

Due: 9:00AM, Friday, November 16 in class

10 points total

Please show your answers (and all work) on these sheets. Fold your paper the long way and write your name and **Homework 8** on the outside.

The following program reads a string on standard input and prints an integer in hexadecimal format based on the input string that it read.

```
#include <stdio.h>

/* Read a string from stdin into u.buf */
int evil_read_string()
{
    union {
        char buf[2];
        int i[1];
    } u;
    gets(u.buf);
    return u.i[0];
}

int main()
{
    printf("0x%08x\n:", evil_read_string());
    return 0;
}
```

Here is the gdb disassembly of the corresponding machine code on a Linux/x86 machine:

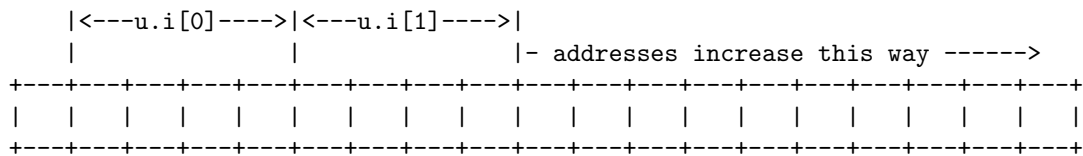
```
Dump of assembler code from 0x80483f0 to 0x8048441:
0x080483f0 <evil_read_string+0>:    push    %ebp
0x080483f1 <evil_read_string+1>:    mov     %esp,%ebp
0x080483f3 <evil_read_string+3>:    sub     $0x18,%esp
0x080483f6 <evil_read_string+6>:    lea    0xffffffc(%ebp),%eax
0x080483f9 <evil_read_string+9>:    mov     %eax,(%esp)
0x080483fc <evil_read_string+12>:   call   0x804830c <gets@plt>
0x08048401 <evil_read_string+17>:   mov     0xffffffc(%ebp),%eax
0x08048404 <evil_read_string+20>:   leave
0x08048405 <evil_read_string+21>:   ret
0x08048406 <evil_read_string+22>:   lea    0x0(%esi),%esi
0x08048409 <evil_read_string+25>:   lea    0x0(%edi),%edi
0x08048410 <main+0>:                lea    0x4(%esp),%ecx
0x08048414 <main+4>:                and    $0xfffff0,%esp
0x08048417 <main+7>:                pushl  0xfffffc(%ecx)
0x0804841a <main+10>:               push   %ebp
0x0804841b <main+11>:               mov    %esp,%ebp
0x0804841d <main+13>:               push   %ecx
0x0804841e <main+14>:               sub    $0x14,%esp
0x08048421 <main+17>:               call   0x80483f0 <evil_read_string>
0x08048426 <main+22>:               movl   $0x8048520,(%esp)
0x0804842d <main+29>:               mov    %eax,0x4(%esp)
0x08048431 <main+33>:               call   0x804832c <printf@plt>
0x08048436 <main+38>:               add    $0x14,%esp
0x08048439 <main+41>:               xor    %eax,%eax
0x0804843b <main+43>:               pop    %ecx
0x0804843c <main+44>:               pop    %ebp
0x0804843d <main+45>:               lea   0xfffffc(%ecx),%esp
0x08048440 <main+48>:               ret
End of assembler dump.
```

This problem tests your understanding of the stack discipline and byte ordering. You will need to know the hex values of the following characters:

Character	Hex value	Character	Hex value
'a'	0x61	'f'	0x66
'b'	0x62	'g'	0x67
'c'	0x63	'h'	0x68
'd'	0x64	'i'	0x69
'e'	0x65	'j'	0x6a

- Suppose we run this program on a Linux/x86 machine, and give it the string “edcba” as input on stdin. Suppose that the value of the `%ebp` register on entry to `evil_read_string` is `0xbf950998`.

Here is a template for the stack, showing the locations of `u.i[0]` and `u.i[1]`. Fill in the values of `u.i[0]` and `u.i[1]` (in hexadecimal) **and** indicate where `ebp` points just **after** `gets` returns to `evil_read_string`.



What is the 4-byte integer (in hex) printed by the `printf` inside `main`? (ignore any possible subsequent segmentation faults)

`0x_____`

- Suppose now we give it the input “ihgfedcba” (again on a Linux/x86 machine). Suppose again that the value of the `%ebp` register on entry to `evil_read_string` is `0xbf950998`.
 - List the contents of the following memory locations just **after** `gets` returns to `evil_read_string`. Each answer should be an unsigned 4-byte integer expressed as 8 hex digits.

`u.i[1] = 0x_____`

`u.i[2] = 0x_____`

- Immediately **before** the `ret` instruction at address `0x08048405` executes, what is the value of the frame pointer register `%ebp`?

`%ebp = 0x_____`

- Immediately **before** the `ret` instruction at address `0x08048405` executes, what is the big-endian hexadecimal value of the 4-byte word on the top of the stack?

Top of stack = `0x_____`

You can use the following template of the stack as *scratch space*. *Note*: this space is not graded.

