





## Problem 2 – Becoming the supervisor (22 points)

You're writing a program for MiniOS when you realize that you need to get into supervisor mode. However, all programs in MiniOS are run in user mode. Dismayed, you stop by Scott's office hours to ask for some help. Scott points out that your problem can be solved (without rewriting MiniOS) because MiniOS doesn't protect any of its memory.

Explain how this fact helps you:

**If no memory is protected, that means that you can write to low memory and install your own code as a trap handler. Then, you can simply cause a trap to happen, which will transition to supervisor mode and call your code.**

**Many people said that “unprotected memory” meant that you could write to the SR. In fact, the SR is not stored in memory, but inside the processor. “MOVE to SR” is a special instruction in the 68K that requires supervisor mode to use.**

Write a code snippet that will take advantage of this “security flaw” to transition your code from user mode to supervisor mode. Your code should have no side effects. In other words, don't permanently modify anything that another program might need to use.

**There were various solutions, but they all had to do with using traps (although if someone claimed that you could write to the SR in part a, I gave them partial credit if they did it right). One solution is as follows:**

```
SupTrap PROC
    ORI.W    $2000, (A7)
    RTE

    RTS                ; This just makes MacsBug happy...
ENDPROC

Snip:
...
MOVEA.L    $0080, A1
LEA        SupTrap(PC), A0
MOVE.L     A0, $0080
TRAP      #0
MOVE.L     A1, $0080
...
```



**Problem 4 – But, where were they going, without ever knowing the way? (10 points)**

Give the address of the next instruction to execute under the following conditions. Assume that each instruction is loaded at address \$1000. Hint: the table on page 1 might prove useful.

Instruction	Next PC (32-bit)
ILLEGAL	2CE2 D2C0
TRAP #3	22A2 D2CE
JMP \$8086.W	FFFF 8086
RTE (in user mode)	1CE2 ECEA
ORI.B #0, D0	0000 1004

**Problem 5 – Returning from whence we came (10 points)**

The state of memory is as follows with (SSP) = 0347 A01A.

Address	Contents
0347 A018	0000
<b>0347 A01A</b>	A010
0347 A01C	004F
0347 A01E	ACEE
0347 A020	0000
0347 A022	7FFF
0347 A024	DCBA

What is the address of the next instruction to execute and the corresponding stack pointer in the following cases? You may assume that you are in supervisor mode.

	Instruction Address	Stack Pointer
RTD #4	A010 004F	0347 A022
RTS	A010 004F	0347 A01E
RTE	004F ACEE	0347 A022
RTD #-4	A010 004F	0347 A01A

### Problem 6 – I/O (22 points)

Assuming that the ACIA is set up, write the interrupt handler for receive:

```
IHand    PROC
         MOVE.L    D0, -(A7)

         MOVE.B    $00100004, D0
@Loop    BTST.B    #1, $00100000
         BEQ      @Loop
         MOVE.B    D0, $00100004

         MOVE.L    (A7)+, D0
         RTE

         RTS      ; To make MacsBug happy...
         ENDPROC
```

Install the interrupt handler in the correct place:

```
LEA      IHand(PC), A0
MOVE.L   A0, $00CC
```

Write the code to setup the ACIA:

```
MOVE.B   #$03, $00100000
MOVE.B   #$D5, $00100000
```

### Problem 7 – Interrupts and Masking (6 points)

a) If you're executing at priority level **2** and a level **3** interrupt occurs:

When will the interrupt be processed? **after the current instruction**  
What will the priority level be for the next instruction executed? **3**

b) If you're executing at priority level **3** and a level **3** interrupt occurs:

When will the interrupt be processed? **after the PPL falls below 3**  
What will the priority level be for the next instruction executed? **3**

c) If you're executing at priority level **7** and a level **7** interrupt occurs:

When will the interrupt be processed? **after the current instruction**  
What will the priority level be for the next instruction executed? **7**