Agenda

- Course outline/ goals and approach
- Class roster
- Get and check out boards
- Quick Overview of Computer Architecture
- Definition of Microcontrollers
- Components
- Microcontroller Market
- Key Suppliers
Introduction

- Basics of computer architecture
- How does a computer work
- Has anyone done assembly language programming of computers?
- Microcontrollers, microprocessors and microcomputers
- Microcontrollers are general purpose digital logic replacements
Basic Computer Architecture

Control Logic

Instruction Register

Registers

Operand 1

ALU

Operand 2

Instruction Memory

Results/data bus

Accumulator

I/O
MicroChip PIC Block diagram From: data sheet

Note 1: Higher order bits are from the STATUS register.
State Machine representation of CPU

- Instruction Fetch
- Load Instruction Register
- Instruction Decode
- Fetch Operands
- Execute
- Memory Access
- Save Result
RISC Vs. CISC

- RISC = Reduced Instruction Set Computer
- CISC = Complex Instruction Set Computer
- Microcontrollers = mostly RISC
- Microcomputer for PC = x86 is CISC
- Pipelining = easier in RISC
- PIC = 35 instructions
### Examples of Instructions: 1

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD LW</strong></td>
<td>Add Literal and W</td>
</tr>
<tr>
<td>Syntax:</td>
<td>([label] \ ADD LW \ k)</td>
</tr>
<tr>
<td>Operands:</td>
<td>(0 \leq k \leq 255)</td>
</tr>
<tr>
<td>Operation:</td>
<td>((W) + k \rightarrow (W))</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, DC, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.</td>
</tr>
</tbody>
</table>

| **ANDWF**   | AND W with \(f\) |
| Syntax:     | \([label] \ ANDWF \ f,d\) |
| Operands:   | \(0 \leq f \leq 127\)  
|             | \(d \in [0,1]\) |
| Operation:  | \((W) \ .AND. (f) \rightarrow (destination)\) |
| Status Affected: | Z |
| Description: | AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. |

Immediate: Add literal and w  
Register: AND W with F  

From: PIC data sheet
Assembly Language

- Human readable coding of machine language
- Assembler translates to binary
- Each line -> one instruction
Example of assembly Language program

```assembly
...... P1 PROGRAM : .......................................................

P1874

Toggle the top three LEDs every half second in sequence: bottom, middle, top.
Use 4 MHz crystal for 1 microsecond internal clock period.
Use Timer2 to obtain ten millisecond loop time.
Toggle bit 5 of PORTA each time around the mainline loop (to test loop time).
Echo RPG "encoder emulator" outputs to the bottom two LEDs in response to
presses of the "INC" and "DEC" pushbuttons.

...... Program hierarchy : .................................................

Mainline
  Initial
  Blink
  BlinkTable
  LoopTime
  
  IntService
  
  RPG

..................................................

list P=PIC16F874, F=INHX8M, C=160, N=77, ST=OFF, MM=OFF, R=DEC, X=OFF

#include P16F874.inc

__config(_CP_OFF & _PWRT_ON & _XT_OSC & _WDT_OFF & _BODEN_OFF)

...... Equates : .........................................................

BankORAM equ H'20' ;Start of Bank 0 RAM area
MaxCount equ 50 ;Number of loops in half a second
...... Variables : ..........................................................

cblock BankORAM
  W_TEMP ;Temporary storage for W during interrupts
  STATUS_TEMP ;Temporary storage for STATUS during interrupts
  BLNKCNT ;Loop counter for blinking LEDs every half sec.
  TEMP
  endc

...... Macro definitions : .............................................

MOVLF     macro   [literal,dest
  movlw    [literal
  movwf    dest
  endm

MOVFF     macro   [source,dest
  movf     [source,W
  movwf    dest
  endm

...... Vectors : .......................................................

org H'000' ;Reset vector
nop        ;'no operation'; needed for in circuit debuggгер
goto Mainline ;Branch past tables
org H'004' ;Interrupt vector
goto IntService ;Branch to interrupt service routine

...... BlinkTable subroutine : ......................................

This subroutine reads PORTD and retains only the upper three LED bits. It
```
BlinkTable

