Homework 6: Registers, segmentation and paging

Answer the following questions.

1. Consider the following program. `section-1` and `section-2` of the `foo` routine don't access stack. The compiler wants to use all registers (including `%esp`) in `section-1` and `section-2`. In the rest of the code (apart from `section-1` and `section-2`), the compiler is only accessing the `caller-saved` registers and stack. Discuss which registers need to be saved and restored. Which parts of the code (e.g., before `section-1`), the registers can be saved and restored by the compiler? How will the assembly code corresponding to saving and restoring of registers look like? You can assume that the program is single-threaded. [3]

```
void foo() {
some code here
/* section-1 */
some code here
call bar
some code here
/* section-2 */
some code here
}
```

2. What will be your solution to the previous problem for a multi-threaded application? [1]

3. Consider the case when we are using the segmentation for memory isolation (paging is disabled). The `user_to_kernel` routine takes a user's buffer and the size of buffer as input and returns the kernel virtual address corresponding to the user's address. The current GDT is shown below. What are the checks `user_to_kernel` will perform (write absolute values in the checks)? What will be the return value of `user_to_kernel`? The user's buffer address is `20000`. The size of the buffer is `1024`. [1]

idx	base	limit	DPL
0	1000	0xFFFFFFFF	0
1	30000	21024	3

4. Consider the two-dimensional page table discussed in the class. Let us say the virtual address is `0x30710011`. During the address translation: [1]

- What will be the index in the page-directory?
- What will be the index in the page-table?

How to submit

Submit your handwritten homework in the submission box placed at the old academic building (2nd floor). The box will be placed on days when the homework is due.