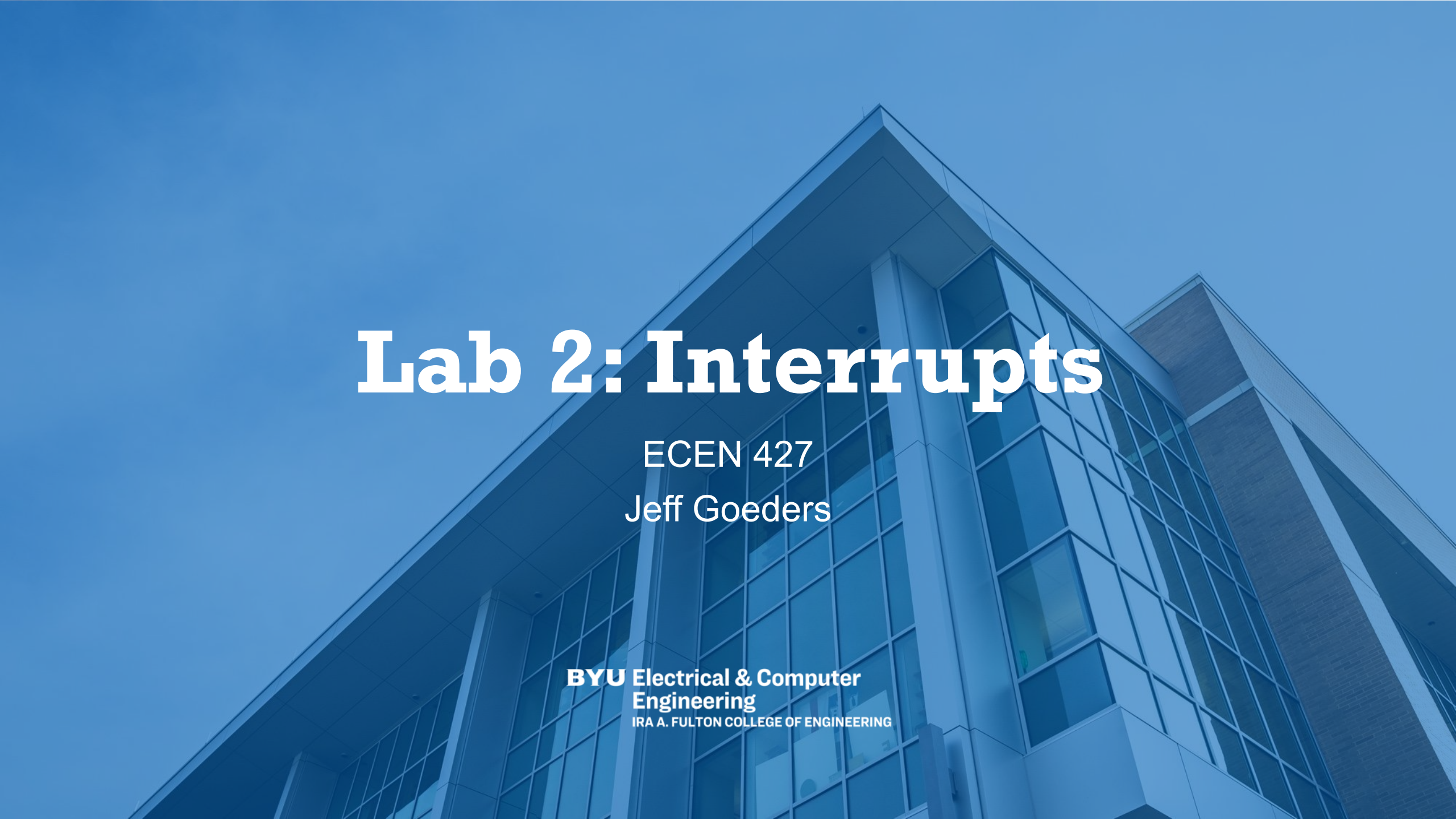


Lab 2: Interrupts

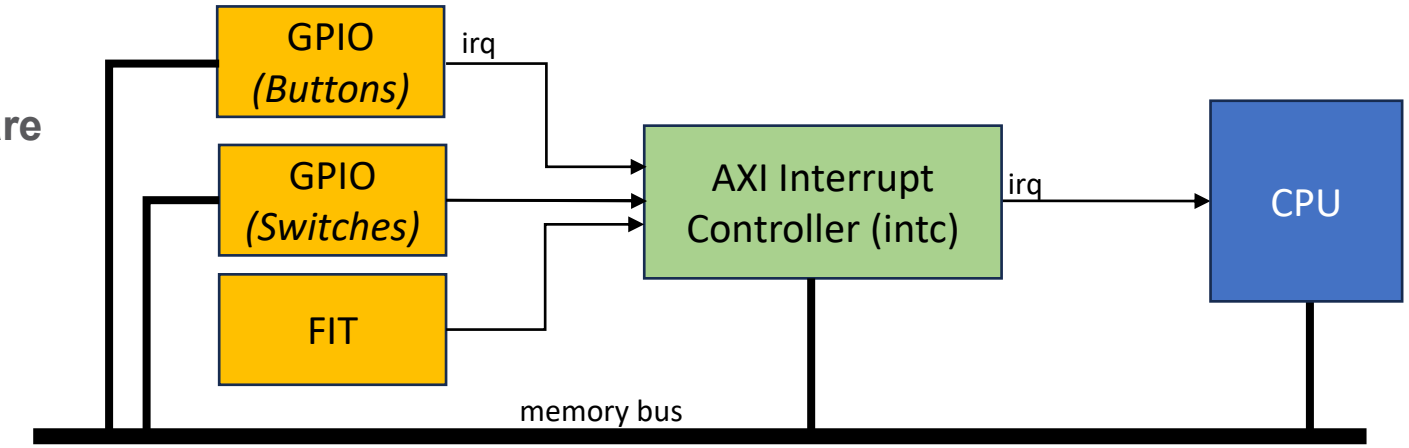
ECEN 427

Jeff Goeders

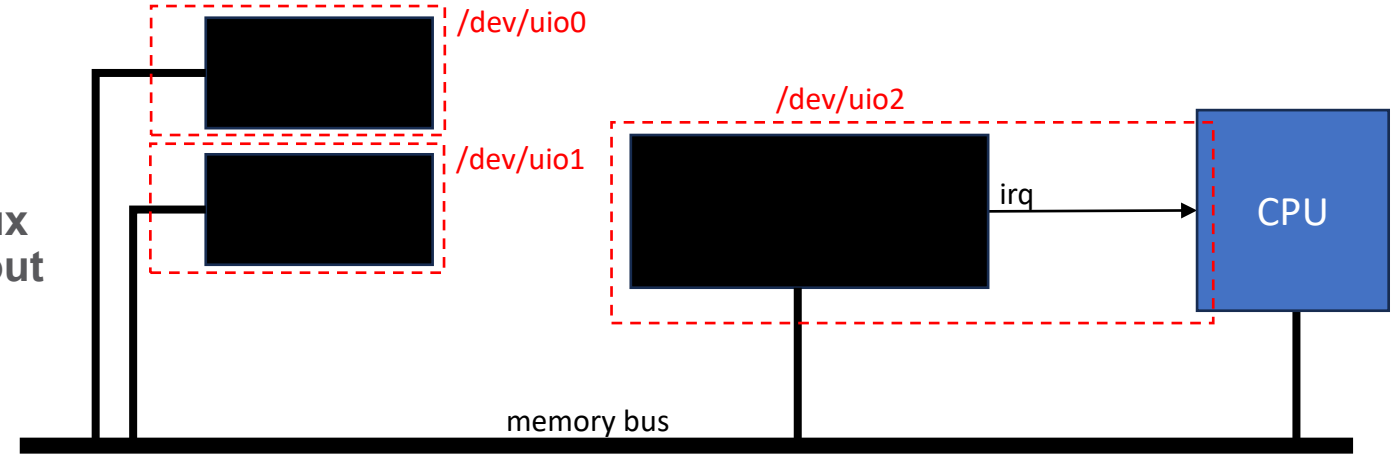
BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

The Lab 2 HW System

Our hardware system



What Linux knows about



- The focus of Lab 2 M2 is to write a driver for the AXI Interrupt Controller
- One you have your GPIO and intc drivers working, you can test them with provided test applications.

