CENG 336

Architecture
Areas of use

• You are used to chips like the Pentium and the Athlon, but in terms of installed machines these are a small portion of total computer use.
• Think how many computers you have at home
• Digital cameras, video cameras, TVs, mobile phones, calculators, micro-wave ovens etc all contain processors.
Microcontrollers

- In order for a microprocessor to be used, other components such as memory, or components for receiving and sending data must be added to it.
- On the other hand, microcontroller is designed to be all of that in one. No other external components are needed for its application because all necessary peripherals are already built into it. Thus, we save the time and space needed to construct devices.
- The PICmicro was originally designed around 1980 by General Instrument as a small, fast, inexpensive embedded microcontroller with strong I/O capabilities. PIC stands for "Peripheral Interface Controller".
- General Instrument recognized the potential for the PIC and eventually spun off Microchip, headquartered in Chandler, AZ to fabricate and market the PICmicro.
Numbers of machines

• You might have 1 or 2 Pentium class chips at home. (Each PC will also contain several embedded processors)
• You will have perhaps 10 to 20 other embedded computers in other devices.
• If you think of childrens toys, the numbers grow even higher.
Characteristics

• Embedded computers have to be very low cost, simple and reliable.

• They cannot use any moving parts (disk drives) because:
  1. These are power hungry
  2. They are bulky
  3. They are expensive
Features

• Program in Read Only Memory – ROM
• Limited RAM storage – variables only not code
• Built in I/O devices
• Use very little power
Components of a Computer System

CPU

Memory System

Input/Output System
Microcontrollers

- Microcontrollers integrate all the components of a computer system onto a single chip
- All components are optimized to perform the functions necessary to control a larger system
- Size, capability, cost, and power consumption are more important considerations
- 8 bit microcontrollers have the majority of the market right now, but 16 and 32 bit microcontrollers are available and gaining market share
Common Microcontrollers

- Atmel
- ARM
- Intel
  - 8-bit
    - 8XC42
    - MCS48
    - MCS51
    - 8xC251
  - 16-bit
    - MCS96
    - MXS296
- National Semiconductor
  - COP8
- Microchip
  - 12-bit instruction PIC
  - 14-bit instruction PIC
    - PIC16F84
  - 16-bit instruction PIC
- NEC

- Motorola
  - 8-bit
    - 68HC05
    - 68HC08
    - 68HC11
  - 16-bit
    - 68HC12
    - 68HC16
  - 32-bit
    - 683xx
- Texas Instruments
  - TMS370
  - MSP430
- Zilog
  - Z8
  - Z86E02
Families

- Zilog Z8 series
- Intel 8051 series
- Arm 32 bit microcontrollers
- MicroChip – PIC microcontrollers
Zilog

• This company did more than any other to spread use of microprocessors in the 1970s and early 80s with their Z80 micro.
• This was the mainstay of early personal computers.
Current Z80e has

- eZ80® core operating at speeds up to 50MHz - Achieves high performance
- On-chip 10/100BaseT Ethernet MAC (eZ80F91 only) - Enables low-cost network applications
- Up to 256KB on-chip Flash memory
- Up to 16KB on-chip SRAM

This is a relatively high end micro controller
Intel 8051

- This was one of the first microcontrollers to integrate ROM, RAM and I/O on one chip.
- Made by many other companies still, Amtel, Texas instruments etc.
8051 Internal design
ARM series

- One of the first RISC processors
- 32 Bit architecture
- Low transistor count compared to conventional processors
- Very low power consumption
- Used in digital Phones and PDAs.
- At the top end of the micro-controller performance spectrum.
PIC

• Range of low end 8 bit microcontrollers.
• smallest have only 8 pins, largest 40 pins.
• Typical chip is an 18 pin one.
• Very cheap, you can pick them up at less than £1 each.
• Targeted at consumer products, burglar alarms etc.
Advantages of PIC

- It is a RISC (Reduced Instruction Set Computer) design
- Only thirty seven instructions to remember
- Its code is extremely efficient, allowing the PIC to run with typically less program memory than its larger competitors.
- It is low cost, high clock speed
Harvard architecture

• Like many micros the PIC is a Harvard not a von-Neumann machine
• This is simpler and faster

• Separate program bus and data bus: can be different widths!
• For example, PICs use:
  – Data memory (RAM): a small number of 8-bit registers
  – Program memory (ROM): 12-bit, 14-bit or 16-bit wide (in EPROM, FLASH, or ROM)
comparison

