SYSC3601 LABORATORY #3 (Summer 2009)

Interrupts

Fill in the blanks on the answer sheets provided by the TAs (one per group). The T.A. will ask you questions about your work. *You will have to demo your work to the T.A. Be prepared to submit your program listing at the end of the lab for marking*.

Resources for this lab

- Download the software package for lab3 (Lab3Materials.zip). Recall that you need ML.exe, LINK.exe, etc. to be in the same directory as your new ASM source files. The zip file contains:
  - LAB3A.ASM – Code for use with Part A below.
  - LAB3B.ASM – Code for use with Part B below.

- The **SDK-86 (MCS-86 System Design Kit) User's Guide** *(available in the lab)*

- Appendix B of the **Brey text** provides descriptions of the 8086 instruction set including instruction timing information. You may want to especially look at the LEA, XLAT, STI, and CLI instructions.

- 1990 Microprocessors Handbook, Interrupt Operations *(available in the lab)*

- The Orange Book (Peripherals Manual) for description and programming of:
  - Programmable Peripheral Interface – PPI – **8255A**
  - Programmable Interval Timer – PIT – **8253**
  - Programmable Interrupt Controller – PIC – **8259A**
    - **Be aware that portions of the 8259A documentation are specific to the MCS-80/85 and some portions are specific to the 8088/8086!**
  - Programmable Keyboard/Display Interface – **8279** (*focus on display functionality*)

- **Diagram of SDK system extensions** with PIC/PIT connections (available on the course website)

Learning Objectives

- Understand Interrupts and their use in a microprocessor-based system
  - What is the sequence of operations that occur during an interrupt?
  - What are the signals that appear on the bus during an interrupt cycle?
  - What is the Interrupt Vector Table and how is it used?
- Understand the Programmable Interrupt Controller (8259A) and its interface to the 8086 on the SDK
  - What are the key signals that connect the 8259A to the 8086?
  - What is the Interrupt Mask Register and how is it used?
  - How are interrupt priorities handled using the 8259A?
  - What are ICW1, ICW2, ICW4, OCW1, OCW2?
- Understand the Programmable Interval Timer (8253)
  - What is the 8253, and how is it used?
  - How is the 8253 connected to the 8086 and the 8259 (review Appendix B)
  - How are the counters initialized and loaded?
  - What possible clock sources exist (review SDK extensions diagram from website)
- Understand the Keyboard Display Controller (8279)
  - How do you write to display RAM using the 8279?
  - How can you address the 7-segment LEDs on the SDK-86?

Introduction to the lab

The SDK board is connected in such a way that a timer chip (8253) is connected to a programmable interrupt controller (PIC) (8259A), and the PIC is connected to the 8086 microprocessor. The timer produces waveforms to trigger the PIC. When the PIC receives interrupt requests, it ranks the requests according to their priorities and services them accordingly. The PIC will send a signal to the INTR pin of the 8086. Eventually the 8086 responds with an interrupt acknowledge signal (INTA). The PIC will then send the interrupt vector to the 8086, and the 8086 confirms with another INTA and executes the appropriate interrupt service routine (ISR) determined by the interrupt vector. Before the 8086 executes the ISR, it saves the flags, CS and IP registers. Once the ISR has been executed, the 8086 resumes its original task via the IRET instruction, which also restores the flags.

PART A: SDK-86 Interrupt Timing Diagrams

Check that the following jumpers are in place: W40 (2.45 MHz), W27 (zero wait states). Check that the following jumper is NOT in place: W36 (interrupt disable). On the website, you will find LAB3A.ASM in Lab3Materials.zip. You have been expected to get a copy of it to study before the lab.

The LAB3A.ASM program initializes the SDK-86 hardware to generate and allow interrupts. The main program is an infinite loop. The interrupt service routines do nothing except return (IRET) from the interrupt. Some of the chips that have been initialized are on the Carleton extension of the SDK-86 board. On the actual board you can see this extension as a column of chips along the left side of the board. See the diagram of the SDK extensions on the course website. It contains two 32K byte RAM chips, an A/D converter, a D/A converter, a timer chip (8253) and an interrupt controller (8259A). The timer and interrupt controller are of concern in Part A of this lab.

1. According to the program and peripherals manual, what has the timer chip (8253) been programmed to do?

   ___A1_____

2. What has the peripheral interrupt controller (8259A) been programmed to do?

   ___A2_____

3. Assemble/link and download LAB3A.ASM to the SDK-86 and run it from the serial monitor. (For the details of this procedure see Lab #1)
4. Using the logic analyzer, observe the signals INTR and $\overline{\textbf{INTA}}$ to complete the A3 timing diagram. The logic analyzer has been set to trigger on IR1.

