LABORATORY #3

Interrupts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Student#1 | Student # 2  | TA Initials | Date |
| Student Name |  |  |  |  |
| Student Number |  |  |  |  |
| Prelab /Attendance |  |  |  |  |
| Lab # 3 Completion |  |  |  |  |

**HOME PREPARATION:**

1. **EXTREMELY IMPORTANT NOTE**: Go to the lab in the week before your scheduled lab period and take copies of LAB3A.ASM and LAB3B.ASM. Print out these two programs while you are in the lab. There are office hours posted on the door of the lab. You cannot do your home preparation without these programs.

**Review**: Appendix A, logic analyzer operation

 Appendix B, in particular the timer and PIC connections,

**Read**: 8086/8088 User’s Manual, Chapter 4, Interrupt Structure section

 1990 Microprocessor Handbook, Interrupt Operations

 Peripherals Manual for Timer (8253)

 Peripheral Interrupt Controller (8259A)

 Keyboard/Display Controller (8255A).

1. This lab has two parts. From the reading, understand interrupt structure of the SDK-86; you will need this in both parts. For part B, write the assembler code before coming to the lab.
2. Note that many blanks on these sheets can actually be filled in before you come to the lab. The programs you get from the lab have spaces for comments. You must fill in these blanks before handing in the lab. Bring in a 3.5” diskette.

**LAB # 3: INSTRUCTIONS**:

Fill in the remaining blanks on these question sheets (One per group). These sheets constitute your lab report. 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 these lab sheets and your program listing at the end of the lab for making.

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 request 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, then executes the appropriate interrupt service routine (ISR) determined by the interrupt vector. Before the 8086 executes ISR, it saves the flag, 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 (2.45MHz), W27 (zero wait states).Check that the following jumpers is not in place: W36 (interrupt disable). In the 361/461 directory you will find LAB3A.ASM. You have been expected to have a copy of it and study it the week 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 a return from interrupt. Some of the chips that have been initialized are on the Carleton extension of the SDK-86 board. On the actual board you have seen this extension as a column of chips at the left. At the end of the lab is a schematic diagram of the Carleton extension of the SDK-86 board. 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?
2. What has the peripheral interrupt controller (8259A) been programmed to do?
3. Assemble, link and download LAB3A.HEX 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 INTA to complete the following timing diagram. Use the logic analyzer to trigger on IR1 and sketch INTR and INTA. (See appendix A for a listing of the Logic Analyzer/CPU pins).
5. Complete the following timing diagram:

1. Describe what is happening:

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

You are expected to get a copy of LAB3B.ASM and study it in the week before your scheduled lab period. Here is a description of how it works.

The program initializes hardware for interrupt 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 saw tooth waveform. The ISR will output a GARBAGE character to the rightmost digit of the SDK-86 LED display.

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. What has the peripheral interrupt controller (8259A) been programmed to do? (Is this different from part A? \_\_\_\_\_\_\_\_\_\_\_\_ )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. What has the display chip (8279) been programmed to do? (Is this different from part A? \_\_\_\_\_\_\_\_\_\_\_\_\_ )

5. Now that you have confirmed that LAB3B.ASM is generating interrupts as you would expect, you will have to change the program. Your task is to add a third ISR called ISR2 that does NOT interfere with the existing ISRs. ISR2 should be invoked every 0.5 second. To do this you will have to reprogram the timer chip to generate a 0.5 sec period signal on pin OUTW. Note that OUT2 is connected to IR2 on the PIC. You also have to reprogram the PIC and the 8086. You are required to have ISR2 run at high priority than ISR1 and ISR0. ISR2 should display a digit in the leftmost LED display of the SDK-86 which continually counts down from 9 t0 0. Refer to LAB1D.ASM for the binary to seven-segment digit translation code required by the 8279.

Thus when your program is done, you should see a saw tooth waveform (from ISR0), a garbage character being updated every second in the rightmost LED (from ISR1) and a downward count every o.5 second displayed in the leftmost digit(from ISR2).

6. In your program, what have you programmed the timer chip (8253) to do?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. What have you programmed the peripheral Interrupt controller (8259A) to do?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. What have you programmed the display chip (8279 to do?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. In your program, what ISR has the lowest priority? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. Write down the new 8253 control words that you would use to double the saw tooth frequency without changing the frequencies of the two SDK LED display updates. Note you are not required to download a new program; just write down what you would do.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_