CS 220
Homework Assem06

Turn in hardcopy of any programs.
You should test and debug programs. Do not submit just a paper solution!

1. Write an a assembly language program that displays the following figure:

   ****
   ****
   ***
   **
   *

You should define only a string containing “*****” in the data section of your program.
(Pass addresses to different portions of the string to print_str to display only portions of the
string.) Although you can directly calculate the address of a substring by adding an offset to the
address of the beginning of the string, the lea (load effective address) instruction can greatly
simplify address calculations. lea calculates the address of a memory reference and stores the
address in a destination register. Recall that memory references are of the form disp(base,
index, scale) where disp and scale are constants and base and index are registers. An
instruction of the form:

    lea disp(base, index, scale), %eax

would calculate the address (equal to disp + base + index * scale) and move the address into
the EAX register. All assembly instructions other than lea (when used with a memory reference
operand) would access the data at that address.

2. Repeat the previous problem, but dynamically build the string on the stack. (Don't forget to
null terminate the string.) Do not allocate any space in the data or bss sections. Be sure to
decrement the stack pointer before placing the characters on the stack and increment the stack
pointer (back to its original position) after you are finished with the string. You can also use
the lea instruction to calculate the addresses of items stored on the stack. Here are two
examples that illustrate the use of lea to calculate the address of items on the stack:

    lea -20(%ebp), %eax         # Address 20 bytes below EBP value
                            # (Base of 5 element int array)
    movl $3, %esi              # Index of 4th array element
    movl (%eax, %esi, 4), %ebx  # Copy 4th element to EBX

    movl $-3, %esi
    lea (%ebp, %esi, 4), %eax   # Address 12 bytes below EBP value
    movl (%eax), %ebx