Lab Overview

In this lab assignment, you will do the following:
- Learn how to use an assembler and simulator. Start using the 8051 and ARM processors.
- Learn how to use the WinCUPL/WinSim design suite for the Atmel AT16V8C SPLD.
- Plan the layout of your development board and obtain your parts kit. Build your basic hardware.
- Start learning schematic capture and gain skills in soldering.

The Part 1 Elements of this lab assignment should be done (flexible date) by **Friday, Jan. 25, 2019**.
The signature due date for the entire lab assignment is **Friday, Feb. 8, 2019**.
The submission due date (when your files must be turned in) is **11:59pm Saturday, Feb. 9, 2019**.
The cutoff date for this lab is **Saturday, Feb. 16, 2019**.

Students can collaborate on Lab #1 but must individually develop their own unique hardware and software. This lab is easier than the following labs and it is weighted as a small part of your course grade. This gives you a chance to come up to speed on basic hardware and firmware concepts, to make sure that you have all the parts you need to do the assignment, and to make sure that you have access to the lab and the lab equipment is all functional. The following lab assignments will be more involved and more time consuming, so you will have to plan your time wisely.

Lab Details

1. Review Homework assignments #1 and #2, which are associated with Lab #1.

2. Obtain lab access. If you do not already have one, obtain a Buff OneCard, needed for access to the engineering center and the 1B28 laboratory. Continuing Education students not seeking a degree can obtain a card for ~$30. You will need to get this card from the Buff OneCard office on campus. For more information, see https://services.jsatech.com/index.php?cid=59. **Please get your card ASAP, and submit the lab access request form at** http://bit.ly/ECEELabs or https://docs.google.com/forms/d/e/1FAIpQLScSfPh2zOIPFGjfJ8ruJbOS_7qk7zuh5k9BUA2IEI6dU8Z39w/viewform?usp=sf_link

3. For the software part of this lab, you will need to use an 8051 assembler and simulator of your choice. The ASM51 or AS31 or ASX51 (included in the SDCC distribution) assemblers and the Emily52 or EdSim51 simulator (http://www.edsim51.com/ are recommended. You may use the tools installed on the computers in the lab, or you can obtain a copy of the Emily52 demo kit from the course web site. This demo kit comes as a self-extracting .exe file, and includes a version of both the ASM51 assembler and the Emily52 simulator, as well as example files. Refer to the assembler homework assignments for more information.

> Note: You may need to use an emulator like DOSBox in order to install and run DOS-based tools in 64-bit environments. If you're running 64-bit Windows, go to www.dosbox.com and download for Windows OS. This is an x86 emulator which will run in Windows x64. To keep things simple to use, create two directories with their corresponding files in the C: directory – you'll have a Dosbox folder and an Emily52 folder. Run the dosbox executable and the Z: prompt will come up. Follow these steps:
>
> Z:\> mount c c:\emily52
> Z:\> C:
> C:\> emily52 filename.hex /overlap
>
> Emily52 is now running in a x86 emulator.

For more information, see Homework #2 and the Assembly Language and Emily52 overview documents available on the course web site. We do not intend to spend too much time on these tools this semester.

1. Write a single assembly language program which meets the requirements below.

   This program shall calculate the following equation using integer division: $Z = ((X*4)/Y)$
   In this assignment, X, Y, and Z are unsigned 8-bit values.

   Example: If X=0x6 and Y=0x05, then Z=0x04.   (6*4 = 24 and 24 div 5 = 4 with a remainder of 4)

   Preconditions:
   - Accumulator contains value of X.
   - B register contains the divisor Y.

   Post Conditions:
   - Internal data memory (IRAM) location 0x20 contains the value of (X*4)
   - IRAM location 0x21 contains Z, the 8-bit quotient of (X*4)/Y (in hexadecimal)
   - IRAM location 0x22 contains the 8-bit remainder of (X*4)/Y (in hexadecimal)
   - IRAM location 0x30 contains the error code.

   Other requirements:
   - This program must implement an algorithm that does <u>not</u> use the 8051 divide (DIV) or multiply (MUL) instructions.
   - Use an 8051 shift/rotate instruction to implement the multiplication.
   - The program must start at address 0000h.
   - When the calculation is complete, the program jumps to a label named 'ENDLOOP' and enters an infinite loop.

