ECE 354
Assembly Tutorial

02/04/03

Ramaswamy Ramaswamy

UMassAmherst

Outline

- Scope
- PIC 16F877 architecture overview
- PIC assembly
  - Instruction set examples
  - Assembler directives
  - Subroutines
  - Interrupts
- Common code examples

Reference Material

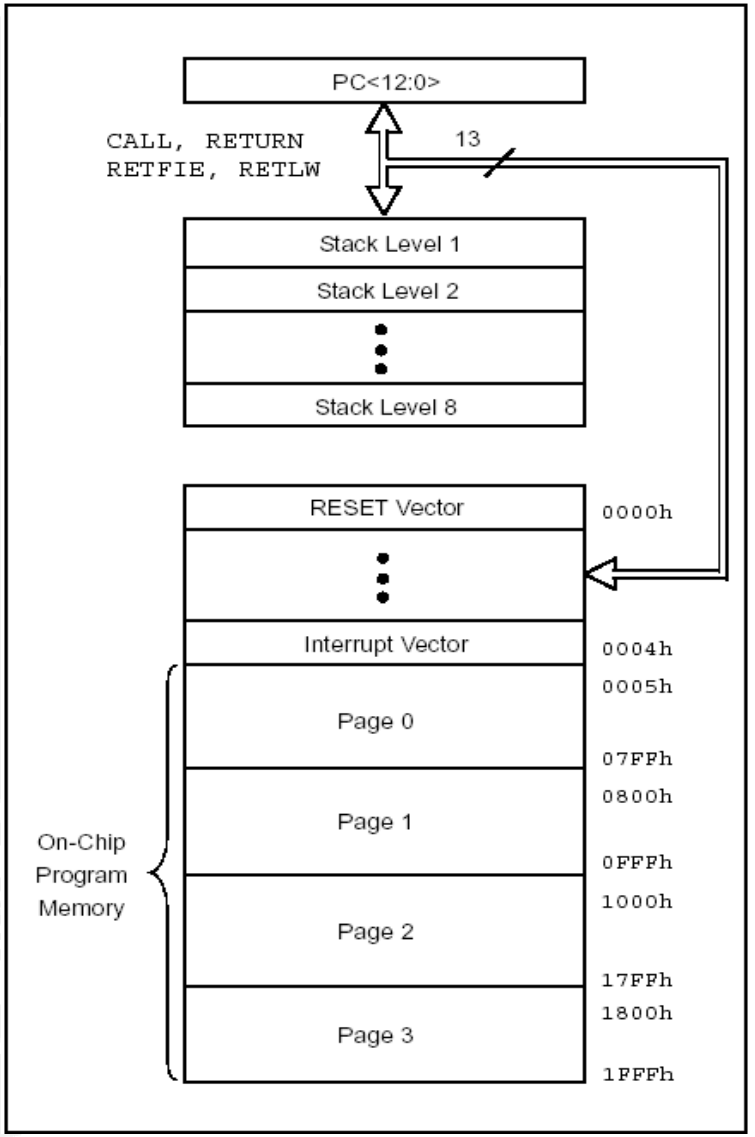
- PIC 16F87X data sheet
- PICMicro Mid-Range MCU Family Reference Manual
- Available on course web page

PIC16F877 Architecture Overview

- RISC CPU
- Harvard architecture
  - Program storage 8k x 14 bit
  - Data storage 256 x 8 bit, 384 x 8 bit
- 35 instructions
  - 14 bit instructions
  - 8 bit data values

