# CS 331 Lab #1
# Basic I/O with the A/D – D/A Card

Spring 2003

Week #1: 2/5/2003 – 2/7/2003
Week #2: 2/12/2003 – 2/14/2003
Report due in lab: 2/19/2003 – 2/21/2003

## 1 Overview

There are two parts to this lab:

1. Write C functions to communicate with the I/O card at the port level. These functions should utilize inline AT&T style assembly.

   `void hw_out(unsigned int port, unsigned char val)` takes the 8-bit value in *val* and writes it to the port address specified in *port*.

   `unsigned char hw_in(unsigned int port)` retrieves the 8-bit value from the port address specified in *port* and returns it.

2. Write a C or C++ program (using `hw_out()` and `hw_in()`) to repeatedly...

   (a) Read a decimal voltage value between −5 and +5 volts from the user.

   (b) Output the specified voltage on the I/O card's D/A channel 0.

   (c) Sample the analog voltage on the I/O card's A/D channel 0 and print the value to the screen.

   The I/O card's "output" (D/A channel 0) has been connected to its "input" (A/D channel 0) with a wire. Therefore, the voltage that your program reads back in step (c) should be that which it output in step (b). The oscilloscope is also connected to the output–input loop and may be used to verify the voltage on the wire.

## 2 Intel x86 I/O Ports

The Intel x86 architecture separates I/O port addresses from memory addresses (although memory mapped I/O is also supported). For example, using the Intel syntax, "`IN AL, 40H`" or "`OUT 40H, AL`" would be used to read or write to port 40H. Only $2^8 = 256$ ports (00H to FFH) may addressed in this mode. However, by using registers we can address up to $2^{16} = 65,536$ ports. For example, "`IN AL, DX`" or "`OUT DX, AL`" where the port address is stored in `DX`.

# 3 Procedure

1. Make a `lab1` directory to put your code for this lab in. Copy ∼`cs331/lab1/Makefile` to your directory.

2. Create a file called `iodas.c` for the `hw_out()` and `hw_in()` functions. The following code implements `hw_out()`. You need to finish `hw_in()`.

```
1  void hw_out(unsigned int port, unsigned char val)
2  {
3          __asm__ __volatile__ ("outb %b0, %w1"
4                                : /* no output */
5                                : "a" (val), "d" (port) );
6  }
7
8  unsigned char hw_in(unsigned int port)
9  {
10         /*  Declare a variable to hold the input value.  */
11         /*  (What should the type of this variable be?)  */
12
13         /*  Use AT&T style assembly to read the input value from the desired port.  */
14
15         /*  Return the input value.  */
16 }
```

3. When finished writing `iodas.c`, compile it into an object file by running "`make iodas.o`".

4. Execute ∼`cs331/lab1/demo` to see an example of the program you now need to write.

5. Create a file called `lab1.cc` and implement the program described in the Overview section using C or C++. The following hints will be of use as you work on the program.

   - In order to use the `hw_out()` and `hw_in()` functions, `lab1.cc` must begin with the following code.

```
1  extern "C"
2  {
3          void hw_out(unsigned int port, unsigned char val);
4          unsigned char hw_in(unsigned int port);
5  }
```

   - The I/O card has a base address of `0x300`. Use the lecture notes and the *CIO–DAS1600 User's Manual* to determine which offsets are needed in your program.
   - The DAS–1600 interface uses 12 bits for input and output. Therefore, the range of input and output values is $0 \ldots (2^{12} - 1)$ or $0 \ldots 4095$. The hardware configuration maps this range to $-5 \ldots +5$ volts for output and $-10 \ldots +10$ volts for input.
   - It takes a brief amount of time for the voltage on the output line to stabilize after it is changed. Therefore, you should pause the program momentarily before starting the A/D conversion. A simple for loop that counts from 0 to $50,000$ is a hackish way to do this.

6. When finished writing `lab1.cc`, compile it into an executable by running "`make`". Execute the program with "`./lab1`". Small errors are expected in the D/A → A/D trip. How do your errors compare to those seen with the demo program?

# 4 Demonstration

Notify the TA when you are done with the assignment and ready to demonstrate your code. The TA may randomly pick one of you to do the demonstration and ask another to explain how it was done. *Make sure that everyone in your group understands how all of the code works before your demonstration.* Some lab topics and problems will appear on the mid-term and final exams.

# 5 Lab Report

Your group should turn in one typed lab report when Lab 2 starts (2/19/2003 – 2/21/2003). The report should include the following items.

- An organized header that, at the minimum, contains your names and NetIDs, as well as your lab section (day and time), workstation, and username. For example:

> CS 331 Lab Report #1
>
> Tim Eriksson (eriksson@uiuc.edu)
> Xiaolei Li (xli10@uiuc.edu)
>
> Section: Friday at 10:00am
> Workstation: emb8
> Username: group38

- An overview of your program implementation.
- A discussion of any problems you encountered while working on this assignment, and how you overcame them.
- What each team member contributed to this particular lab assignment.
- A listing of your commented code (`iodas.c` and `lab1.cc`).