   Error handling:
   - If no error occurs during program execution, the program shall write a '0x00 into the error code in internal memory.
   - If the divisor is 0, then the program shall write a '0x01' into the error code in internal memory and then immediately jump to 'ENDLOOP' without performing any calculations
   - If the calculation exceeds an 8-bit value when multiplying, then the program shall leave the least significant 8 bits of the result in Z and shall write a '0x02' into the error code in internal memory.

   Implementation suggestions:
   - Refer to the test1.asm and test3.asm files distributed with the demo version of Emily52 for simple programming examples.
   - Implement your code in stages. One possibility for this type of incremental development is:
     i. Stage 1: First, implement the program using the DIV instruction and do not include any error handling.
     ii. Stage 2: Once that is working, replace the DIV instruction with your algorithm.
     iii. Stage 3: After that is working, add the instructions necessary to implement the required error handling.

   Test your program with a simulator by setting the accumulator and B register to various initial values.
   Note: Use a combined code and data space in the 8051 (/overlay option in Emily52). Practice using the various capabilities of the simulator.
   During the demo, the TA will have you use the simulator to test the results for various values of dividend and divisor. Make sure your program can correctly handle different starting values.

- <mark>Submit your final commented firmware.</mark> During signoff, show the TA your code and .LST file, if it exists.

2. For the SPLD design suite part, you will need the Atmel WinCUPL tools, which are available on the Microchip web site and through a link on our course web site. These tools are also installed on the computers in our lab. Learn how to use Atmel WinCUPL and WinSim. Review the tutorials and example code available on the course web site. Focus on learning the basics of these tools only, as they are not a primary element of this course.

3. Using WinCUPL, develop code for the Atmel AT16V8C SPLD. Assign A15, A14, A13, A12, /RD, and /PSEN from the processor to six of the SPLD inputs. Generate two outputs:
   /READ = /RD & /PSEN, and **/CSPERIPH = !(A15 & A14 & A13 & A12)**.

   Verify your logic with WinSim, using at least <u>six</u> test vectors to show correct logic functionality for a well-chosen subset of all possible input conditions. The /READ output of the SPLD should be toggling when either /RD or /PSEN toggles. The /CSPERIPH OUTPUT should be high most of the time, and should be low only for addresses in the range of F000h-FFFFh.

- As part of the signoff procedure, show the TA your commented .PLD source file and .SI simulator input file. Explain how these files are structured.
- Note: Students are allowed to use discrete logic in their hardware implementation if they prefer that solution over the SPLD.

COMPLETE ALL THE ITEMS ABOVE FOR THE PART 1 ELEMENTS SIGNOFF SHEET.

COMPLETE ALL THE ITEMS BELOW FOR THE PART 2 ELEMENTS SIGNOFF SHEET.

4. Obtain a parts kit for the lab. The TA's will make these available for purchase.

   - Payment may be made via PayPal to Linden.McClure@Colorado.EDU or by cash.
   - Upon receiving the kit, check to make sure all items are present and contact the TA's right away if any parts are missing. The parts kit contents list is posted on the course web site.
   - Wire wrap wire may be obtained in the laboratory. Please only take what is required to implement your microcontroller board (please only take 2-3 yards of each color needed at a time).
   - Soldering irons and solder will be available to use in the laboratory.
   - Please leave the lab equipment in the lab, make sure to turn off the soldering irons, and make sure the door to the lab is closed and locked when you leave.
   - You may sign out a tool kit with wire wrap tool, cutters, needle nose mini-pliers, power supply, 28-pin ZIF socket, NVRAM, and RS-232 cable for the semester. You may also sign out items such as a portable logic analyzer, digital logic probe, high frequency oscope probe, etc. **Take good care of these items, as you are financially responsible for them.**

5. If you don't have experience with soldering electronics, read the article "A Guide to Better Soldering", available in the lab or visit a web site (e.g. YouTube or Metcal) and read about soldering techniques and tips. Answer the following questions for yourself:
   - Why is flux used? What type of flux should be used with electronic circuits? Will flux remove grease from a connection point?
   - What is a cold solder joint, and how is it created?

