculate K displacement
    addl %ecx, %edi

    #finally assign X the value
    movl (%esi, %edi, 4), %edi

11. Given that array is defined as `int** array` translate the following line of C to assembly code: `X = Array[I][J][K]`. Assume the following
    Array is stored in ESI
    X is EDI
    I is stored in EAX
    J is stored in EBX
    K is stored in ECX
    That all registers besides the ones mentioned are available to use.
    That I, J, K, and Array are not need after this line completes.

    movl (%esi, %eax, 4), %esi
    movl (%esi, %ebx, 4), %esi
    movl (%esi, %ecx, 4), %edi

12. Complete the following function, `illegal`. This function returns the ith argument of the nth stack frame located above this current frame. Arguments are numbered left to right from 0 to number of arguments - 1. If we consider `illegal`, `i` would be argument 0 and `n` would be argument 1. If n is 0 then we are considering illegal's stack frame. If n is 1 then the frame of the function that called illegal and so on.

```
int illegal(int i, int n){
  //accesses the ith argument of the nth stack frame

  int result;

  __asm__(
//ecx will be used to keep track of how many frames we have
//passed through

  "movl $0, %%ecx;"

//esi will contain the pointer to the current stack frame we are on
  "movl %%ebp, %%esi;"

  //first locate correct stack frame
  "locate_frame:;"
  "cmpl %[n], %%ecx;"
  "jge get_arg;"
  "movl (%%esi), %%esi;"
  "incl %%ecx;"
  "jmp locate_frame;"

  //now that we are are at the correct frame access the variable
  "get_arg:;"
  "movl 8(%%esi, %[i], 4), %[result];"

  :
  [result] "=r" (result) :
  [i] "r" (i), [n] "r" (n) :

  "%ecx", "%esi", "cc");


  return result;
}
```

13.
14. Question showing stack ask what is and isn't part of stack and stack frame.
15. Explain what direct memory access is. What is the purpose of direct memory access?