

## CS 110 Practice Final Examination

---

### General instructions:

- The exam is open book, open notes, closed computer. If nothing else, you will definitely be using the M68000 programmer's reference manual.
- You may, but are not required to, use a simple hex calculator on the exam as mentioned in class. It shouldn't be needed.
- You will have 3 hours to complete the exam. It should take you less than this (though feel free to take the whole time). Note that the first question has a large number of points allocated to it because I expect it to take you the longest.

Problem	Points	Score
1	40	
2	22	
3	10	
4	10	
5	10	
6	22	
7	6	
<b>Total</b>	<b>120</b>	

**Name:** \_\_\_\_\_

**ID Number:** \_\_\_\_\_

Honor Code:

*"In accordance with both the letter and the spirit of the Honor Code, I didn't cheat on this exam."*

**Signature:** \_\_\_\_\_

Exception Vector Table (to be used for **all** problems)

Address		Assignment
0000 0000	1CE2 92C0	Reset : Initial SSP
0000 0004	2CE2 28C2	Reset: Initial PC
0000 0008	2CE2 6AC4	Bus Error
0000 000C	1CE2 A7C8	Address Error
0000 0010	2CE2 D2C0	Illegal Instruction
0000 0014	1CE2 2CCE	Zero Divide
0000 0018	2CE2 A2CA	CHK
0000 001C	2CE2 22CE	TRAPV
0000 0020	1CE2 ECEA	Priv
0000 0024	2CE2 CECE	Trace
0000 0028	1CE2 A2CE	Line 1010 Emulation
0000 002C	2CE2 C2CE	Line 1111 Emulation
0000 0030		Reserved
0000 0034		Reserved
0000 0038		Reserved
0000 003C	1CE2 2ECE	Unitialized Interrupt Vector
0000 0040 - 0000 005C		Reserved
0000 0060	2DE2 22CA	Spurious Interrupt
0000 0064	32A2 22CE	Level 1 Autovector
0000 0068	0000 2CEA	Level 2 Autovector
0000 006C	42A2 0000	Level 3 Autovector
0000 0070- 0000 007C	12A2 A2CE	
0000 0080	2AE2 E2CE	Trap #0
0000 0084	13A2 2ECE	Trap #1
0000 0088	0000 22E8	Trap #2
0000 008C	22A2 D2CE	Trap #3
0000 0090- 0000 00BC	33A2 2DC8	
0000 00C0- 0000 00FC		Reserved
0000 0100- 0000 03FC	22C2 22DA	User Interrupt Vectors





### c) MYPS calling convention

A friend over at Motorola tells you that he's come up with a "new" instruction designed to speed up a program's execution. The new instruction is called "JAL" (jump and link). It works the same as "JSR" except that instead of pushing the return address on the stack, it places the return address in A0. This same friend also tells you that he's pushing for a new calling convention that he calls the MYPS convention. His MYPS convention is just like the C convention except that the JAL instruction is used for the call and the first data and address parameters are placed in D1 and A1 respectively (any additional parameters are passed as usual on the stack).

Implement the callee and caller using the MYPS calling convention. You are not required to use LINK or UNLK nor must you put local variables on the stack (though you're free to do either of these). You may assume that A0, A1, and D1 have already been saved by the caller routine prior to the call of `Compute()`. You do not need to draw a picture of the stack for this problem.

caller:

callee:

## **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. Note: (Spring '99): Scott was the TA the quarter this was given. 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:

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.



**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	
TRAP #3	
JMP \$8086.W	
RTE (in user mode)	
ORI.B #0, D0	

**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		
RTS		
RTE		
RTD #-4		

### **Problem 6 – I/O (22 points)**

Write code that will automatically “echo” everything coming into the serial port. That means that every time a character is received, it should be sent out. Guidelines:

- You should use the ACIA (MC6850) described in class. Assume that the ACIA is configured to produce level-3 autovectored interrupts (when it needs to produce interrupts).
- The ACIA control/status register is at \$0010 0000; the xmt/rcv register is at \$0010 0004.
- Your communications mode should be: divide-by-16 clock, 8 data bits, 1 stop bit, no parity, RTS inactive.
- You should receive characters using interrupt I/O, but you should send characters using busy-wait I/O. This should be fine as long as nobody else is sending characters and your sending is the same speed as your receiving.

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

**Install the interrupt handler in the correct place:**

**Write the code to setup the ACIA:**

**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? \_\_\_\_\_

What will the priority level be for the next instruction executed? \_\_\_\_\_

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

When will the interrupt be processed? \_\_\_\_\_

What will the priority level be for the next instruction executed? \_\_\_\_\_

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

When will the interrupt be processed? \_\_\_\_\_

What will the priority level be for the next instruction executed? \_\_\_\_\_