6. [Optional] Skim the article "Handling of Power Plastic Transistors", available in the laboratory. You should always take care not to stress the pins or packaging of any electronics.

7. You will be using the 6"x8" printed circuit board (PCB) from your parts kit. Study the schematics included in the document "ECEN 5613 PCB Layout and Partial Schematics", which is available on the course web site. The schematics show you some of the circuit connections that are present on your PCB. Note that not all of the components shown in the partial schematics will end up being populated on your board; the components are in the schematics so that pads (through hole or surface mount) would be present in the PCB design. You will need to determine which components you will need to load during the semester.

8. Your initial hardware assignment will be to implement your core microcontroller design. The basic circuit elements consist of an 80C51 microcontroller with power-on and run-time reset circuitry and an 11.0592MHz crystal oscillator, a 7805 (or 340T5) +5V voltage regulator, and hardware, such as your PCB, standoffs, power connector, 9-pin RS-232 connector, etc.

Design your power supply circuit and draw a schematic using any tool like Orcad Capture, KiCad, Altium, or Mentor. You should include the 2.1mm power jack, power switch, the 7805 voltage regulator, and a power-on LED which glows whenever power is applied to your board. Some starter schematics are on the course web site (in OrCad .DSN and Adobe .PDF formats).

Design your oscillator circuit using the 11.0592 MHz crystal and the 27pF capacitors and add this circuit to your schematic.

Design your power-on reset circuit with run-time reset capability. Include an RC circuit and a pushbutton. You may choose to mount the pushbutton on the top or the bottom of your board.

- ▪ Include a PDF printout of your schematic as part of your lab submission.

9. **Before soldering in any components and before using your bare PCB, you must test to make sure that you do not have any short circuits. Using a multimeter, test between your +5V and ground connection on the PCB to verify that there is an open circuit between power and ground.** If you catch a PCB flaw at this point, you won't waste time and money debugging your board after you've added your circuits.

Study your PCB carefully and compare it to the partial schematics. Use a multimeter to learn how the power and ground planes on your board are designed. Are they connected directly to the VCC and GND pins on each chip? Are they connected directly to the VCC and GND headers distributed around the board? Which of the capacitors in the partial schematics are through hole and which are surface mount technology (SMT)?

10. Spend some time planning your layout (see the "Example Board Layout" document provided). A lot can be learned from laying out a circuit poorly, but it's usually much easier to correct an error at the layout stage of the game rather than after you have already soldered in chips and wrapped 50 wires. You will want to try to keep your wires short to minimize noise problems. To help in development, you will probably use standoffs to support your board in its four corners. It is recommended <u>not</u> to drill any additional holes in your PCB, as you take the chance of damaging existing traces or shorting together the VCC and GND planes. In addition, it is suggested that you leave a keep-out area (perhaps 0.5" to 0.8" wide) around the edge of your board, so that you can add connectors or other parts later in the semester. **NOTE:** Parts to be added to your board this semester include a 32KB EPROM; (optionally) a 32KB NVRAM/SRAM; supporting glue logic, such as your decoding circuits, pull-up resistors and an address/data line demultiplexer; a MAX232 RS-232 line driver/receiver; 8-pin serial EEPROM, and an LCD. You need to leave room for these parts on your board.

11. Using a marker or pen, neatly print **<u>your name</u>** and the text "**S2019**" in the big white space on the bottom side of the PCB.

12. If you are new to soldering, you may want to complete the SMT soldering lab and also do some practice soldering on a scrap PCB before doing any soldering on your board for this course. Ask the instructor for details during lecture.