Von Neumann machine

Harvard Machine
• Harvard architecture is a newer concept than von-Neumann's. It rose out of the need to speed up the work of a microcontroller.
• In Harvard architecture, Data Access and Address Access are separate. Thus a greater flow of data is possible through the central processing unit.
• PIC16F877 uses 14 bits for instructions which allows for all instructions to be one word instructions.
Advantages of Harvard model

- An add operation of the form \( a := b + c \) must fetch 2 operands from memory and write 1 operand to memory. In addition it is likely to have to fetch 3 instructions from memory.
- With a single memory this will take 6 cycles. With 2 memories, we can fetch the instructions in parallel with the data and do it in 3 cycles.
- We have different word lengths for instructions and data – 8 bit data and perhaps 12 bit instructions.
Figure 4-1: Harvard vs. von Neumann Block Architectures
Von Neuman Architecture

- CPU
- MEMORY (program, data, stack)
- I/O Units

Data bus

Address and control busses
Von Neumann Architecture

- Used in: 80X86 (PCs), 8051, 68HC11, etc.
- Only one bus between CPU and memory
- RAM and program memory share the same bus and the same memory, and so must have the same bit width
- Bottleneck: Getting instructions interferes with accessing RAM
RISC Architecture

• Complex/Reduced Instruction Set Computers
• A minimal set of instructions, combined, can do every operation
• Usually execute in a single cycle
• CPU is smaller
• Other hardware can be added to the space: (overlapping register windows)
PIC Architecture: Background

Traditionally, CPUs are “CISC”
- Complex Instruction Set Computer (CISC)
- Used in: 80X86, 8051, 68HC11, etc.
- Many instructions (usually > 100)
- Many, many addressing modes
- Usually takes more than 1 internal clock cycle (Tcyc) to execute

Example:

MC68HC05: LDAA 0x55

1000 1100
01010101

2 bytes, 2 cycles
PIC Architecture: Background

PICs and most Harvard chips are “RISC”

- Reduced Instruction Set Computer (RISC)
- Used in: SPARC, ALPHA, Atmel AVR, etc.
- Few instructions (usually < 50)
- Only a few addressing modes
- Executes 1 instruction in 1 internal clock cycle (Tcyc)

Example:

```
PIC16CXXX: MOVLW 0x55
```

```
1100XX
01010101
```

1 word, 1 cycle
PIC Architecture: Convergence

Many Microcontrollers and DSP chips are “converging”

- Heading towards some mean between RISC and CISC
- Large CPUs (DSPs) are adding microcontroller like options
  (the 32bit, 100MHz StrongARM draws only 70mA)
- Small microcontrollers are getting more powerful, now able to do some DSP
- General trend: Smaller packages, less power consumption, faster
- Future possibility: “Sea of gates” reconfigurable processor
Example PIC: 12C508 Block Diagram
The PIC Family: Cores

PICs come with 1 of 4 CPU ‘cores’:

- **12bit** cores with 33 instructions: 12C50x, 16C5x
- **14bit** cores with 35 instructions: 12C67x, 16Cxxx
- **16bit** cores with 58 instructions: 17C4x, 17C7xx
- **‘Enhanced’ 16bit** cores with 77 instructions: 18Cxxx
The PIC Family: Packages

PICs come in a huge variety of packages:

8 pin DIPs, SOICs: 12C50x (12bit) and 12C67x (14bit)
18pin DIPs, SOICs: 16C5X (12bit), 16Cxxx (14bit)
28pin DIPs, SOICs: 16C5X (12bit), 16Cxxx (14bit)
40pin DIPs, SOICs: 16Cxxx (14bit), 17C4x (16bit)
44 - 68pin PLCCs*: 16Cxxx (14bit), 17C4x / 17Cxxx (16bit)

* also TQFPs, etc.
The PIC Family: Speed

PICs require a clock to work.

- Can use crystals, clock oscillators, or even an RC circuit.
- Some PICs have a built in 4MHz RC clock
  - Not very accurate, but requires no external components!
- Instruction speed = $1/4$ clock speed ($T_{cyc} = 4 \times T_{clk}$)
- All PICs can be run from DC to their maximum spec’d speed:

<table>
<thead>
<tr>
<th>Model</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>12C50x</td>
<td>4MHz</td>
</tr>
<tr>
<td>12C67x</td>
<td>10MHz</td>
</tr>
<tr>
<td>16Cxxx</td>
<td>20MHz</td>
</tr>
<tr>
<td>17C4x / 17C7xxx</td>
<td>33MHz</td>
</tr>
<tr>
<td>18Cxxx</td>
<td>40MHz</td>
</tr>
</tbody>
</table>
The PIC Family: Program Memory

PIC program space is different for each chip.