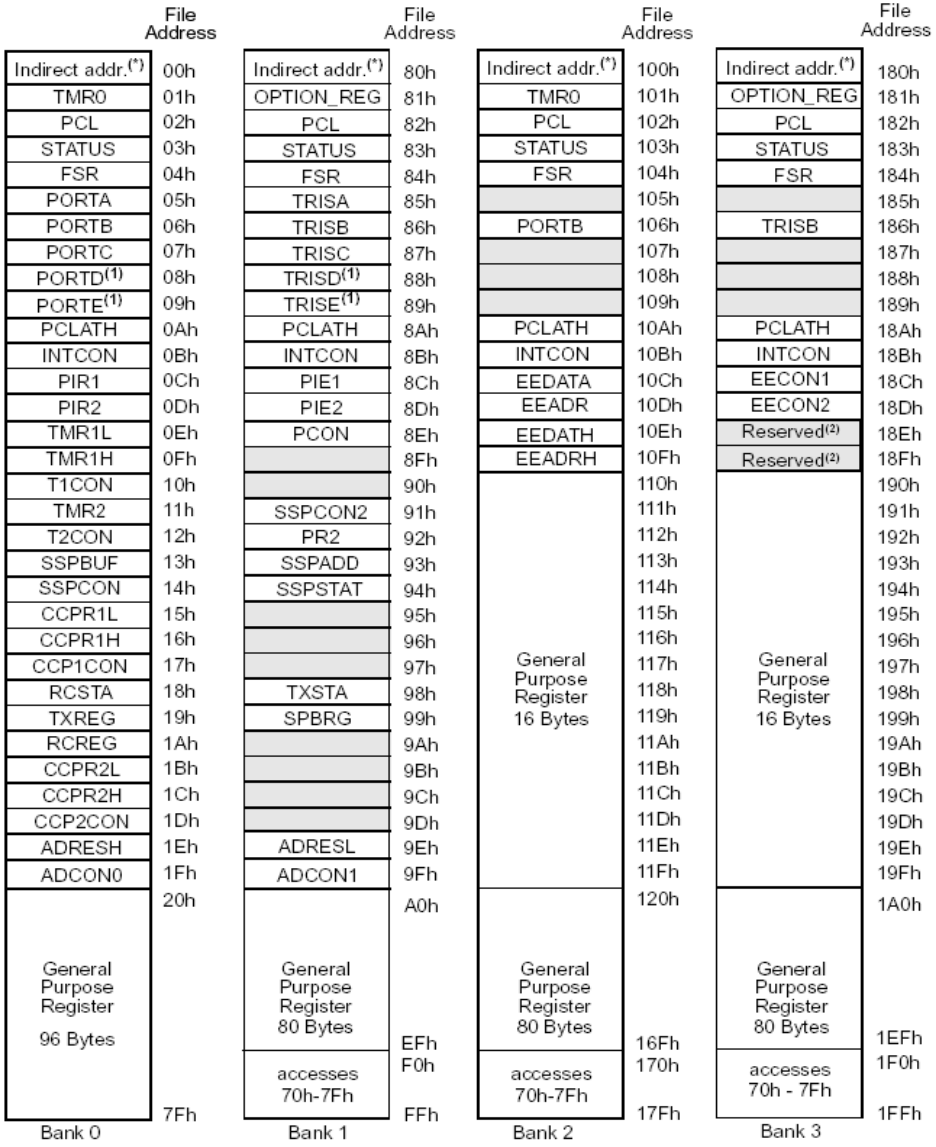
Program Memory

- 13 bit program counter
- Divided into 4 banks
- Important locations
  - 0x00 reset vector
  - 0x04 interrupt vector
- Programs start **AFTER** 0x04
- 8 level deep stack



Data Memory

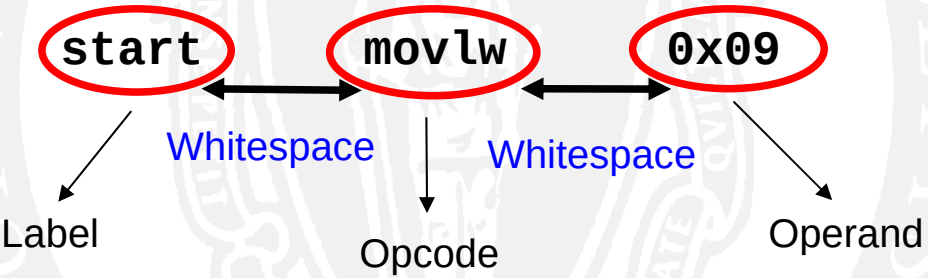
- 4 banks
- Important registers
  - STATUS
  - INDF
  - FSR
  - INTCON
- Use bit 5 and 6 of STATUS (RP0,RP1) to select bank #
- General purpose registers store user data
- Registers addressed directly, indirectly
- W register in accumulator



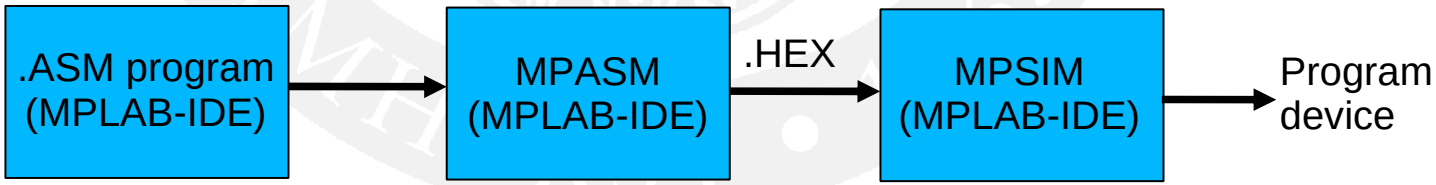
PIC Assembly Format

Column 1	Column 2	Column 3
Label,constants	Assembly opcode	Operands

Example :