13. Insert the wire wrap socket(s) from the top side of the board. Carefully solder the wire wrap sockets to the PCB (be careful not to use too much solder, as you don't want to solder together your power and ground planes by mistake). If you examine the PCB carefully, you'll notice that not all the pins for each chip have electrical connections on the board - the VCC pin, GND pin, and some signal pins on the chips have connections to traces on the PCB. At a minimum, the VCC and GND pins of each socket must be soldered to the board, as well as the pins which have traces on the board. **Make sure you've completed all the soldering before you start wire wrapping to a socket!** It's easy to melt

14. You will need to wire up the 2.1mm power jack, a master on/off power switch and +5V regulator. If you plan to use a heat sink on your regulator, be sure to leave enough space on the board. Pay close attention to the polarity of the wall adapter plug. Different plugs have different polarities, and if you're not careful and use the wrong polarity, you can destroy your components on your board. You will use the diode bridge rectifier on your PCB to protect against damage caused by using a power supply with the opposite polarity. Use heavier wire (e.g. 22 AWG) for your main power connections or double up your wire wrap connections. Remember, when you solder circuitry such as the 7805 regulator, be careful about applying too much heat, or you'll end up damaging or destroying the circuitry. If you don't use the diode bridge, clearly label the polarity of your power connection so that you'll always know what kind of adapter to use.

    Example:　　+ ──⊙── –

15. You will also want to mount your RS-232 connector at this time. Solder in the 9 pins of the RS-232 connector (reference designator J5). You can also solder in an 8-pin or 9-pin header so that you'll be able to wire wrap to the RS-232 pins later in the semester. If using an 8-pin header, make sure to solder it in to positions 1-8 on the header outline for reference designator J4.

16. The individual T44 wire wrap pins can be used to hold discrete components as necessary.

17. You must choose wire colors to use when implementing your circuits. For this class, red is always used for power and black is always used for ground. Your other connections must be colors other than red or black. You are free to choose whatever other colors you wish; however, remember that when debugging, it is easier when particular wire colors signify a particular type of signal, and when the chosen colors contrast with each other and the background to a great degree. For example, it is very difficult to debug a board when the background is white and when only white wire wrap wire is used for all connections. One color scheme is shown below:

Signal Type	Color	Comment
Power	**Red**	**Must be red**
Ground	**Black**	**Must be black**
Multiplexed Address/Data	Gray	Your choice of color
Buffered Data	Yellow	Your choice of color
Latched Address	Blue	Your choice of color
Control Signals	Orange	Your choice of color
Serial Port Signals	Green	Your choice of color

18. After you have become proficient at basic wire wrapping technique, it is time to start wire wrapping your base microcontroller board. The strategy often used is first to lay down the wires which are least likely to move or change, so that any wires you have to adjust later are laying on top of the other wires. First hook up all your ground connections. Remember what you learned about loop size and placement of the decoupling capacitors–**keep the caps close to the IC power and ground pins and keep the leads as short as possible**. If you need to solder your decoupling caps into the T44 pins, now is a good time to do this. When wiring, make sure you get good connections so that the sockets are not terribly loose. Leave just a little slack in the wire connections so that you can debug easier later in the semester. If the wires are too tight, then it's difficult to move them when you need to see silkscreen labels or other wires underneath. After you have completed wiring up your ground connections, wire all your +5V connections. Remember to strip at least 3/4" of bare wire in order to wire wrap. 0.75 inches is about this long: ▬▬▬▬

19.

problem. **Use this approach as you are wiring up your board—take small steps, and verify your work often.** This will save time in the long run.

Now, **before** putting in any chips, turn on power and verify that you have +5V and ground at the correct pins on your sockets. Measure the voltage with a digital multimeter. Using the oscilloscope, verify that the VCC voltage is close to 5.0V and looks stable at 5.0V, without big oscillations.

21. Wire up your reset circuitry and your oscillator circuitry, and tie the 8051 $\overline{EA}$ line low**. Keep the oscillator circuitry close to the microcontroller and use very short wires. Pull-up your Port 0 pins to V$_{CC}$ so that the processor doesn't go into power-down mode.** You can use a resistor network in a SIP package or discrete resistors. You may not need to pull up all 8 data lines.

22. If you have verified all the basic connections and circuitry, you can now insert your microcontroller (with the power off). Then, turn on power, and use an oscilloscope verify that your ALE line is oscillating at the correct frequency (1/6 the XTAL2 frequency). Use an oscilloscope to view the XTAL2 waveform and check its frequency. If you want, use a logic analyzer to verify that a fetch from address 0000h is occurring immediately after the processor comes out of reset. You need to make sure that your oscillator starts reliably **every time** power is turned on and after a reset. **You may need to adjust the oscillator load capacitance**. Be aware that the oscilloscope probe capacitance can affect your oscillator startup when you probe the XTAL1 and XTAL2 pins.

