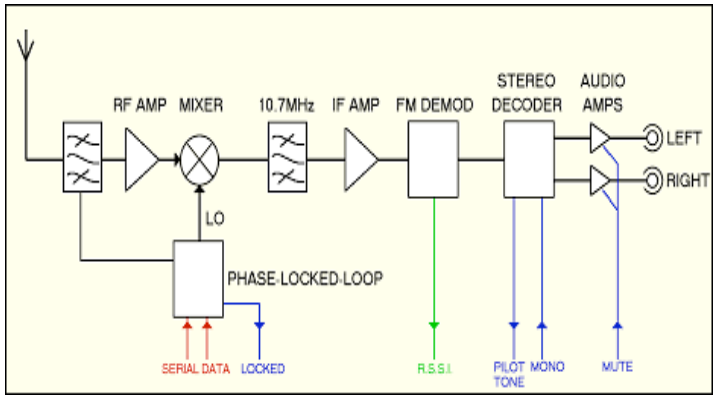
PIC Programming in C and Assembly

Krerk Piromsopa, Ph.D.
Department of Computer Engineering
Chulalongkorn University

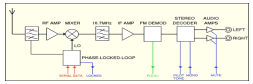
Outlines

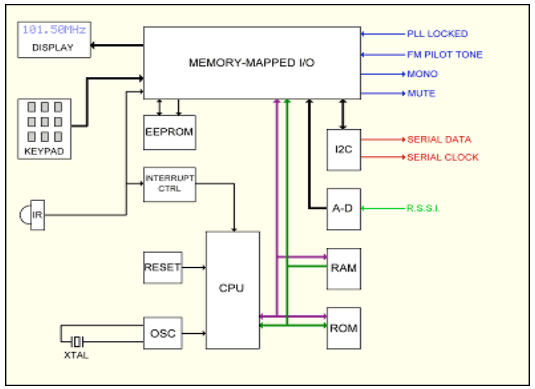
- Microprocessor vs. MicroController
- PIC in depth
- PIC Programming
  - Assembly Programming
  - Embedded C
  - In-line assembly
- PIC Interface