5. Describe what is happening (i.e. what is the $\overline{\text{INTA}}$ signal doing relative to the IR0 & IR1 signals?):

   ___A4_____

6. Why is IR2 not included in the timing diagram? (i.e. why does it have no effect on INTR?)

   ___A5_____

PART B: PROGRAMMING THE TIMER, INTERRUPT CONTROLLER, AND DISPLAY CHIP

The following is a description of how the program in `LAB3B.ASM` works:

The program initializes hardware for interrupts and then enters an infinite loop and waits for interrupts. The interrupt service routine for `IR0` will output a value to the D/A converter to generate a sawtooth waveform. The interrupt service routine for `IR1` will output a **garbage** character to the rightmost digit of the `SDK-86` LED display.

1. Assemble/link and download LAB3B.ASM to confirm that the program works. Use the oscilloscope to check that a sawtooth waveform is being produced by the D/A converter.

2. What has the display chip (8279) been programmed to do?

   ___B1_____

3. Now that you have confirmed that LAB3B.ASM is generating interrupts as you would expect, run your `LAB3C.ASM` program (from your prelab), which has added a third interrupt service routine. Again, this third ISR (ISR2) should be invoked every 0.5 seconds. To do this, you have reprogrammed the timer chip to generate a 0.5 second signal on pin `OUT2` (note that `OUT2` is connected to `IR2` on the `PIC`). The `PIC` and `8086` have also been reprogrammed, and remember that ISR2 runs at a higher priority than ISR0 and ISR1. ISR2 should display a digit in the leftmost LED display of the `SDK-86` which continually counts down from `A-0`.

   Thus when your program is run, you should see a sawtooth waveform (from ISR0), a garbage character being updated once per second in the rightmost LED (from ISR1), and a hex (`A–0`) digit decrementing every 0.5 seconds displayed in the leftmost display digit (from ISR2).

4. In your new program, what have you programmed the timer chip (`8253`) to do in addition to the previous settings?

   ___B2_____

5. What have you programmed the peripheral interrupt controller (`8259A`) to do in addition to the previous settings? Give the new control words.

   ___B3_____

6. What have you programmed the display chip (`8279`) to do?

   ___B4_____

7. In your program, which ISR has the lowest priority?

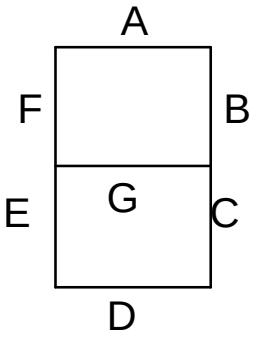
   ___B5_____

8. Write down the new 8253-related values that you would use to double the sawtooth frequency without changing the frequencies of the two SDK LED display updates. Note that you are *not required* to create a new program, just write down what you would do. Pay attention to the fact that ISR1 and ISR2 are related to `OUT0`…

   ___B6_____

Bonus:

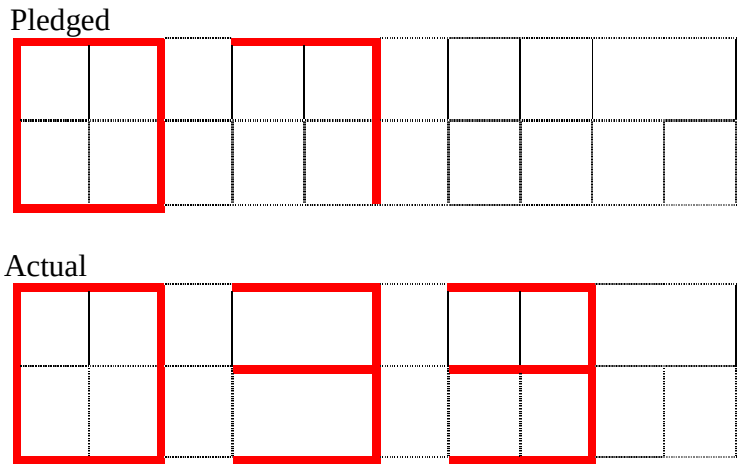
Let's have some more fun with the LED display! Using the following 7 digit translation table value encoding:



0	G	F	E	D	C	B	A
MSB							LSB

For example, the character 'A' is 01110111 = 77H

Bonus 1: Modify ISR2 to show the percentage of Canada's GDP is has pledged to foreign aid (0.7%), and the actual amount it is currently giving (0.33%). Cycle between these every 0.5 seconds.



Bonus 2: Modify ISR2 to display an arbitrary string scrolling across the display, shifting one character every 0.25 seconds. The string should repeat once the entire string has been displayed and the screen has emptied.