TECH 3233
Lab #2
Atmel, ASM and C

[note: highlighted notes made during lab]

Purpose: To familiarize students with how a C program is compiled into Machine Language and how it is placed in memory.

Discussion: As we learned in Lab #1, data can be placed into and retrieved from memory. This week we will type in our first C program in Atmel Studio 7.0, compile it and then send it to the Arduino board. We will also look at how that program is stored in memory (in machine language).

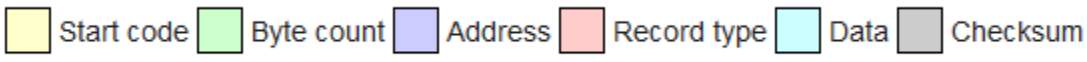
When we type in a C program, the program follows the rules of the C programming language with key words and phrases (ie while () being a loop). When we build or COMPILE a program, these key words are translated to what the CPU understands (Machine language). To send code over to the Arduino, this machine language is put into a .HEX file.

The hex file contains the code, data and additional information that helps the CPU place it in memory and make sure the code has been sent correctly.

Lets take a look at an example HEX file[1]:

```
:10010000214601360121470136007EFE09D2190140
:100110002146017E17C20001FF5F16002148011928
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

Each line of the hex file can be broken down into the following parts:



The “Data” is actually what is stored in memory. The Start Code, Byte Count, Record Type and Checksum are used to direct the receiving CPU on how to handle the Data and to help the CPU determine if the data was received correctly. The address is the location (address) to place the information.

We will use the above latter in the experiment.

Procedure:

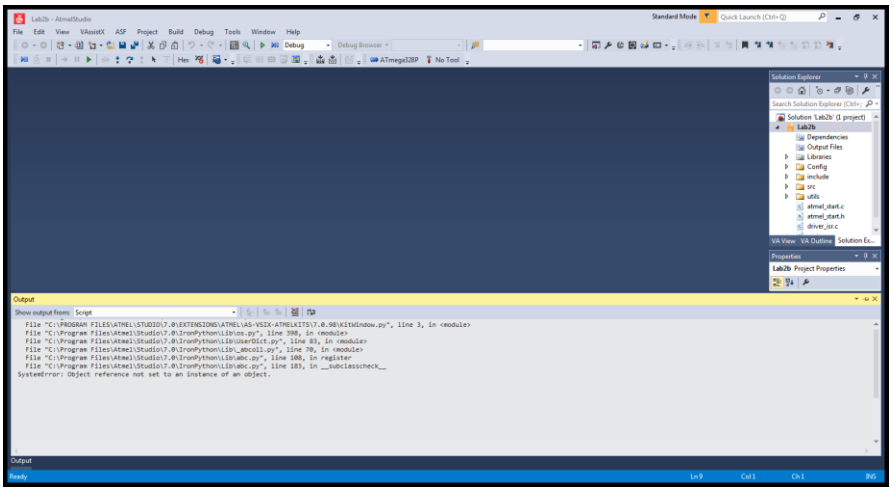
[1] https://en.wikipedia.org/wiki/Intel_HEX

1. Open Atmel Studio 7.0 and do FILE | NEW | Atmel Start Project. This will start a project wizard. Select the IC that is on the Arduino Uno Board (atmega328p-pu), and hit the "Create New Project" button. (you can configure some hardware on this page before hitting the button, but for this lab we do not need to do so).

2. On the next screen we can do more configuration. We do need to change one item, since the IC defaults to 8MHz and the Arduino board uses a 16MHz Crystal. To do this click on the Clock Icon  . On the screen look for the Input Clock box and hit the settings icon within the box  . Now change the value of the "Input Clock Frequency" to 16000000 (16MHz) and click close. Now hit "Generate Project" on the bottom of the window.

3. A screen will pop up. Give the project the name "Lab2". If you wish to save the file to your thumb drive, change the location now by browsing to the directory you wish to use. This will create the basic structure of your project within a directory called Lab2. Within that directory will be a file called Lab2.atsln which is the project file.

4. You should now have the following screen:



   Off to the right you should have a "Solution Explorer" window. If you do not you can click VIEW | Solution Explorer to open it. This shows you all the files associated with the project. The one we are most interested in is within utils and it is called **main.c**. Double click on this to open the file.

5. Now type in the following program (note the basic structure will already have been created by the project wizard):

