Microprocessors and Microcontrollers

Basic organization of a simple microprocessor

This is typical of an early or small processor such as the 80x86 family, still used for applications like automotive processors.

The CPU has the ability to flexibly change state based on input data (instructions). It has an instruction pointer (address) register and many data registers - may be 8, 16, 32, or 64 bits wide.

Interrupts are also supported; IP and other registers immediately saved in “stack”, execution transfers to high-priority code.

Typically, a microprocessor is faster but has no built-in peripheral devices or I/O. By contrast, a microcontroller is a self-contained single-chip computer system, in most cases.

On the following pages we take a quick look at the Microchip PIC32MX250 microcontroller (a $4).
Overall organization of Microchip PIC32MX250 32-bit microcontrollers

CTMU: charge/timing measurement unit (for touchpad detection, etc.)
PWM: Pulse-width modulator
SPI, I2C: Clocked serial interfaces
PMP: Parallel master port (a versatile parallel digital data interface)
UART: RS232-type serial interface
RTCC: Real-time clock/calendar
Comparator: Analog comparator
DMAC: Direct memory access controller
ICD, JTAG: Programming and debugging
Pins can be used as digital input, digital output, or analog input; when used as outputs they can be configured normally or in open-drain configuration. As inputs, they include optional pullup resistors. Finally, peripheral devices like UARTs and counters can be flexibly assigned to varying pins.
Example of an internal peripheral controller: USB interface unit
Programming

The program memory is written with a special high-voltage programming pulse, generated by a PICkit3 programmer or similar device. The processors have a register and stack structure optimized for programming in C or C++. The actual hardware instruction set consists of primitive operations like the following:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Integer Add</td>
<td>$R_d = R_s + R_t$</td>
</tr>
<tr>
<td>ADDI</td>
<td>Integer Add Immediate</td>
<td>$R_t = R_s + \text{Immed}$</td>
</tr>
<tr>
<td>ADDIU</td>
<td>Unsigned Integer Add Immediate</td>
<td>$R_t = R_s + \text{U Immed}$</td>
</tr>
<tr>
<td>ADDU</td>
<td>Unsigned Integer Add</td>
<td>$R_d = R_s + \text{U} R_t$</td>
</tr>
<tr>
<td>AND</td>
<td>Logical AND</td>
<td>$R_d = R_s &amp; R_t$</td>
</tr>
<tr>
<td>ANDI</td>
<td>Logical AND Immediate</td>
<td>$R_t = R_s &amp; (0_{16}</td>
</tr>
<tr>
<td>B</td>
<td>Unconditional Branch (Assembler idioms for: BEQ r0, r0, offset)</td>
<td>$PC += (\text{int})\text{offset}$</td>
</tr>
</tbody>
</table>
| BAL         | Branch and Link (Assembler idioms for: BGEZAL r0, offset) | GPR[31] = PC + 8  
PC += (\text{int})\text{offset}$ |
| BEQ         | Branch On Equal | if $R_s == R_t$  
PC += (\text{int})\text{offset}$ |
| BEQL        | Branch On Equal Likely | if $R_s == R_t$  
PC += (\text{int})\text{offset}$  
else  
Ignore Next Instruction$ |
| BGEZ        | Branch on Greater Than or Equal To Zero | if $R_s[31]$  
PC += (\text{int})\text{offset}$ |
| BGEZAL      | Branch on Greater Than or Equal To Zero And Link | GPR[31] = PC + 8  
if $R_s[31]$  
PC += (\text{int})\text{offset}$ |

Most instructions will operate on 32-bit registers in a single instruction cycle; i.e., at a rate of 40-50 MHz. This is equivalent to a microprocessor with a much higher clock speed, since a complete instruction executes on each cycle.
**Typical circuit**

The schematic diagram below shows a PIC32 microcontroller with a precise crystal clock, a 6-pin programming header, a serial display output, and an SPI interface to a dual 12-bit DAC. It is similar to the configuration used in Lab 10 for Physics 3150.