Lab #4 Introduction to Assembly Programming

Objectives:

- to introduce the basic operation of a decoded I/O port
- to introduce assembly language programming on a microprocessor (microcontroller)
- to introduce the operation of an assembler and other software development tools

Introduction and Background:

The microprocessor we will be using is the Intel 8051which is one of the most widely used microcontrollers in use today. The programming methods you learn here can be used on any microprocessor by adapting to individual instruction sets.

The target machine is the McM51A, a single-board computer based on the 8031 microprocessor and designed here at McMaster. There are many commercial implementations based on this processor and its derivatives. Although very small in comparison to current personal computers, it is a complete system including a single-chip CPU as well as static RAM and non-volatile memory in the form of Flash memory. Communication with the McM51A is via an RS-232C serial port that can be readily used with any terminal emulator running on a PC. This is a good machine for learning the fundamentals of assembly language programming since there is convenient access to individual memory locations, registers and the physical pins of the chips.

Preparation:

In preparation for this lab you should familiarize yourself with the basic architecture of the 8051 microcontroller (see the textbook) as well as the memory map of the implementation in our lab as available on the course web site. Study the code examples given in part 1 and 2 of the lab below. Familiarize yourself with the 8051 instruction set summary before attempting the programming exercises in part 4.

We will be using the 74374 octal D-latch to implement a simple output port that will respond to a “chip select pulse” derived from decoding the address bus. Find the TTl data sheet for the 74374 and make sure that you understand the operation of this circuit. Note that we will not require its output to be 3-state since we are driving LEDs as an output port therefore the 3-state enable input can be tied LO.

Devices used:

McM51A Microcontroller
74374 Octal D-latch

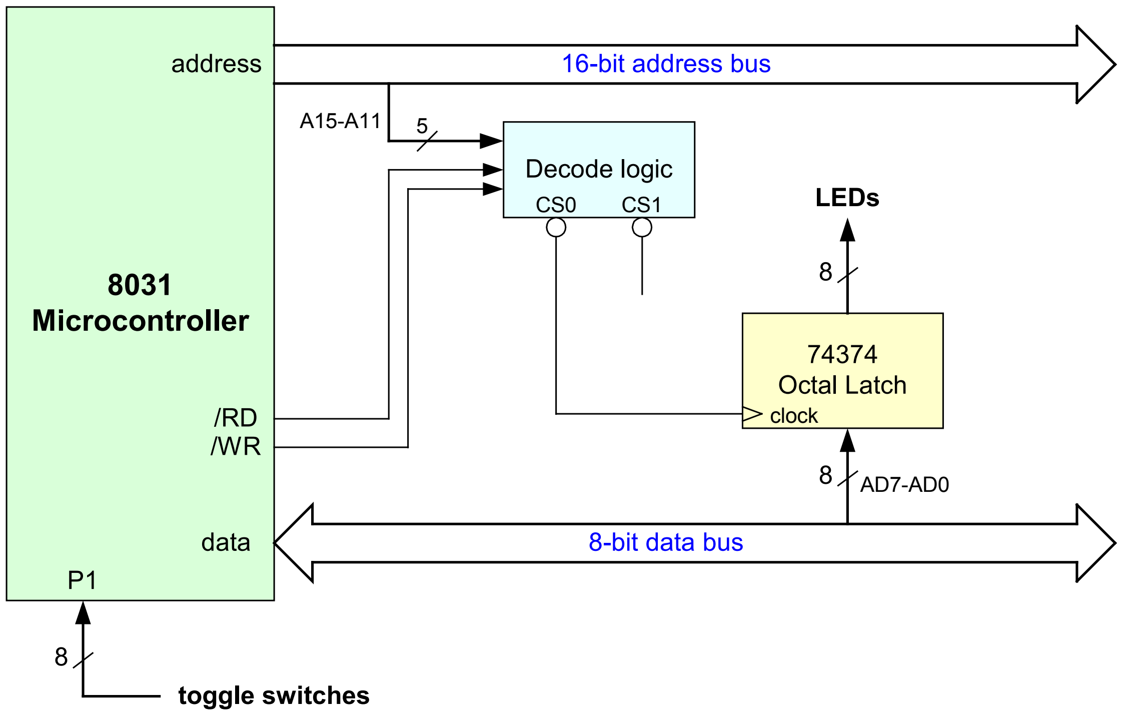
1. Decoded I/O ports

I/O ports on the 8051 microcontroller are “memory mapped”. This means that an I/O device looks like a memory device and occupies space in the memory map. I/O port addresses are decoded as if they are a memory device organized as 1x n (n=8 for the 8051). There are no input or output assembly language instructions in a memory mapped processor. Some processors such as the x86 family (including the Pentium processors) do have special assembly language instructions (IN, OUT) that implement input and output operations. They in fact behave very much like the memory mapped approach, but without occupying locations in the memory space. A separate “I/O map” similar to a memory map is used.

In this lab we implement an output port that drives LEDs with the 74374 octal D-latch that captures data from the 8051 data bus clocked with a chip select CS0 as shown below. Similarly, an input port could be implemented using a chip select pulse to gate inputs onto the 8051 data bus. Any device that implements an input port ***must*** have 3-state outputs since data would be placed ***onto*** the data bus. However, we will implement an input port using one of the 8051 built-in I/O ports (port P1) which is not connected to the data bus as shown below.

