Questions Lab 2

1. Assuming that the following JOS kernel code is correct, what type should variable x have, uintptr_t or physaddr_t?

```
        mystery_t x;
char* value = return_a_pointer();
*value = 10;
x = (mystery_t) value;
```

Uintptr_t because x is holding a pointer(VA)

2. What entries (rows) in the page directory have been filled in at this point? What addresses do they map and where do they point? In other words, fill out this table as much as possible:

Entry	**Base Virtual Address**	**Points to (logically):**
1023	**?**	**Page table for top 4MB of phys memory**
1022	**?**	**?**
.	**?**	**?**
.	**?**	**?**
.	**?**	**?**
2	**0x00800000**	**?**
1	**0x00400000**	**?**
0	**0x00000000**	**[see next question]**

3. We have placed the kernel and user environment in the same address space. Why will user programs not be able to read or write the kernel's memory? What specific mechanisms protect the kernel memory?

When in user mode only other user mode pages will be readable. When in sudo mode, all pages are accessible. This is because of privilege levels preventing the user from accessing high level things.

4. What is the maximum amount of physical memory that this operating system can support? Why?

4 GB. THe maximum number of bytes we can access with 32 bits.

5. How much space overhead is there for managing memory, if we actually had the maximum amount of physical memory? How is this overhead broken down?

   4MB for pageInfo with each struct page info being 8 bytes. 4MB for pagetables because each is 4096 bytes and we can reference 1024 pages. 4KB for one page directory.

   8.00400MB

6. Revisit the page table setup in kern/entry.S and kern/entrypgdir.c. Immediately after we turn on paging, EIP is still a low number (a little over 1MB). At what point do we transition to running at an EIP above KERNBASE? What makes it possible for us to continue executing at a low EIP between when we enable paging and when we begin running at an EIP above KERNBASE? Why is this transition necessary?

We transition to an EIP above KERNBASE when we go to the relocated tag. We can do still execute at a low EIP between when we enable paging and when we start running at a high EIP because entry.pgdir.c maps both virtual addresses 0-4MB and virtual addresses KERNBASE-KERNBASE+mb to the same physical address. This is necessary because the rest of the kernel is at high addresses and we wouldnt be able to run them otherwise.