Introduction to Microprocessor Systems
What Is A Microprocessor System?
Well……

The Simple Answer:

It's an electronic system in which a microprocessor plays an important role. Duh!

Why Do We Care?

Because microprocessors are absolutely everywhere! You simply cannot exist anymore without interacting with microprocessors. You can't start a car, microwave a cup of coffee, turn on a television or CD player, or look at a digital watch without interacting with a microprocessor.

If you don't understand them, your capabilities as an engineer will be severely limited!

But What Is A Microprocessor System?

Actually, you already know. You just don't know you know!

It's simply a fairly complicated state machine!

If you passed EE3801, you already know just about everything there is to know about microprocessor systems. All we do here is add significantly more wires and show you how to manage the complexity to solve some really cool problems.
Let's Start From The Top

Any useful microprocessor-based computer system must have:

**Memory -**
- To store programs.
- To store data.

**I/O Devices -**
- LEDs
- 7-Segment Displays
- Video Monitors
- Relays
- Motors
- Keyboards

To allow information to be input and output.

**A Processing Unit -**
Which is a complex state machine that manipulates data and controls I/O devices according to the program stored in memory.
I Still Say You Know (Almost) Everything!

Where's The Memory?

Where's The IO?

Where's The Processor?

Is This A Microprocessor System?
It is a basic form of a computer.

A microprocessor system is usually far more flexible than a simple state machine. One part of this complexity is that the function of a microprocessor based system is determined by a program that controls the system operation. Change the program, and you change the function the system performs!

```c
main()
{
    printf("Hello World\n");
}
```

"Fantasy"

"Reality"
Here's Where Things Get Complicated

A microprocessor does not know anything about "C" or Assembly Language. It's just a big, ugly, state machine. It only looks at a pattern of 1s and 0s and reacts by spitting back 1's and 0's. The problem is that it spits out a lot of them -- fast!