Tips for Lab Success

Programming

- Write your code in small pieces (subroutines). You will then be able to test each subroutine. This makes debugging far easier than writing the entire program and debugging the entire thing at once.
- Reuse subroutines from previous labs. Many subroutines can be reused from lab to lab with minor modifications.
- Your main program should consist of very little code other than subroutine calls.
- Shells were provided to you for early labs but you should make your own shell to use throughout the semester. Your shell should contain many of the often used subroutines plus all the jump vectors and equates that you have been using. This will greatly reduce 'typos' and save you time.
- Make use of the example programs on the web, there is usually a very high correlation between them and the labs.
- Make use of the TA's. Office hours are your opportunity to get help, USE THEM!!

Debugging at home

- Try your code before coming to lab.
- Even if a lab requires use of hardware in the lab you can usually test much of your code home.
  Examples
  - If you have a menu system you should be able to see if that works before ever coming to lab.
  - If you have a timer interrupt you should be able to set a breakpoint at the beginning of its ISR and see if you ever get there.
  - Invest a few dollars in some hardware. The keypad lab could have been done at home for a few dollars in parts.

In Lab

- Once in the lab all the code should be written. The first thing you should do in lab is hook up any necessary hardware and try your program. You should not come into lab and start looking at your code, that should have been done before you came to lab.
- If your program does not work do NOT start looking at your code. Set a breakpoint after your first subroutine. See if you get there and also see if what has been done is what you expect; have memory locations and registers been changed as you expected? If not then you have narrowed down a problem to one subroutine and can then set breakpoints and use the trace command in that subroutine to find your problem. If everything is working up to this point then set a breakpoint after the next routine, see if you get there and continue the process.
- If working with interrupts an important thing to find out is whether or not you ever get to your ISR, this can be determined by setting a breakpoint at the first line of your ISR. If you do not get there then some possible problems are that your interrupt vector is wrong, you have not enabled the interrupt system (CLI), or you have not correctly set up that particular interrupt.
- Make use of your TA. Ask questions!

Odds and Ends

- **Independent thinking** - Do not accept everything in the lab as the way things must be done. As long as your program fits the guidelines it will be fine. If you have a question ask. It is very possible that you will come up with a method of doing something that is better than what is suggested to you in the lab. Do not blindly follow the labs suggestions if you do not understand why something needs to be done a certain way.
- If you want to trace through some code but cannot because interrupts are interfering you can turn off interrupts by modifying the condition code register to set the interrupt mask.
- Do NOT write the entire lab at once. Most labs can be easily split into parts. Start with the part that you think will be easier. Then do the others.