Chip selects CS0 and CS1 are generated with decode logic, exactly as memory device chip selects are implemented. The decode logic for CS0 and CS1 as used in this lab is shown in the McM51A memory map document available on the course web site. The decode logic is implemented in an Altera 3032 device that resides on the McM51A board (it also contains the 8-bit latch required to de-multiplex the address and data bus using the ALE signal). CS0 activates (goes LO) in response to a *write* to address 1000H. CS1 pulses LO in response to a *read* from address 1800H.

Connect toggle switches to port P1 and the 74374 latch to LEDs, CS1and 8051 data bus lines (AD7-AD0) as shown below:



2. Assembly language program interface to the I/O port

Example #1

The 8051 code given below continuously generates an input (read) chip select (CS1) followed by an output (write) chip select (CS0).

```
ex1:  mov DPTR, #1800H    ; 1800H is the address of input chip select CS1
      movx A, @DPTR       ; generate CS1 (input external data to reg A)
      mov DPTR, #1000H    ; 1000H is the address of output chip select CS0
      movx @DPTR, A       ; generate CS0 (output reg A to external I/O device)
      ljmp ex1            ; jump back and repeat forever
```

(a) Enter the 8051 assembly language program given above, assemble it and download to the MCM51A target machine. Start the code executing and connect the logic analyser to CS0 and CS1 and verify the positions of these 2 pulses.

(b) Calculate the execution time of one loop of the program above: ______________

(c) Measure the time for one loop of the program above using the 'scope: _____________

(d) Since the decode logic uses A15-A11 from the address bus only, bits A10-A0 are don't cares. Therefore any 16-bit value of the format 1000 0xxx xxxx xxxx generates a pulse on CS0 and any 16-bit value of the format 1000 1xxx xxxx xxxx generates a pulse on CS1. Vary the code above to try several values in register DPTR to verify this.

Example #2

The 8051 microcontroller contains built-in I/O ports (this is a common characteristic of microcontrollers which are meant to be single-chip implementations). One of these ports (**P1**) is available on the McM51A and can be used for input or output. Enter the code as follows:

```
      mov DPTR, #1000H    ; 1000H is the address of output chip select CS0
echo: mov A, P1           ; read the value of switches from port P1 (SFR 90H) into A
      movx @DPTR, A       ; write the toggle switch values to port controlled with CS1
      ljmp echo           ; jump back and repeat forever
```

Assemble, download and run this code. With the 8 input switches on **P17-P10**, verify its operation.

(a) Calculate the execution time of one loop of the program above: ______________

(b) Measure the time for one loop of the program above using the 'scope: ______________

The built-in I/O port uses a fixed address (90H) that we do not need to specify as we did for the output port - its value is assumed in the execution of the instruction **mov A, P1**. Port P1 is one of several "special function registers" (SFRs) that are used in the 8051 architecture. SFRs are used as the basic programming interface to ports, timers/counters etc.

(c) Using the logic analyzer, verify the value that appears on the 8031 data bus when CS1 is asserted while this code is executing.

Example #3

Using the 8-bit output port, we can implement an 8-bit binary counter in 8051 register A as follows:

```
         mov DPTR, #1000h     ; address for CS0
         clr a                ; start a count value at 0 in Register A
repeat:  movx @DPTR, A        ; output the count value to the 8-bit latch (CS0)
         inc a                ; increment our counter (register A)
         ljmp repeat          ; repeat forever
```

Assemble, download and run this code.

(a) Calculate the execution time of one loop of the program above: ______________

(b) Measure the time for one loop of the program above using the 'scope: ______________

(c) Using the logic analyzer, verify the count value that appears on the output port.

3. Timed, 8-bit binary counter in assembly language

A slower counter can be implemented by introducing a delay within the loop. One way to do this is with "nested" program loops as shown below:

```
         mov DPTR, #1000h      ; address for CS0
         clr a                 ; start a count value at 0
repeat:  movx @DPTR, A         ; output the count value to the 8-bit latch (CS0)

         mov r7, #2CH          ;  delay of approx 1 sec
loop1:   mov r6, #2CH          ;  using 3 nested loops
loop2:   mov r5, #0FFH
loop3:   djnz r5, loop3
         djnz r6, loop2
         djnz r7, loop1

         inc a             ; increment our counter (register A)
         ljmp repeat       ; repeat forever
```

The initial values loaded into R5, R6 and R7 have been pre-calculated to provide a delay of approximately 1 second.

Assemble, download and run this code and verify that its counting period is approximately 1 second.

What is the minimum count period that can be attained with this code? ___________

What is the maximum counting period that can be attained with this code? ___________

4. Programming exercises

Starting with the 8051 assembly language code in Part 3, implement each of the following functions:

(a) a binary count with a period of approximately 0.5 second on the LEDs.

(b) modify your code from part (a) so that toggle switch 0, when HI, clears the count.

(c) modify your code from part (d) so that toggle switch 1 determines the count direction.

(d) modify your code to display a BCD count with a period of approximately 0.5 second.

(e) modify your code to display a left-to-right scan of a single HI bit on the 8-bit output port.