23. Using the oscilloscope, verify that the VCC voltage is close to 5.0V and that the peak to peak noise between VCC and GND on your C501 is less than ~800mV. If necessary, add more bypass capacitors (1.0uF/4.7uF/etc.) across the C501 power pins.

24. Wire up the '373 latch to the microcontroller.

25. All ICs you add to your board must be labeled with identification of the IC name, pin numbers, and all signal names, similar to the information present on the PCB silkscreen. Wire wrap label templates are available on the course web site (search for "Wire Wrapping ID"), and may be helpful to you for the parts you add (e.g. the SPLD) which don't already have silkscreen present on the PCB.

26. Connect and test your digital logic.

    a) Wire the Atmel AT16V8C SPLD socket. Attach A15, A14, A13, A12, /RD, and /PSEN from the processor to six of the SPLD inputs (matching the assignments in your WinCUPL code).

    b) Use digital logic (e.g. program the SPLD) to generate two outputs:
    /READ = /RD & /PSEN, and **/CSPERIPH = !(A15 & A14 & A13 & A12)**.

    c) Verify the SPLD outputs. Examine the /READ output of the SPLD to verify that it is toggling when /PSEN toggles. The /CSPERIPH OUTPUT should be high most of the time, and should have a 15/16 duty cycle (it will be low only for addresses in the range of F000h-FFFFh, while the processor continuously cycles through addresses 0000h to FFFFh).

27. Think carefully about the steps you will need to take to complete your base board, and formulate a plan for incrementally wiring and testing parts of your circuit.

28. Start learning about the ARM architecture and using your ARM development board:

    a) Learn about the TI MSP432 (ARM Cortex M4) Launchpad (http://www.ti.com/tool/msp-exp432p401r).

    b) Read the overview (http://www.ti.com/lit/pdf/slau596) and run the code specified by the TA on your Launchpad.

    c) Explain your key learnings to the TA's.

NOTE: Carefully prepare for your signoff with the TA's. Make sure you are well organized and that you understand the lab material and your implementation completely. Be prepared to demonstrate your skills with the lab equipment and answer questions posed by the TA. Be ready to start your signoff on time. Demonstrate all elements of the lab assignment in a time efficient manner and utilize clear communication.

You will need to obtain the signature of your TA on the following items in order to receive credit.

==The Part 1 Elements of Lab #1 should be completed and signed off by **Friday, Jan. 25, 2019** in order to give you time to complete the Part 2 Elements upon receipt of your parts kit. Both signoffs are due by **Friday, Feb. 8, 2019**. You need to submit both of your signoff sheets and other required elements by **11:59pm Saturday, Feb. 9, 2019**.== Labs completed after the signature due date or submitted after the submission due date will usually receive grade reductions, but there is leniency on Lab #1.

Print your name below and then demonstrate your working hardware/firmware in order to obtain the necessary signatures. All items must be completed to get a signature, but partial credit is given for incomplete labs. Receiving a signature on this signoff sheet does not mean that your work is eligible for any particular grade; it merely indicates that you have completed the work at an acceptable level.

Student Name: _____

Checklist

- ❑ Student demonstrates detailed knowledge of a simulator (including changing register values, editing data memory, using breakpoints, single stepping, uses /overlay option, etc.)
- ❑ Student assembly program works correctly
- ❑ Student demonstrates detailed knowledge of WinCUPL and WinSim, logic equations correct

Student Answers to Lab Questions

1. **How many bytes of code space does your program require?**
   (Show how you arrived at your answer.)

   Code Size? _____

2. **How long did your program take to execute for X=0x20 and Y=0x0A? Assume an 11.0592 MHz clock and include the instructions executed from the beginning until you reach the ENDLOOP label. Show the TA your detailed calculations on the code listing during your signoff.**

   Execution Time? _____

Instructor/TA Comments: □   □   □   **TA signature and date**

FOR INSTRUCTOR USE ONLY	Not Applicable	Poor/Not Complete	Meets Requirements	Exceeds Requirements	Outstanding
SPLD code	□	□	□	□	□
Assembly Language Code Style	□	□	□	□	□
Required Elements functionality	□	□	□	□	□
Sign-off done without excessive retries		□	□		
Student understanding and skills	□	□	□	□	□
Overall Demo Quality	□	□	□	□	□

Comments:

==NOTE: This submission sheet should be the top/first sheet of your submission.==

Print your name below, answer the questions, and then demonstrate your working hardware in order to obtain the necessary signatures. All items must be completed to get a signature.