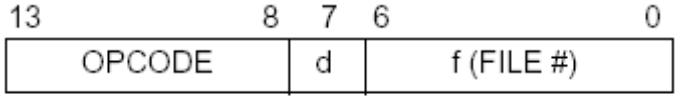
Assembly flow :



Instruction Format

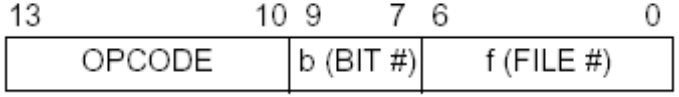
- 3 instruction types
  - Byte oriented
  - Bit oriented
  - Literal and control
- Full instruction listing in datasheet and Peatman
- All instructions take one cycle except conditional

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

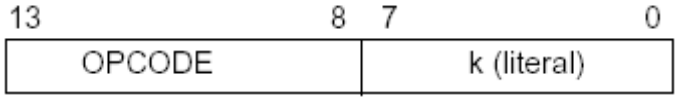
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

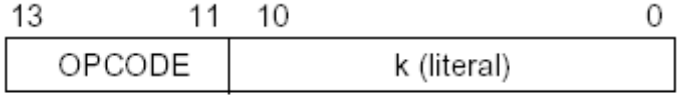
Literal and control operations

General



k = 8-bit literal (immediate) value

`CALL` and `GOTO` instructions only



k = 11-bit literal (immediate) value

Instruction Examples

- Byte oriented operations
  - **`ADDWF f,d`**
    - Add contents of W with register f
    - If d=0 store result in W else store in register f
    - Example : **`addwf 0x20,0`**
    - Can use constants to refer to file registers (recommended)
  - **`CLRF f`**
    - Contents of register f are cleared and Z bit (STATUS) is set
    - Example : **`clrf 0x30`**
  - **`MOVWF f`**
    - Move data from W register to register f
    - Example : **`movwf 0x04`**

Instruction Examples

- Byte oriented operations (contd.)
  - **`MOVF f,d`**
    - Move contents of register f to register W (d=0) or itself (d=1)
    - Example : **`movf 0x20,1`**
  - **`DECFSZ f,d`**
    - Decrement register f, place result depending on value of d
    - Skip the next instruction if result = zero
    - Example : **`decfsz 0x20,1`**
      - » instruction A
      - » instruction B
  - **`DECF f,d`**
    - Decrement f, place result depending on value of d
    - Example : **`decf 0x30,0`**

Instruction Examples

- Bit oriented operations
  - **`BSF f,b`**
    - Bit b in register f is set to 1
    - Example : **`bsf 0x03,5`**
    - **Manipulate bits of STATUS and INTCON register**
      - **enable/disable interrupts, select register banks**
  - **`BTFSC f,b`**
    - Test bit b of register f, skip next instruction if bit is 0
    - Skip the next instruction if result = zero
    - Example :

```
btfsc 0x03,2
instruction A
instruction B
```

Instruction Examples

- Literal and control operations
  - **`ADDLW k`**
    - Add literal k to register W
    - Example : **`addlw 0x05`**
  - **`MOVLW k`**
    - Move literal k into register W
    - Example : **`movlw 0x21`**
  - **`GOTO k`**
    - Unconditional branch. Literal k is loaded into PC
    - Example : **`GOTO THERE`**
    - Use of labels is recommended