; Copy present state of LEDs to TEMP
MOVF PORTD, TEMP
; Move bits 7,6,5 to bits 3,2,1
SWAPF TEMP,W
; and on to bits 2,1,0 of W
RRF TEMP,W
; Keep only bits to be shifted
ANDLW B'000000111'
ADDWF PCL,F
; Change PC with PCLATH and offset in W
RETlw B'001000000'
; (000 -- 001) reinitialize to bottom
RETlw B'011000000'
; 001 -- 010 bottom to middle
RETlw B'110000000'
; 010 -- 100 middle to top
RETlw B'010000000'
; (000 -- 001) reinitialize to bottom
RETlw B'110000000'
; 010 -- 011 top to bottom
RETlw B'110000000'
; (101 -- 001) reinitialize to bottom
RETlw B'111000000'
; (110 -- 001) reinitialize to bottom
RETlw B'111000000'
; (111 -- 001) reinitialize to bottom

; End of Tables

; Mainline program

Mainline

call Initial ; Initialize everything
MainLoop

call Blink ; Blink upper three LEDs
call LoopTime ; Force loop time to be ten milliseconds
goto MainLoop

; Initial subroutine

; This subroutine performs all initializations of variables and registers.

Initial

BF SF STATUS,RP0 ; Set register access to bank 1
MOVLW B'000000100',ADCON1 ; Select PORTA pins for ADC or digital I/O
MOVLW B'000000101',TRISA ; Set I/O for PORTA
MOVLW B'000000111',TRISB ; Set I/O for PORTB
MOVLW B'110101111',TRISC ; Set I/O for PORTC
CLR TRISD ; Set I/O for PORTD
MOVLW B'000000100',TRISE ; Set I/O for PORTE
MOVLW 10,PR2 ; Set up Timer2 for a looptime of 10 ms
BCF STATUS,RP0 ; Set register access back to bank 0
MOVLW B'010011011',T2CON ; Finish set up of Timer2 (see page 62)
CLR PORTD ; Turn off LEDs
MOVLW B'111010000',INTCON ; Enable RB0/INT interrupts (see page 98)
return

; Blink subroutine

; This subroutine blinks a new LED every 0.5 second.

Blink

DECFSZ BLNKCNT,F ; Decrement loop counter and return if not zero
goto BlinkEnd
MOVLW MaxCount ; Reinitialize BLNKCNT
MOVF BLNKCNT
CALL BlinkTable ; Get bits to change into W
XORWF PORTD,F ; and toggle them into PORTD
BlinkEnd
return

; LoopTime subroutine


Higher Level Languages

• Compiler translates from code to computer instruction set.

• Examples: Basic, Fortran, C, C++, Java

• Application packages: Mathematica, Mathcad, spreadsheets \(\rightarrow\) even higher level languages
Operating Systems

• Perform I/O i.e. printing, storage to disk
• Graphical user interface [GUI], i.e. ‘Windows’
• Virtual memory
• Resource sharing [allows multiple programs to be running at the same time]
• Network interface
• Security
Embedded Systems

- Dedicated computers for ‘SMART’ applications
- Automotive: engine control, transmission control
- Microwaves, CDs, cellular telephones, remotes
- DSP: embedded controller optimized for math operations [multiplies, etc.] often used for image and sound processing
Summary

• We have given you a brief survey of computer essentials
• Computer engineers use computers as part of larger devices, i.e. electronic and control systems [embedded computers]
• Such computers can act as logic replacement devices
• Computer based systems are easier to document and upgrade because the design is in the [changeable] coding
Microcontrollers Overview

- Microcontrollers - a key impact technology for the 21st century

- Microcontrollers.com: “In the aggregate, PC microprocessors are responsible for less than 1% of all processors sold. Embedded processors outsell PC processors by more than 99%.”

- This course will provide enough information AND practical experience to get you started on the road to developing your own designs
Microcontrollers and Embedded Controllers

- Controls some process or aspect of the environment: [Microcontrollers Vs. DSPs](#)
- DSPs optimized for math [multiplies]
- Embedded controller may not be a microcontroller *per se* but is used for special purpose control application
- Typical applications: temperature control, smart instrument, GPS, digital lock, cell phone, etc.
Examples

• Personal information products: Cell phone, pager, watch, pocket recorder, calculator
• Laptop components: mouse, keyboard, modem, fax card, sound card, battery charger
• Home appliances: door lock, alarm clock, thermostat, air conditioner, tv remote, hair dryer, VCR, small refrigerator, exercise equipment, washer/dryer, microwave oven
• Toys; video games, cars, dolls, etc.
• Cars are about 20-30% silicon today, mostly microcontrollers ($4b/yr)
• Smart cards [credit cards plus]
• Usually anything with a keypad [simple calculators however have dedicated calculator chips]
Assignment 1: Due Next Monday

• One page [no more] written assignment *to be handed in.*

• Find a microcontroller based device or product [i.e. Logitech optical computer mouse].

