Overview

- Interrupt System Specifications
- Multiple Sources of Interrupts
- Interrupt Priorities
- Polling

Five Components of any Computer

How CPU Interacts with I/O?

Two Choices:
- Interrupts.
  - I/O devices generate signals to request services from CPU.
  - Need special hardware to implement interrupts.
  - Efficient.
    - A signal is generated only if the I/O device needs services from CPU.
- Polling
  - Software queries I/O devices.
  - No hardware needed.
  - Not efficient.
    - CPU may waste processor cycles to query a device even if it does not need any service.
Interrupt System Specifications (Cont.)

1. Allow for synchronous events to occur and be recognized.
2. Wait for the current instruction to finish before taking care of any interrupt.
3. Branch to the correct interrupt service routine (interrupt handler) to servicing interrupting device.
4. Return to the interrupted program at the point it was interrupted.
5. Allow for a variety of interrupting signals, including levels and edges.
6. Signal the interrupting device with an acknowledge signal when the interrupt has been recognized.

7. Allow programmers to selectively enable and disable all interrupts.
8. Allow programmers to enable and disable selected interrupts.
9. Disable further interrupts while the first is being serviced
10. Deal with multiple sources of interrupts.
11. Deal with multiple, simultaneous interrupts.

Interrupt Recognition and Ack

- An Interrupt Request (IRQ) may occur at any time.
  - It may have rising or falling edges or high or low levels.
  - Frequently it is a active-low signal and multiple devices are wire-ORed together.
- Signal Conditioning Circuit detects these different types of signals.
- Interrupt Request Flip-Flop (IRQ-FF) remembers that an interrupt request has been generated until it is acknowledged.
  - When IRQ-FF is set, it generates a pending interrupt signal that goes towards the Sequence Controller.
  - IRQ-FF is reset when CPU acknowledges the interrupt with INTA signal.
Interrupt Recognition and Ack (Cont.)

- The programmer has control over interrupting process by enabling and disabling interrupts with explicit instructions
  - The hardware that allows this is Interrupt Enable Flip-Flop (INTE-FF).
  - When the INTE-FF is set, all interrupts are enabled and the pending interrupt interrupt is allowed through the AND gate to the sequence controller.
  - The INTE-FF is reset in the following cases.
    - CPU acknowledges the interrupt.
    - CPU is reset.
    - Disable interrupt instruction is executed.

Interrupt Recognition and Ack (Cont.)

- An interrupt acknowledge signal is generated by the CPU when the current instruction has finished execution and CPU has detected the IRQ.
  - This resets the IRQ-FF and INTE-FF and signals the interrupting device that CPU is ready to execute the interrupting device routine.
- At the end of the interrupt service routine, CPU executes a return-from-interrupt instruction.
  - Part of this instruction’s job is to set the INTE-FF to reenable interrupts.
  - If the IRQ-FF is set during an interrupt service routine a pending interrupt, there is one, will be recognized by the sequence controller immediately after the INTE-FF is set. This allows nested interrupts i.e. interrupts interrupting interrupts.

Multiple Sources of Interrupts

- Determine which of the multiple devices has generated the IRQ to be able to execute its interrupt service routine.
  - Two approaches: Polled interrupts and vectored interrupts.
- Resolve simultaneous requests from interrupts with a prioritization scheme.

Polled Interrupts

- Software, instead of hardware, is responsible for determining the interrupting device.
  - The device must have logic to generate the IRQ signal and to set an “I did it” bit a status register that is read by CPU.
  - The bit is reset after the register has been read.
- IRQ signals the sequence controller to start executing an interrupt service routine that first polls the device then branches to the correct service routine.
**Polled Interrupt Logic**

- Logic to generate IRQ
- Logic to reset IRQ when status register is read
- Logic to read status register and reset “I did it” bit
- Logic to set “I did it” bit
- Status register

**Vectored Interrupts (I)**

- CPU’s response to IRQ is to assert INTA.
- The interrupting device uses INTA to place information that identifies itself, called vector, onto the data bus for CPU to read.
  - An vector is the address of an interrupt service routine.
- CPU uses the vector to execute the interrupt service routine.

**Vectored Interrupting Device Hardware (I)**

- Logic to generate IRQ
- Logic to reset IRQ
- Vector Information
- Three-State Driver

**Vector Interrupts (II)**

- CPU has multiple IRQ input pins.
- CPU designers reserve specific memory locations for a vector associated with each IRQ line.
- Individual disable/enable bit is assigned to each interrupting source.
**Interrupt Priorities**

- When multiple interrupts occur at the same time, which one will be serviced first?
- Two resolution approaches:
  - Software resolution.
    - Polling software determines which interrupting source is serviced first.
  - Hardware resolution.
    - Daisy chain.
    - Separate IRQ lines.
    - Hierarchical prioritization.
    - Nonmaskable interrupts.

**Daisy Chain Priority Resolution**

- CPU asserts INTA that is passed down the chain from device to device. The higher-priority device is closer to CPU.
- When the INTA reaches the device that generated the IRQ, that device puts its vector on the data bus and not passing along the INTA. So lower-priority devices do NOT receive the INTA.

**Daisy Chain Priority Resolution (Cont.)**

**Hardware Priority Resolution**

- Separate IRQ Lines.
  - Each IRQ line is assigned a fixed priority. For example, IRQ0 has higher priority than IRQ1 and IRQ1 has higher priority than IRQ2 and so on.
- Hierarchical Prioritization.
  - Higher priority interrupts are allowed while lower ones are masked.
- Nonmaskable Interrupts.
  - Cannot be disabled.
  - Used for important events such as power failure.
Transferring Control to Interrupt Service Routine

- Hardware needs to save the return address.
  - Most processors save the return on the stack.
  - ARM uses a special register, link register, to store the return address.
- Hardware may also save some registers such as program status register.
  - AVR does not save any register. It is programmer’s responsibility to save program status register and conflict registers.
- The delay from the time the IRQ is generated by the interrupting device to the time the Interrupt Service Routine (ISR) starts to execute is called interrupt latency.

Interrupt Service Routine

- A sequence of code to be executed when the corresponding interrupt is responded by CPU.
- Consists of three parts: Prologue, Body and Epilogue.
- Prologue:
  - Code for saving conflict registers on the stack.
- Body:
  - Code for doing the required task.
- Epilogue:
  - Code for restoring all saved registers from the stack.
  - The last instruction is the return-from-interrupt instruction.
    - iret in AVR.

Software Interrupt

- Software interrupt is the interrupt generated by software without a hardware-generated-IRQ.
- Software interrupt is typically used to implement system calls in OS.
- Most processors provide a special machine instruction to generate software interrupt.
  - SWI in ARM.
- AVR does NOT provide a software interrupt instruction.
  - Programmers can use External Interrupts to implement software interrupts.

Exceptions

- Abnormalities that occur during the normal operation of the processor.
  - Examples are internal bus error, memory access error and attempts to execute illegal instructions.
- Some processors handle exceptions in the same way as interrupts.
  - AVR does not handle exceptions.
Reset

- Reset is a type of interrupt in most processors (including AVR).
- It is a signal asserted on a separate pin.
- Nonmaskable.
- It does not do other interrupt processes, such as saving conflict registers. It initializes the system to some initial state.

Reading

2. Interrupts. Mega64 Data Sheet.