CS220 - Computer System II
Assignment 10 (100 points)

Due: 11/11/2016, 11:59pm

1 Question 1 (15 * 3 = 45 points)

Implement the following macros in a file called macros.h. This is along the lines of what you did in Lab 10. If you need to implement helper macros, you are free to do so.

1. Macro TEST_IF_ANY_SET(v, start, end) that tests if *any* of the bits between start and end (inclusive) in vector v is set.

2. Macro TEST_IF_ALL_SET(v, start, end) that tests if *all* of the bits between start and end (inclusive) in vector are set.

3. Macro COUNT_NUM_SET(v, start, end) that counts the number of set bits between start and end (inclusive) in the vector.

2 Question 2 (50 points)

You will implement a function `void print_backtrace(int count)` in file bt.c. This function will print no more than `count` number of return addresses in preceding calling functions or up to main, whichever smaller. The call trace should be similar to what is obtained when we type `bt` on gdb, EXCEPT that this function will not print the names of the functions in the trace, just the return addresses in the callee functions. You are free to implement helper functions. You are guaranteed that all functions in the program will use frame pointer.

Hint: A backtrace is nothing but the sequence of return addresses in the preceding stack frames. A couple of points to note:

1. The return address (in the calling function) is stored on the stack.

2. On entry to each function, the old frame pointer (EBP register) is pushed on to the stack (its position is right after the return address).

3. In any given function, the return address is always stored at [ebp+4] and ebp of the calling frame is always stored at [ebp].

Strategy:

- As a first step, find the bounds of main function (you may do this in a separate function). Start from address of main, and start reading bytes till you hit the return instruction (you have already done this in an earlier assignment). Note the address of the return instruction. The start of main function and the address of the return instruction form the bounds of main function.
- The goal now is to print `count` number of return addresses in the preceding frames, or till you hit a return address that is inside main function. So:

```
curr_ebp <- get current ebp
while count > 0:
        ret_addr <- get current return address which is in [curr_ebp + 4]
        print ret_addr
        if ret_addr is an address in main:
            return
        curr_ebp <- get ebp for previous frame, which is in [curr_ebp]
        decrement count
```

- So, how would you get the current ebp? You have 2 choices. (1) Implement an assembly function that will mov ebp into eax and return, or (2) look at the address of count and infer the address that ebp points to.

3 Submitting the result (5 points)

Write a Makefile. Upon running make, your Makefile should generate libbt.so, a dynamic library, and libbt.a, a static library of the print_backtrace function. When you are ready to submit your code, store bt.c and macro.h along with a Makefile in a folder named Assn10. We will write a driver program to test the macros and the print_backtrace function and link to your library.

```
$ tar -cvzf assn10\_submission.tar.gz ./Assn10
```

Submit assn10.tar.gz on Blackboard.