Carter Nettesheim
ECE5780
HW4

1 Hardware interrupts are asynchronous because they can occur at any time while the program is running
Software interrupts are synchronous because they are "traps" that occur at known times while the program is running

2 Using polling takes a lot of energy and processor power. Essentially all the processor does is constantly look for an interrupt. However interrupts allow the processor to do other things (including sleep) until the interrupt is triggered and starts a new process on its own.

3 When a non-vectored interrupt is triggered the current position in the program (PCR) is stored and the program jumps
to the interrupt handler. The program then needs to figure out who triggered the interrupt. The interrupt is handled and the code attached to the interrupt is executed. Then the PCR is restored and the program returns to its original position and continues to execute.

4 It's important to push the the current context onto the interrupt stack so that whatever was originally executing can return to it's normal execution without any errors being introduced to the system. It's just as important to return to the normally executed code exactly how you left it, as it is to handle the interrupt.

5 The interrupt mask register (IMR) can be used to enable and disable interrupts.

6 Interrupt nesting allows an interrupt to be interrupted by another interrupt. For example, an interupt being handled by the OS could be interrupted by a interrupt triggered on the BIOS. There can be problems with this process because
some interupts may not meet their set deadlines because of another interrupt.

7 Level sensitive device drives the signal to a level, and then holds the signal at that level until its been cleared
Edge triggered device drives a level transition, falling or rising. This allows multiple interupts on the same line

8 The processor gets the vector number of the current interrupt from the interrupt vector table. The interrupt vector table will the point the processor to the correct starting address of the target ISR.

9 A software interrupt is a "trap" in the code where it stops the current code running to wait for a process. The most common software interrupt is waiting for a character or line to be inputted from the keyboard during a program.
Sofware interrupts are important in todays computers because the OS uses software interrupts in the BIOS in order to
communicate and requrest processes from the computer's hardware.

10 A processor is coded to meet certain deadlines. If interrupts are introduced, they can skew the processors ability to meet its timing constrains. In order for processor to meets its timing constraints the interrupts is moved to a server task to be handled so that the program can return to the normal execution and meet timing constraints.

13 Sometimes if data is shared between normal programming and interrupts, the data can change during the normal programming causing the system to have incorrect data when the process returns to normal programming. Interrupt disabling fixes this problem by disabling the interrupts during the overlap of data sharing in normal programming. This allows the data to be paused and played so that the "correct" data isn't changed during a normal programming process. Double buffering solves this problem by creating a "read" register and a "write" register. This allows the program to always have the "previous" correct values while allowing the interupt to continue to pick up and change new incoming data.

14 Interrupt latency can be caused by interrupt processes being too long, or if a higher priority IRQ is already running.
Scheduling delay can be caused by the amount of time is takes to save the processors current spot and jump to the correct interrupt, or if a higher priority IRQ is already running.

15 256 interrupt vectors