Interrupt Systems

There are many pieces to building a correctly functioning interrupt system:

1. **Initializing the system**. This includes enabling interrupt handing in the individual hardware devices, and registering an interrupt handler.

2. **Responding to an interrupt**. When the system receives an interrupt, there is usually work to be done. This is typically device and application dependent.

3. **Acknowledging the interrupt**. The hardware may continue sending the interrupt until we explicitly acknowledge it.

Interrupt Systems

There are many pieces to building a correctly functioning interrupt system:

1. **Initializing the system**. This includes enabling interrupt handing in the individual hardware devices, and registering an interrupt handler.

2. **Responding to an interrupt**. When the system receives an interrupt, there is usually work to be done. This is typically device and application dependent.

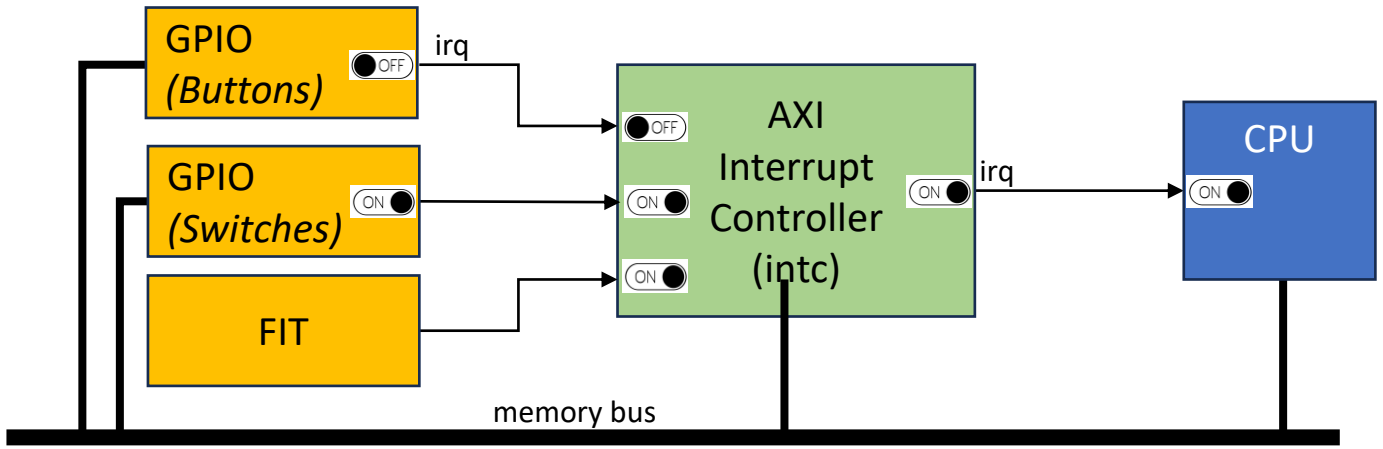
3. **Acknowledging the interrupt**. The hardware may continue sending the interrupt until we explicitly acknowledge it.