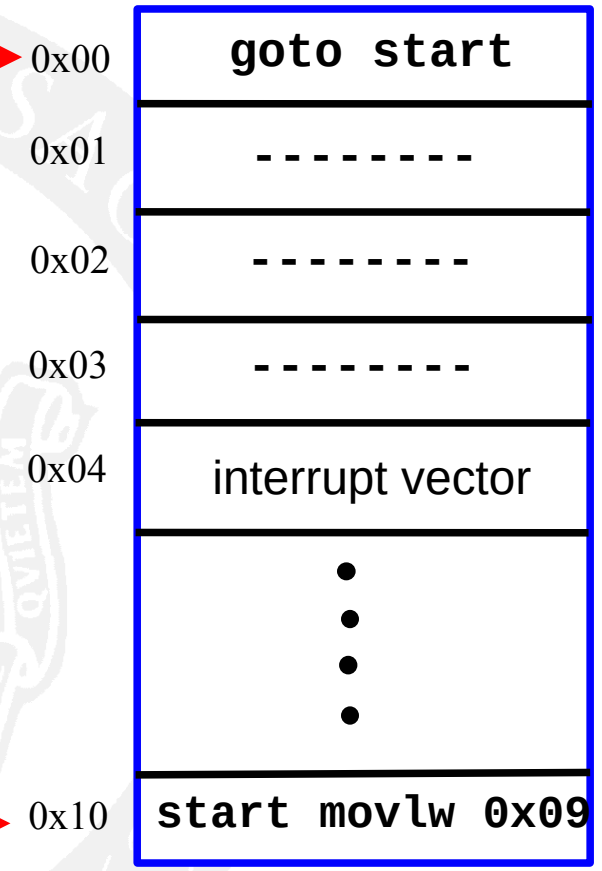
Assembler Directives

- Special instructions to assembler
- Important directives
  - **`ORG k`**
    - Place next instruction at location specified by k in program memory
    - Example :
```
ORG 0x10
movlw 0x09
```
  - **`constant EQU value`**
    - Value is assigned to constant
    - Example : **`COUNT EQU 0x20`**
    - Useful for linking register addresses to variable names
  - **`END`**
    - Indicates end of assembly program

Reset Sequence

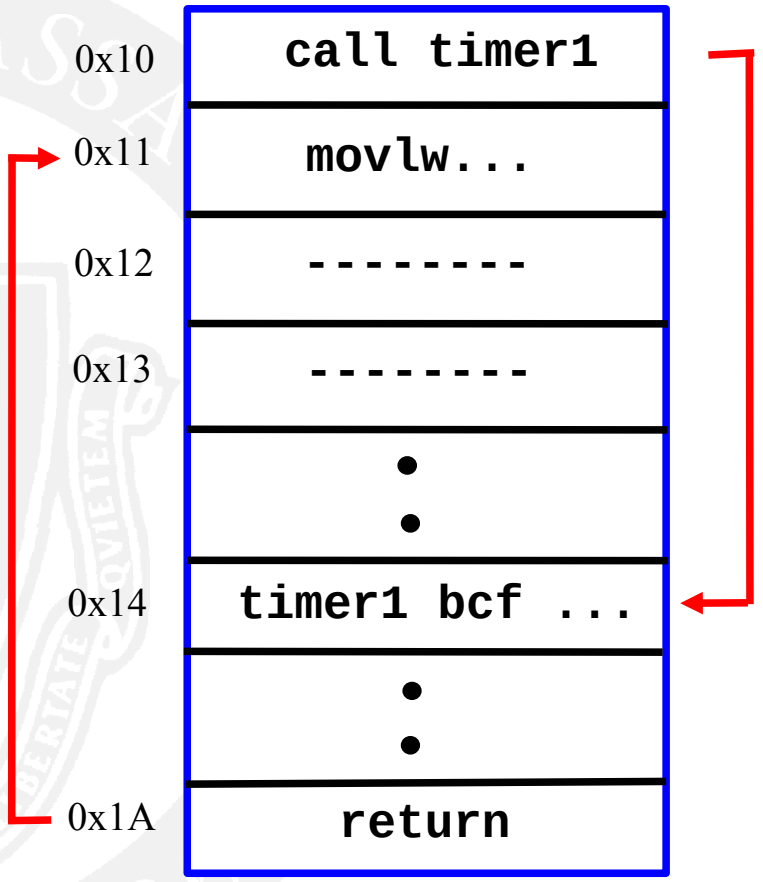
- Device reset causes PC to be reset to 0x00 (reset vector)
- Interrupt vector at 0x04
- All programs must start at 0x00
- Actual program starts after 0x04
  - Use ORG directive

```
org 0x00
goto start
org 0x10
start movlw 0x09
```



Subroutines

- 0x11 pushed onto stack
- PC loaded with 0x14
- Subroutine executes till 0x1A
- 0x11 popped off stack and stored in PC
- Upto 8 levels possible
  - What does this mean ?
- Stack wraps around



Interrupts

- Many sources of interrupts
- INTCON register
  - Bit 7 (GIE) enable/disable interrupts
  - GIE cleared when interrupt occurs
  - GIE set when “retfie” is executed
- Location 0x04 contains interrupt vector
  - Interrupt Service Routine (ISR) at 0x04
  - Return PC value saved onto stack
  - ISR must store/restore values of other registers (STATUS,W)
  - ISR must check type of interrupt