```
#include <atmel_start.h>
#include <avr/io.h>
```

```
#include <util/delay.h>

int main(void)
{
      /* Initializes MCU, drivers and middleware */
      atmel_start_init();
    DDRB = 0b00100000; // configure pin 5 of PORTB as output
(digital pin 13 on the Arduino Uno)

    while(1)
    {
          PORTB = 0b00100000; // set 5th bit to HIGH
          _delay_ms(1000);
          PORTB = 0b00000000; // set 5th bit to LOW
          _delay_ms(1000);
    }
}
```

[note: "(digital pin…" should be a continuation of the previous line"]

6. Once you are finished typing, you need to compile the C program into something the IC will understand (Machine Language). To do this, go to BUILD | BIULD SOLUTION. The output window should end with a message saying "Build succeeded" if not, check your code.

7. Since the code is written for the Arduino, we must send the code over to that IC before we can test it. In industry, IC's like the Atmega328p-pu are programmed out of circuit using a programmer, but since we are using the ATmega on an Arduino board, we are going to use the USB on the Arduino to send the code to it. To do this we must add an external tool to Atmel Studio[2].

    - First Plug in the Arduino Uno to the USB of the computer.
    - Find the COM port of the device
        - Win10 – right click on Windows Start Icon and select Device Manage and look under Ports (COM & LPT)
        - Win10 (Device Manager Locked) – search for "PowerShell" and open. Type in "Get-WMIObject Win32_SerialPort" (no quotes) and check results for Arduino Uno (Com __)
        - Win 7 – use Start | Control Panel | Device Manager and look under Ports (COM & LPT)

    Note which COM port it is (you will need that info in a bit)

[2] http://www.instructables.com/id/How-to-Load-Programs-to-an-Arduino-UNO-From-Atmel-/

- Back in Atmel Studio, click on Tools | External Tools and click the Add button and fill out the form as follows [NOTE: students did not have Tools | External Tools in their menus for some reason. They had to do Tools | Customize | Commands Tab. Then Select "Menu Bar" Radio button. Then in "Controls" window, select "Tools" and then "Add Command". Select "Tools" from Categories window and then "External Tools…" from commands window then click OK then close. Then use new menu option "External Tools" to complete the following]:

    - Title: Arduino Uno

    - Command: (click on the  icon and browse to C:\Program Files\Arduino\ then use the search box to find avrdude.exe and double click on the found file)

    - Copy the path up to avr\ (highlight and control C).

    - Arguments: type in the following line:

      ```
      -C "[ctrl-v]etc\avrdude.conf" -p atmega328p -c arduino -P
      COM7 -b 115200 -U
      flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i"
      ```

      Replace [ctrl-v] with the path from the previous step. Replace COM7 with COM Port found in step 7b. (It is typed as one continuous line) [note: Watch Spaces – one missing or in the wrong place will cause program to act like code is sent to the board, but in reality it is not sent or not sent correctly]

    - Uncheck "Close on Exit" and check "Use Output Window"

    - Click OK

8. Now go back to TOOLS and there should be a new menu item called Arduino Uno. When you click on this, it should send the code to the Arduino board. If successful, the last line in the output window should be "avrdude.exe done. Thank you."

9. Is your LED by Digital Pin 13 blinking at about 1 second intervals? If NO ask for help!

10. If it is blinking…..congratulations you have written and executed your first embedded C program! **Demo to the instructor**

11. In Atmel Studio, find and print the following files:

    - Lab2/utils/main.c

    - Lab2/Output Files/Lab2.hex

- Lab2/Output Files/Lab2.lss

12. On the printout of the .hex file, mark the start code, byte count, address, record type, data and checksum (highlighters work great for this, although underlines and other marking are acceptable).

13. Lay all three printouts side by side. Find "int main(void)" in the .lss file and have that printout opened to that page (there is a lot of other "overhead" in the .lss file…we are just interested in comparing the main in the c and the machine language).

    In the .lss file, it shows both the C program AND the machine language (shown both in Hex and ASM code). Each line starts with the address, followed by a :" find where the first instruction in main is located in memory…..locate that address in the .HEX file and circle it.

14. As we did in class, prove the following lines of ASM code and their hex equivalent are truly the same thing.

```
    DDRB = 0b00100000; // configure pin 5 of PORTB as output (digital pin 13
on the Arduino Uno)
  8a: 80 e2             ldi   r24, 0x20   ; 32
  8c: 84 b9             out   0x04, r24   ; 4
    while(1)
    {
          PORTB = 0b00100000; // set 5th bit to HIGH
  8e: 85 b9             out   0x05, r24   ; 5
|
| (skipped delay)
|
          PORTB = 0b00000000; // set 5th bit to LOW
  a2: 15 b8             out   0x05, r1    ; 5
```

15. In your own words, how does the C code (shown in grey above) relate to the ASM and to the Machine Language? Why does the LED Blink?

Submit the 3 printouts, and the answers to #14 and #15 to the instructor. Due start of next week's lab.