Setting up Interrupts

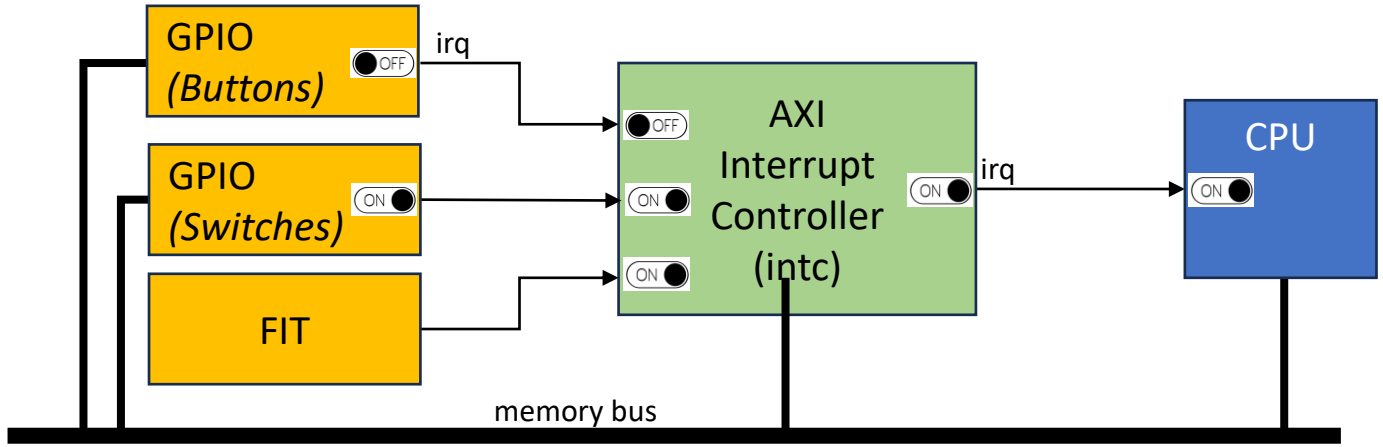
In most systems, enabling the interrupt system is a multi-step process.

1. **Registering a handler function (ISR).** This decides what code is trigger on an interrupt.
   - For our system, the UIO driver is taking care of this and has an ISR registered with the kernel.
   - *Recall:* This ISR will unblock our process that blocked on a read() syscall.

2. **Enabling IRQ lines.** Our system requires many points to be enabled before an interrupt will be able to be transmitted from input event to CPU:

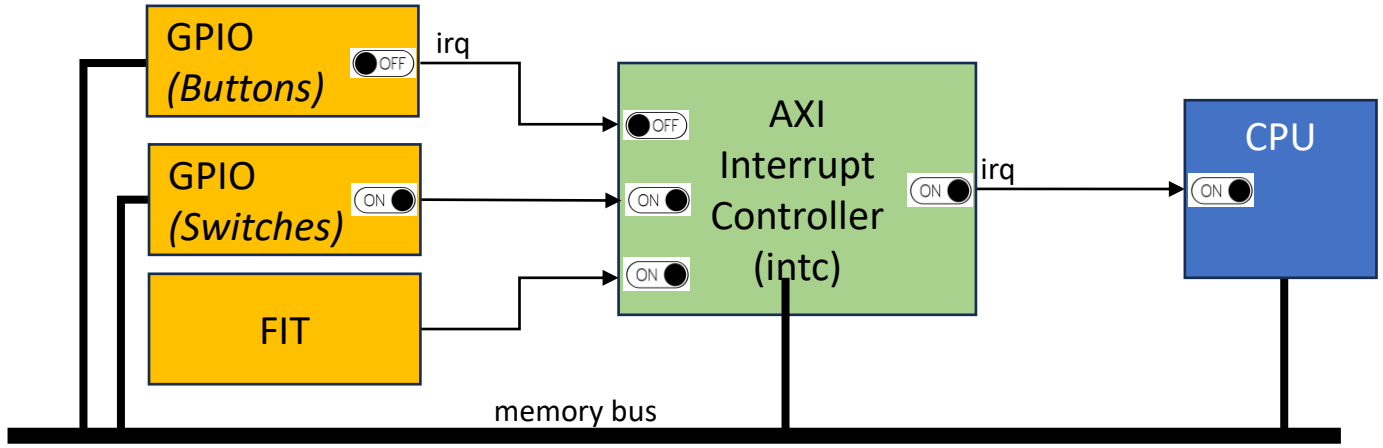


Setting up Interrupts



What needs enabling?	What code does this?	How?

