CIS 314
Memory Mapped I/O and Interrupt Handling

Prof. Michel A. Kinsky
Our fourth view of computer organization: The modern digital computer has three major functional hardware units: CPU, Main Memory and Input/Output (I/O) Units.
Input/Output Devices

- For the computer system to interact with the real world we need input/output functionalities.
For the computer system to interact with the real world we need input/output functionalities.
I/O Device Interfaces

• An I/O device interface or controller is a hardware module that connects a device to the processing elements and memory units
  ‣ Provides the means for data transfer and exchange of status and control information between the device and the rest of the computer system
  ‣ Information includes data, status, and control registers
I/O Device Interfaces

• An I/O device interface or controller is a hardware module that connect a device to the processing elements and memory units
  ‣ Assume that the I/O devices have a way to send a *READY* signal to the processor
    • For keyboard, it indicates that the character is ready to be read
    • For display, it indicates the display is ready to receive the character
  ‣ The *READY* signal in each case is a status flag in status register that is polled by processor
Memory-Mapped I/O

- I/O devices are assigned memory locations
- Interactions with the I/O device is done through memory load and store operations
Program Controlled I/O

• There is a program that is implemented and loaded to perform all of the relevant functions related to the I/O device operation.
Interrupt Controlled I/O

• With programmed I/O, processor needs to poll the I/O ready signal and while waiting cannot do any useful work

• With interrupt controlled I/O, the I/O device sends an interrupt-request signal to the processor when ready and the processor stops what it is doing to handle the interrupt
Exceptions and Interrupts

- Exceptions and interrupts handling in processors
  - Exceptions and interrupts are computing events not part of the main program execution flow that temporally divert the processor's
  - Interrupt requests often come from I/O devices
  - Exceptions are often program execution induced event (e.g., divide by zero, stack overflow, wrong memory address)
  - They are both handled through special subroutines
Exceptions and Interrupts

- Exceptions and interrupts handling in processors
  - For example, assume a PRINT interrupt signal is asserted when processor is executing instruction $i$
  - The processor performs the following actions:
    1. Completes instruction $i$
    2. Saves PC and other system states to temporary location
    3. Jump to the PRINT subroutine
    4. Executes the PRINT subroutine
    5. Disables the interrupt signal upon finishing the PRINT execution
    6. Restores PC with address of instruction $i + 1$
    7. Resumes normal program
Next Class

• Parallel Architectures