• Determine what *specific* microcontroller is used in the device [i.e. PIC 16F874].

• Explain the function or functions of the microcontroller in the device [i.e. converts pulses from rotary encoder to serial RS 232 communication protocol for transmission to computer input port].

• Be prepared to present in class.
Microcontroller Families

• Most manufacturers offer a wide range of devices for low end to higher end applications

• Microchip shipped its 1 billionth microcontroller in the fall of 1999, the 2 billionth in spring of 2002, and its 3 billionth in winter of 2004.
Microcontroller Manufacturers

- Analog Devices
- Atmel
- Dallas Semiconductor
- Freescale Semiconductor
- Hitachi Semiconductor
- Intel
- Microchip
- National Semiconductor
- Renesas
- STMicro
- Texas Instruments
- Zilog
Microcontroller Market

- >40 suppliers, >50 architectures
- $26 Billion market
- Shipments- > 16 Billion in 2000, 8 bit > 1/2 market
- Major Players: Microchip 16Fxx, Intel 8051, Motorola MC68HCXX, National COP800, SGS/Thomson ST62, Zilog Z86Cxx
Microcontrollers

- Processing power: 4 bit, 8 bit, 16 bit, 32 bit
- 2003 market share

- Specific features: communications, keyboard handling, signal processing, video processing
Embedded Controller- components

- ALU (arithmetic logic unit)
- RAM (Random Access Memory)
- EEPROM (Electrically Erasable Programmable Read Only Memory)
- I/O (input/output) - serial and parallel
- Timers [typically three designated 0,1,2]
- A/D converter
- Clock
- USART [univ. synchronous/asynch receiver and xmtr]
- Interrupt controller [PIC16F874 diagram]
Speed comparison

From: Mark Palmer, AN520 A Comparison of 8 bit Microcontrollers, Microchip AN, 1995.
**Microcontroller Core Features:**
- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EPPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

**Peripheral Features:**
- Timer0: 8-bit timer/clock with 8-bit prescaler
- Timer1: 16-bit timer/clock with prescaler, can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/clock with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master Mode) and i²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)
Note 1: Higher order bits are from the STATUS register.
## 16F87x Instruction Set