Student Name: _____

Checklist

- ❑ Schematic of acceptable quality, Student name on board in permanent ink
- ❑ Pins and signals labeled, decoupling capacitors, and two 28-pin wire wrap sockets present on board:
- ❑ Mounting hardware present (e.g. standoffs or an enclosure)
- ❑ Power switch and LED, voltage regulator functional, power jack present
- ❑ Power-on Reset (RC) and Run-time Reset (pushbutton), 8051 bypass cap is present
- ❑ RS-232 connector mounted, 74LS373 transparent latch wired
- ❑ Logic outputs correct (e.g. SPLD generation of /READ and /CSPERIPH; view SPLD code)
- ❑ Student displays good knowledge of oscilloscope
- ❑ Peak to peak noise measured across processor VCC and GND is < 800mV
- ❑ Oscillator functional (check for correct ALE/XTAL2 signals after power on-off cycles)
- ❑ ARM development board functional, student can demonstrate the basic software.

Student Answers to Lab Questions

1. **What voltage is present at the regulator input?** Use a digital multimeter. _____

2. **What voltage is present at the regulator output?** Use a digital multimeter. _____

3. **What peak to peak noise is present across the processor VCC and GND?** Use an oscilloscope.

   Measured value at processor package pins on top side of board: _____

   Measured value at wire wrap socket pins on bottom side of board: _____

4. **How long is the processor held in reset after the run-time reset pushbutton is released?** Use an oscilloscope and try to measure the time between the release of the pushbutton and the time when noise from ALE is observed on the RST signal.

   Measured value: _____

5. **What frequency is present at the ALE pin?** Use an oscilloscope. _____

Instructor/TA Comments: □   □   □   **TA signature and date**

FOR INSTRUCTOR USE ONLY	Not Applicable	Poor/Not Complete	Meets Requirements	Exceeds Requirements	Outstanding
Schematics, SPLD code	□	□	□	□	□
Hardware physical implementation	□	□	□	□	□
Required Elements functionality	□	□	□	□	□
Sign-off done without excessive retries		□	□		
Student understanding and skills	□	□	□	□	□
Overall Demo Quality	□	□	□	□	□

Comments:

NOTE: This submission sheet should be the second sheet of your submission.

Submission Sheet 2

Submission Sheet

Instructions: Print your name below and sign the honor code pledge. Separate the signoff and submission sheets from the rest of the lab and turn in a scan (or clear picture) of these signed forms, the items in the checklist below, and the answers to any applicable lab questions in order to receive credit for your work. No cover sheet please. **Submit all items electronically via Desire2Learn to reduce paper usage. D2L is https://learn.colorado.edu.**

In addition to the items listed on the signoff checklist, be sure to review the lab for additional requirements for submission, including:

- ❑ Scan of signed and dated Part 1 signoff sheet as the top sheet (No cover sheet please)
- ❑ Scan of signed and dated Part 2 signoff sheet as the second sheet
- ❑ Scan of submission sheet with signed honor code pledge as the third sheet
- ❑ PDF of complete and accurate schematic of acceptable quality (all components shown).
- ❑ Fully, neatly, and clearly commented assembly code.

Make copies of your code, SPLD code, and schematic files and save them as an archive.

Student Name: _____

Honor Code Pledge: "On my honor, as a University of Colorado student, I have neither given nor received unauthorized assistance on this work. <mark>I have clearly acknowledged work that is not my own.</mark>"

Student Signature: _____

1. **How much power is dissipated in the regulator, assuming a load current of 300mA?** Assume that the regulator is drawing the max quiescent current shown in the data sheet (use the correct data sheet for the regulator you have on your board). Neatly show all your work.

Calculated value: _____

<u>Comments:</u>