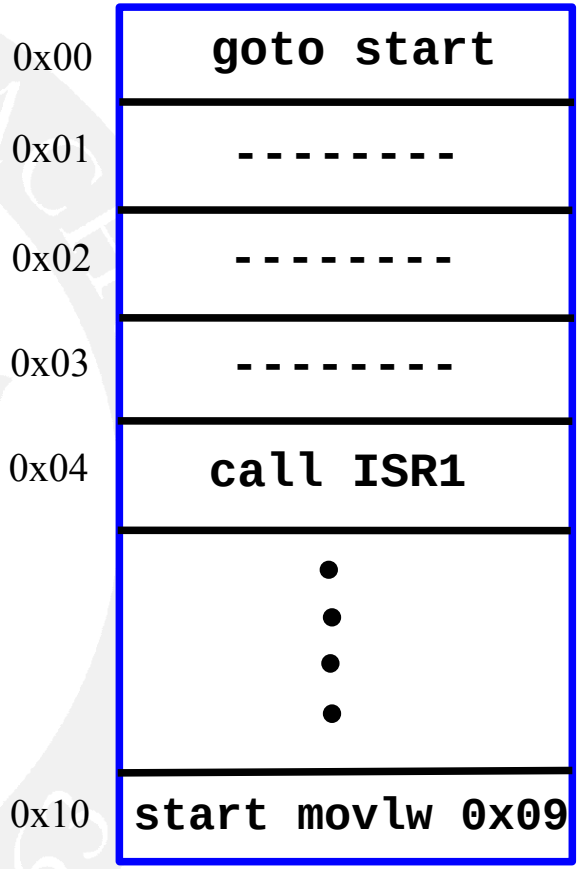
Interrupt Service Routine

```
ISR1   movwf TEMP_W
       swapf STATUS,W
       movwf STATUS_TEMP

       ;
       ; EXECUTE ISR CODE HERE
       ;

       swapf STATUS_TEMP,W
       movwf STATUS
       swapf W_TEMP,W
       retfie
```

store values of W,STATUS

restore values of W,STATUS

return from ISR, enable all interrupts



Indirect Addressing

- Data memory address is not fixed
- Special register (FSR - 0x04) used as pointer to memory location
- Opcode performed on INDF register (0x00)
- Useful for manipulating data tables in memory
- Example: clear a block of memory from 0x20 to 0x2F

```
movlw 0x20
movwf FSR
next clrf INDF
incf FSR,1
btfss FSR,4
goto next
```

initialize the FSR register

clear location pointed to by FSR using INDF

increment FSR, check if done

Common Code Sequences

- if-then-else condition checking

```
        btfsc STATUS,Z
        goto Zset
Zclear  bsf ....          → if (z==0)
        ; instructions to execute if Z=0
        goto Zdone
Zset
        ; instructions to execute if Z=1

Zdone   movlw....    ; Carry on with program
        end
```

- Use
  - btfsc,btfss to test with bits
  - decfsz,incfsz to test with bytes

A Simple Example

```
count equ 0x20           ; Equate register 0x20 to count
      org 0x00           ; Initialize reset vector
      goto start         ;
      org 0x10           ; Actual program starts here
start movlw 0x09         ; Move constant to W
      movwf count        ; Move W to count (0x20)
loop  decfsz count,1     ; Decrement count, skip if zero
      goto loop          ;
      movlw 0xFF         ; If count=0, move 0xFF to W
      movwf count        ; Move 0xFF to count
      end
```

Assembly Program Template

```
count equ 0x30
```

Assign symbolic names to registers

```
        org 0x00
        goto start
        org 0x04
        call IS_ROUTINE
        org 0x10
```

Initialize reset and interrupt vectors

```
    ; PROGRAM BODY
```

Main program body

```
IS_ROUTINE
        ; ISR BODY
        retfie
        end
```

Interrupt Service Routine body

Design Tips

- Use variable equates (equ) to assign registers symbolic names
- Comment your code
- Use MP-SIM to debug your code before you program device
- MPLAB-IDE
  - "RAM" - look at contents of register file
  - "ROM" - look at contents of program memory
  - "SFR" - look at contents of special function registers

Summary

- PIC 16F877
  - architecture
  - instruction set
  - addressing modes
  - special registers STATUS,FSR,INDF,INTCON
- Power on reset
- Subroutines, Interrupts, ISRs
- Use MPLAB-IDE to debug your work