Setting up Interrupts



What needs enabling?	What code does this?	How?
GPIO must be configured to send an interrupt	`buttons_enable_interrupts()`	Configure GPIO registers
Interrupt controller IRQ inputs must be enabled	`intc_irq_enable()`	Configure Intc registers
Interrupt IRQ output must be enabled	`intc_init()`	Configure Intc registers
Interrupts must be enabled for this CPU input in the UIO driver.	`intc_init()`. Plus, anytime an interrupt occurs and you handle it, you must re-enable (`intc_ack_interrupt`)	write() syscall

… let’s look at the interrupt controller documentation…

Interrupt Systems

There are many pieces to building a correctly functioning interrupt system:

1. **Initializing the system**. This includes enabling interrupt handing in the individual hardware devices, and registering an interrupt handler.

2. **Responding to an interrupt**. When the system receives an interrupt, there is usually work to be done. This is typically device and application dependent.

3. **Acknowledging the interrupt**. the interrupt until we explicitly a

Example: In Lab 3, we will use FIT and button/switch interrupts to build a simple clock application.

- When user presses a button, change the time
- When FIT fires N times, increment time by 1 second

Interrupt Systems

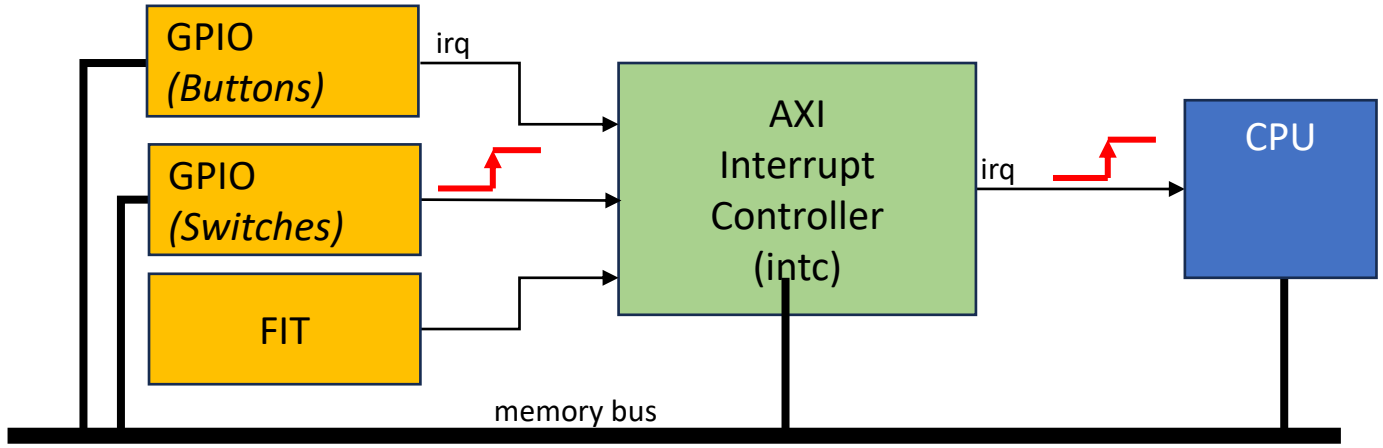
There are many pieces to building a correctly functioning interrupt system:

1. **Initializing the system**. This includes enabling interrupt handing in the individual hardware devices, and registering an interrupt handler.

2. **Responding to an interrupt**. When the system receives an interrupt, there is usually work to be done. This is typically device and application dependent.

3. **Acknowledging the interrupt**. The hardware may continue sending the interrupt until we explicitly acknowledge it.

Acknowledging Interrupts



- The GPIO will keep its interrupt output raised until we acknowledge it
- The Intc will keep its interrupt output raised until we acknowledge it.

- If the FIT generates an interrupt, how many places do we need to acknowledge?

- If the user presses a button, how many places do we need to acknowledge?
- Which should we acknowledge first?

- Let's look at how do we acknowledge the intc…

intc.h

- We have discussed several of the functions in intc.h, let's discuss the rest

https://github.com/byu-cpe/ecen427_student/blob/main/userspace/drivers/intc/intc.h