Microcontroller vs. Microprocessor

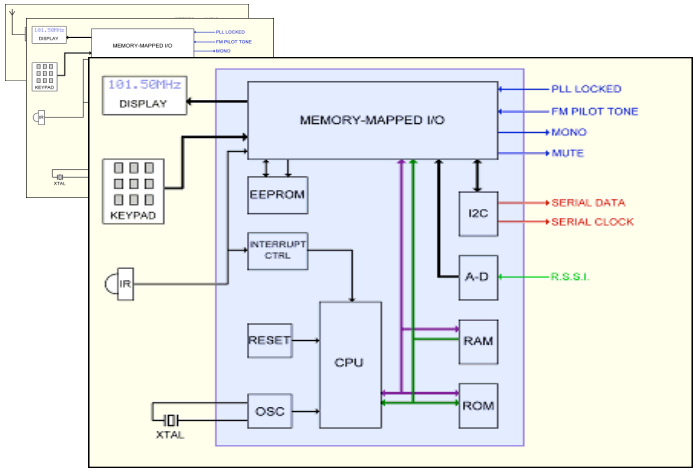


Microcontroller vs. Microprocessor

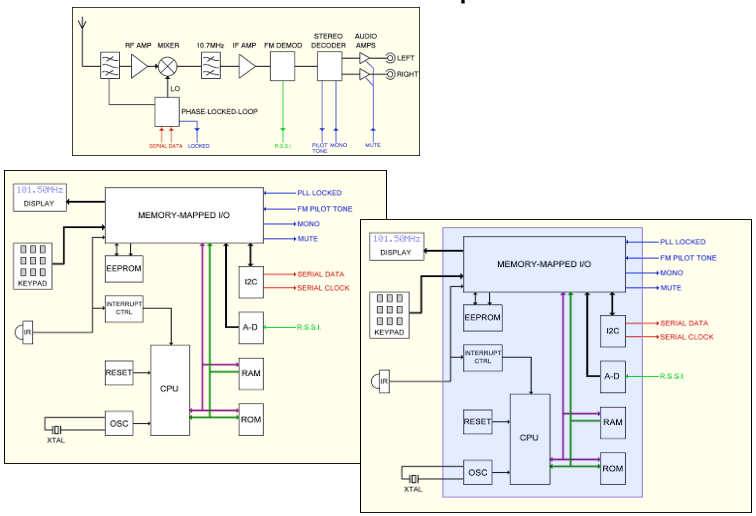




Microcontroller vs. Microprocessor



Microcontroller vs. Microprocessor

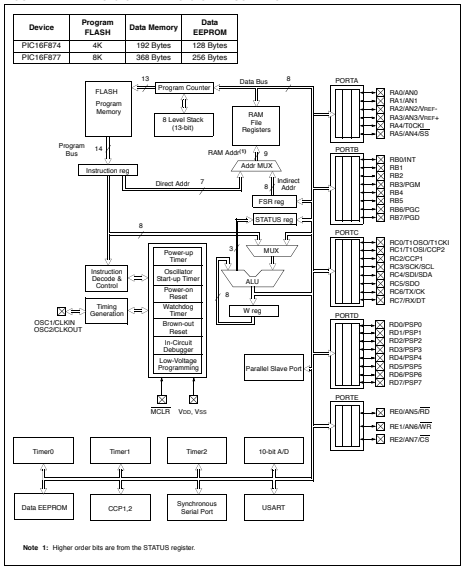


Figures from http://www.mhennessy.f9.co.uk/pic/introduction.htm

PIC in depth

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

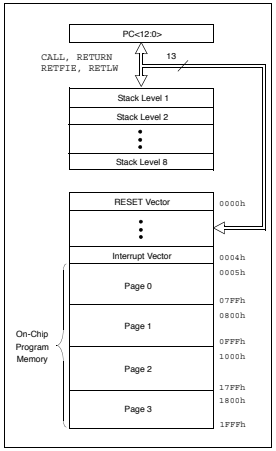
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

Memory Model

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK



Register File Map

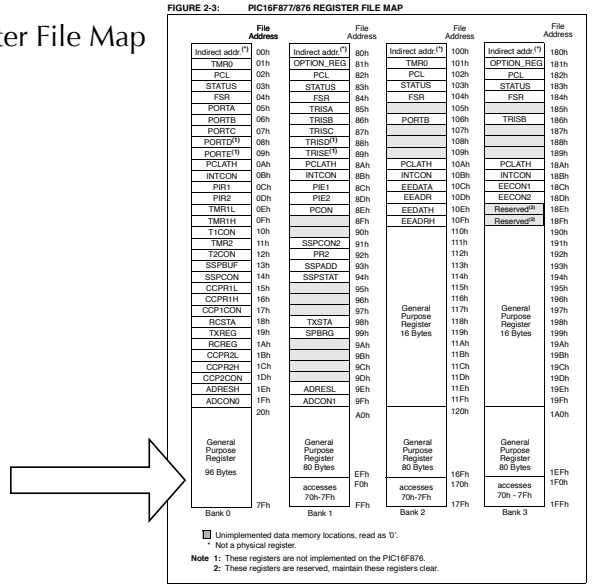


FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

File Address	Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3
00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h	Indirect addr.(*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h		185h	
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h		187h	
08h	PORTD(1)	88h	TRISD(1)	108h		188h	
09h	PORTE(1)	89h	TRISE(1)	109h		189h	
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	Reserved(2)
0Fh	TMR1H	8Fh		10Fh	EEADRH	18Fh	Reserved(2)
10h	T1CON	90h		110h		190h	
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h		115h		195h	
16h	CCPR1H	96h		116h		196h	
17h	CCP1CON	97h		117h	General Purpose Register 16 Bytes	197h	General Purpose Register 16 Bytes
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah		11Ah		19Ah	
1Bh	CCPR2L	9Bh		11Bh		19Bh	
1Ch	CCPR2H	9Ch		11Ch		19Ch	
1Dh	CCP2CON	9Dh		11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h	General Purpose Register 96 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	General Purpose Register 80 Bytes
7Fh		EFh	accesses 70h-7Fh	16Fh	accesses 70h-7Fh	1EFh	accesses 70h - 7Fh
		F0h		170h		1F0h	
		FFh		17Fh		1FFh	

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

Device?

- I/O Port
- EEPROM
- Timer & PWM
- Serial Port
- A2D
- Interrupt
- Watchdog Timer

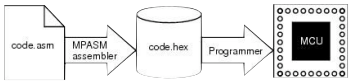
Can you use any of them?

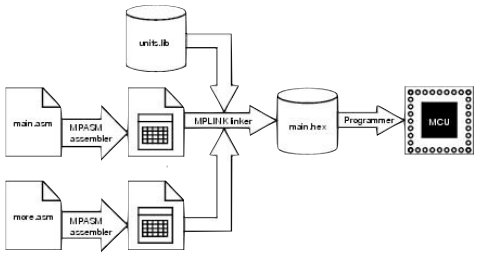
Assembly 101

- Assembly is the language closest to hardware level (machine language level).
- Why assembly?
  - Low level
  - speed
  - time critical
  - small program size

When to program in assembly?