**TABLE 13-2: PIC16CXXX INSTRUCTION SET**

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>14-Bit Opcode</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td><strong>BYTE-ORIENTED FILE REGISTER OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDWF f, d</td>
<td>Add W and f</td>
<td>1</td>
<td>00</td>
<td>0111</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>ANDWF f, d</td>
<td>AND W with f</td>
<td>1</td>
<td>00</td>
<td>0101</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>CLRF f</td>
<td>Clear f</td>
<td>1</td>
<td>00</td>
<td>0001</td>
<td>lfff ffff</td>
</tr>
<tr>
<td>CLRW -</td>
<td>Clear W</td>
<td>1</td>
<td>00</td>
<td>0001</td>
<td>0xxx xxxx</td>
</tr>
<tr>
<td>COMF f, d</td>
<td>Complement f</td>
<td>1</td>
<td>00</td>
<td>1001</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>DECF f, d</td>
<td>Decrement f</td>
<td>1</td>
<td>00</td>
<td>0011</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>DECFSZ f, d</td>
<td>Decrement f, Skip if 0</td>
<td>(2)</td>
<td>00</td>
<td>1011</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>INCF f, d</td>
<td>Increment f</td>
<td>1</td>
<td>00</td>
<td>1010</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>INCFSZ f, d</td>
<td>Increment f, Skip if 0</td>
<td>(2)</td>
<td>00</td>
<td>1111</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>IORWF f, d</td>
<td>Inclusive OR W with f</td>
<td>1</td>
<td>00</td>
<td>0100</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>MOVF f, d</td>
<td>Move f</td>
<td>1</td>
<td>00</td>
<td>1000</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>MOVWF f</td>
<td>Move W to f</td>
<td>1</td>
<td>00</td>
<td>0000</td>
<td>lfff ffff</td>
</tr>
<tr>
<td>NOP -</td>
<td>No Operation</td>
<td>1</td>
<td>00</td>
<td>0000</td>
<td>0xxx xxxx</td>
</tr>
<tr>
<td>RLF f, d</td>
<td>Rotate Left f through Carry</td>
<td>1</td>
<td>00</td>
<td>1101</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>RRF f, d</td>
<td>Rotate Right f through Carry</td>
<td>1</td>
<td>00</td>
<td>1110</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>SUBWF f, d</td>
<td>Subtract W from f</td>
<td>1</td>
<td>00</td>
<td>0010</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>SWAPF f, d</td>
<td>Swap nibbles in f</td>
<td>1</td>
<td>00</td>
<td>1110</td>
<td>dfff ffff</td>
</tr>
<tr>
<td>XORWF f, d</td>
<td>Exclusive OR W with f</td>
<td>1</td>
<td>00</td>
<td>0110</td>
<td>dfff ffff</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BIT-ORIENTED FILE REGISTER OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCF f, b</td>
<td>Bit Clear f</td>
<td>1</td>
<td>01</td>
<td>00bb</td>
<td>bfff ffff</td>
</tr>
<tr>
<td>BSF f, b</td>
<td>Bit Set f</td>
<td>1</td>
<td>01</td>
<td>01bb</td>
<td>bfff ffff</td>
</tr>
<tr>
<td>BTFSC f, b</td>
<td>Bit Test f, Skip if Clear</td>
<td>(2)</td>
<td>01</td>
<td>10bb</td>
<td>bfff ffff</td>
</tr>
<tr>
<td>BTFSS f, b</td>
<td>Bit Test f, Skip if Set</td>
<td>(2)</td>
<td>01</td>
<td>11bb</td>
<td>bfff ffff</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LITERAL AND CONTROL OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDLW k</td>
<td>Add literal and W</td>
<td>1</td>
<td>11</td>
<td>111x</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>ANDLW k</td>
<td>AND literal with W</td>
<td>1</td>
<td>11</td>
<td>1001</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>CALL k</td>
<td>Call subroutine</td>
<td>2</td>
<td>10</td>
<td>kkkk</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>CLRWD T</td>
<td>Clear Watchdog Timer</td>
<td>1</td>
<td>00</td>
<td>0000</td>
<td>0110 0100</td>
</tr>
<tr>
<td>GOTO k</td>
<td>Go to address</td>
<td>2</td>
<td>10</td>
<td>kkkk</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>IORLW k</td>
<td>Inclusive OR literal with W</td>
<td>1</td>
<td>11</td>
<td>1000</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>MOVLW k</td>
<td>Move literal to W</td>
<td>1</td>
<td>11</td>
<td>00xx</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>RETFIE -</td>
<td>Return from interrupt</td>
<td>2</td>
<td>00</td>
<td>0000</td>
<td>0000 1001</td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with literal in W</td>
<td>2</td>
<td>11</td>
<td>01xx</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>RETURN -</td>
<td>Return from Subroutine</td>
<td>2</td>
<td>00</td>
<td>0000</td>
<td>0000 1000</td>
</tr>
<tr>
<td>SLEEP -</td>
<td>Go into standby mode</td>
<td>1</td>
<td>00</td>
<td>0000</td>
<td>0110 0110</td>
</tr>
<tr>
<td>SUBLW k</td>
<td>Subtract W from literal</td>
<td>1</td>
<td>11</td>
<td>110x</td>
<td>kkkk kkkk</td>
</tr>
<tr>
<td>XORLW k</td>
<td>Exclusive OR literal with W</td>
<td>1</td>
<td>11</td>
<td>1010</td>
<td>kkkk kkkk</td>
</tr>
</tbody>
</table>
MPLAB IDE

- IDE = Integrated Development Environment
- MPLAB Editor
- MPLAB Assembler
- MPLAB ICD [in circuit debugger]
- MPLAB SIM [simulator]
- Programmer with ICD module
Blink.c

- Initial()
  - Initialize internal registers
- Blink()
  - Blink the LEDs
- LoopTime()
  - Wait for 10 milliseconds
- interrupt handler()
  - Check for interrupts from buttons
Summary

- Microprocessors and embedded controllers are a ubiquitous part of life today
- These devices come in a wide variety of configurations and designs
- Headhunters report that EEs familiar with $\mu$C, $\mu$P design are in the highest possible demand
- One pager due next Lecture!
More Information

• In the next session we will explore in more depth key features of μControllers
• Boards and ICDs in lab.
• Also 9-12 V \geq 300\ mA power supply.
• All major Manufacturers have web sites