Name: _____

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, SHOW YOUR WORK.

1. Look at Section 4 ("Another Example") of our PLN unit on the JVM.

(a) (15) The assembly code compiled from line 26 in the source code on p.9 is shown in offsets _____ through _____ of **Min()** on pp.10-11.

(b) (20) Suppose we change line 17 in the source code to

```
public void Min(int Q)
```

and we delete line 31. No other changes are made.[1] Then the instructions in offsets 21 and 59 of **Min()** would change to _____ and _____, respectively. Moreover, the instruction in offset 59 would move to offset _____.

2. (15) The machine language for the subroutine **addone()** in our PLN unit on RISC will occupy a total of _____ bytes. (Count only the subroutine, not the calling code in **main()**.)

3. (15) Assume the page table structure on p.16 of the PLN unit on OS, and assume there is no TLB. Consider accesses to the page table when a page fault occurs, during the period just after the fault is discovered to the time that the instruction causing the fault resumes execution. During this time, _____ bytes will be read by the hardware, _____ bytes will be written by the hardware, _____ bytes will be read by the OS, and _____ bytes will be written by the OS. Fill in the blanks with numbers, and again, remember we considering only accesses to the page table.

4. Consider the following program and the GDB session which executes it:

```
.data
nb:  .long 6
us:  .string "ECS50\n"
.text
.globl _start
_start:
      movl ________, %eax
      movl $1, %ebx
      movl $us, %ecx
      movl nb, ________
      int $0x80
      movl $1, %eax
      int $0x80


Breakpoint 1, _start () at write.s:8
8             movl $1, %ebx
(gdb) p/x $eip
$1 = 0x8048079
(gdb) p/x $esp
$2 = 0xbfffda70
(gdb) p/x &nb
$3 = 0x8049094
(gdb) c
```

[1] With a change in the call in line 14. The method wouldn't make much sense then, but just suppose we make the indicated changes.

```
Continuing.
ECS50
Breakpoint 2, _start () at write.s:12
12            movl $1, %eax
(gdb) p/x $eip
$4 = 0x804808a
```

(a) (10) Fill in the blanks in the source code.

(b) (10) Show the hex value of c(ESP), i.e., the location of the top of the stack, at the instant the first **int** instruction finishes execution, just before the next Step A begins for the next instruction.

5. (15) Suppose **R** is a class variable, i.e. is declared **static**, and **S** is a local variable in slot 2. Show JVM assembly code which could be generated by the compiler for the statement

```
S = R*R;
```

Your answer must consist of four or fewer instructions. For full credit, you must minimize the total number of bytes occupied by the compiled code you show.

Solutions:

1.a 25-30; note that offset 30 implements the **break**

1.b The change in heading for **Min()** means two extra items in its Local Variables Area: **this** and **Q**, the former coming from the fact that **Min()** is now non-static. That means that offset 21 of **Min()** will change to **iload_3**. Also, since **Min()** now has type **void**, offset 59 changes to **return**, and we no longer have the instruction in offset 58, which had been the preparation for **ireturn**.

But there is more. Consider offset 2. We'd like to change it to **istore_4**, but there is no such instruction. Instead, it must be changed to **istore 4**; **istore** is a 2-byte instruction (the 4 here goes in the second byte), where we formerly had only one byte. We need to make similar changes at offsets 9, 15, 24, 25, 26, 42, 43, 47 and 48. So, the instruction which had been in offset 59 moves to offset 59-1+10 = 68. (And, as mentioned earlier, that instruction changes to **return**.)

2. There are 4 instruction, 4 bytes each, so 16 bytes in all.

3. The hardware does nothing to the page table during this time; it's all the OS. So, the first two answers are 0. A number of different answers were allowed for the other two blanks, given the multiple interpretations.

4.a $4, %edx

4.b The **int** pushes three items onto the stack, thus 12 bytes. ESP then becomes 0xbfffda70-12 = 0xbfffda64.

5. The straightforward compilation (and in fact the one chosen by **javac**) would be

```
getstatic #____
getstatic #____
imul
istore_2
```

But a shorter one (one byte less) is

```
getstatic #____
dup
imul
istore_2
```