Development Cycle





Component of Assembly

- Labels
  - start in column 1. may follow by a colon (:), space, tab or the end of line.
  - begin with an alpha character or an under bar (_) and may contain alphanumeric characters, the under bar and the question mark.
- Mnemonics, Directives and Macros
- Operands
- Comments

Example

Notice the directive

```
        list      p=18f452
        #include  p18f452.inc
Dest    equ       0x0B    ;Define constant
        org       0x0000  ;Reset vector
        goto      Start
        org       0x0020  ;Begin program
Start
        movlw     0x0A
        movwf     Dest
        bcf       Dest, 3 ;This line uses 2
operands
        goto      Start
        end
```

List File

Notice the directive

```
---  C:\PIC\SAMPLE.ASM  ----------------------------------------------

                            1:          list      p=16f877 , b=2, c=80
                            2:          #include  p16f877.inc
                            3:     Dest equ       0x0B    ;Define constant
                            4:          org       0x0000  ;Reset vector
                            5:          goto      Start
                            6:          org       0x0020  ;Begin program
                            7:     Start
0000    2820    GOTO 0x20   8:          movlw     0x0A
                            9:          movwf     Dest
0020    300A    MOVLW 0xa  10:          bcf       Dest, 3 ;This line uses 2 oper
0021    008B    MOVWF 0xb  11:          goto      Start
0022    118B    BCF 0xb, 0x3
0023    2820    GOTO 0x20
```

Quick tip

Use “banksel” directive to avoid bank switching issue.

```
        BSF     STATUS,RS0
        movlw   0x00
        movwf   TRISB
```

```
0020    1683     BSF 0x3, 0x5    8:     banksel TRISB
0022    3000     MOVLW 0         9:     movlw   0x00
0023    0086     MOVWF 0x6       10:    movwf   TRISB
```

Macro

```
OutPort macro myport, myvalue
    banksel myport
    movlw myvalue
    movwf myport
    endm
```

```
BCF 0x3, 0x5
MOVLW 0x55
MOVWF PORTB

BCF 0x3, 0x5
MOVLW 0xaa
MOVWF PORTA
```

```
OutPort PORTB, 0x55
OutPort PORTA, 0xAA
```

Advanced Macro

```
Add1_x  macro x
    local i=0
    clrw
    while i<x
      addlw   i
      i += 1
    endw

    endm
```

```
Add1_x  10
; This will give
CLRW
ADDLW 0
ADDLW 0x1
ADDLW 0x2
....
....
ADDLW 0xe
ADDLW 0xf
```

See MPASM HELP for more details.

Embedded C

Lab1 in C

```
#include <pic16f87.h>
void main (void) {
  TRISB=0x01;
  while ( 1 ) {
  if (RB0) {
    PORTB |=0x02;
    PORTB &=0xFB;
    //PORTB=0x02;
  } else {
    PORTB |=0x04;
    PORTB &=0xFD;
    //PORTB=0x04;
  }
  }
}
```

Mix C and Assembly

```
#include <pic16f87.h>
void main (void) {
  TRISB=0x01;
  while ( 1 ) {
  if (RB0) {
    #asm
    BSF _PORTB, 1
    BCF _PORTB, 2
    #endasm
  } else {
    asm("BCF  _PORTB, 1");
    asm("BSF  _PORTB, 2");
  }
  }
}
```

- Inline assembly
  - Use #asm and #endasm for multiple line.
  - Use asm(" ") for single line
- Add _ to variable in C

Good library can help inexperience programmer.

See 16 adders in C

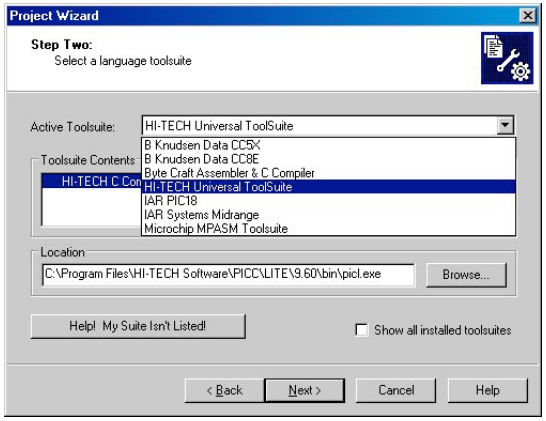
```
int sum(int i, int j) {
  int sum;

  sum=i+j;
  return sum;
}
```

How is your LAB 2?

```
CLRF 0x3
MOVF 0x22, W
ADDWF 0x20, W
MOVWF 0x20
MOVF 0x23, W
BTFSC 0x3, 0
INCF 0x23, W
ADDWF 0x21, W
MOVWF 0x21
MOVWF 0x7f
MOVF 0x20, W
MOVWF 0x7e
RETURN
```

What if there is no library to do your job?

How to use C in my LAB?



When to write in C?
When to write in Assembly?

POP Quiz:

Will recursive program work in PIC?