Some examples are:

- **12C508**  512  12bit instructions
- **16C71C**  1024 (1k) 14bit instructions
- **16F877**  8192 (8k) 14bit instructions
- **17C766**  16384 (16k) 16bit instructions
The PIC Family: Program Memory

PICs have two different types of program storage:

1. EPROM (Erasable Programmable Read Only Memory)
   - Needs high voltage from a programmer to program (~13V)
   - Needs windowed chips and UV light to erase
   - Note: One Time Programmable (OTP) chips are EPROM chips, but with no window!
   - PIC Examples: Any ‘C’ part: 12C50x, 17C7xx, etc.
The PIC Family: Program Memory

PICs have two different types of program storage:

2. FLASH
   ✗ Re-writable (even by chip itself)
   ✗ Much faster to develop on!
   ✗ Finite number of writes (~100k Writes)
   ✗ PIC Examples: Any ‘F’ part: 16F84, 16F87x, 18Fxxx (future)
The PIC Family: Data Memory

PICs use general purpose “file registers” for RAM (each register is 8bits for all PICs)

Some examples are:

- 12C508  25 Bytes RAM
- 16C71C  36 Bytes RAM
- 16F877  368 Bytes (plus 256 Bytes of nonvolatile EEPROM)
- 17C766  902 Bytes RAM

- Don’t forget, programs are stored in program space (not in data space), so low RAM values are OK.
The PIC Family: Control Registers

PICs use a series of “special function registers” for controlling peripherals and PIC behaviors.

Some examples are:

- **STATUS**: Bank select bits, ALU bits (zero, borrow, carry)
- **INTCON**: Interrupt control: interrupt enables, flags, etc.
- **TRIS**: Tristate control for digital I/O: which pins are ‘floating’
- **TXREG**: UART transmit register: the next byte to transmit
The PIC Family: Peripherals

Different PICs have different on-board peripherals

Some common peripherals are:

- Tri-state (“floatable”) digital I/O pins
- Analog to Digital Converters (ADC) (8, 10 and 12 bit, 50 ksp/s)
- Serial communications: UART (RS-232C), SPI, I²C, CAN
- Pulse Width Modulation (PWM) (10 bit)
- Timers and counters (8 and 16 bit)
- Watchdog timers, Brown out detect, LCD drivers
PIC Peripherals: Ports (Digital I/O)

- All PICs have digital I/O pins, called ‘Ports’
  - the 8pin 12C508 has 1 Port with 4 digital I/O pins
  - the 68pin 17C766 has 9 Ports with 66 digital I/O pins
- Ports have 2 control registers
  - TRISx sets whether each pin is an input or output
  - PORTx sets their output bit levels
- Most pins have 25mA source/sink (directly drives LEDs)
- WARNING: Other peripherals SHARE pins!
PIC Peripherals: ADCs

- Only available in 14bit and 16bit cores
- Fs (sample rate) < 54KHz
- Most 8bits, newer PICs have 10 or 12bits
- All are +/- 1LSB and are monotonic
- Theoretically higher accuracy when PIC is in sleep mode (less digital noise)
- Can generate an interrupt on ADC conversion done
- Multiplexed 3 (12C671) - 12 (17C7xxx) channel input
  - Must wait Tacq to charge up sampling capacitor (see datasheets)
PIC Peripherals: USART: UART

- Serial Communications Peripheral: Universal Synchronous/Asynchronous Receiver/Transmitter
- Only available in 14bit and 16bit cores
- Interrupt on TX buffer empty and RX buffer full

- Asynchronous communication: UART (RS-232C serial)
  - Can do 300bps - 115kbps
  - 8 or 9 bits, parity, start and stop bits, etc.
  - Outputs 5V so you need a RS232 level converter (e.g., MAX232)
PIC Peripherals: USART: USRT

- Synchronous communication: i.e., with clock signal

- **SPI** = Serial Peripheral Interface
  - 3 wire: Data in, Data out, Clock
  - Master/Slave (can have multiple masters)
  - Very high speed (1.6Mbps)
  - Full speed simultaneous send and receive (Full duplex)

- **I²C** = Inter IC
  - 2 wire: Data and Clock
  - Master/Slave (Single master only; multiple masters clumsy)
  - Lots of cheap I²C chips available; typically < 100kbps
    (For example, 8pin EEPROM chips, ADC, DACs, etc.)
PIC Peripherals: Timers

- Available in all PICs.
- 14+bit cores may generate interrupts on timer overflow.
- Some 8bits, some 16bits, some have prescalers
- Can use external pins as clock in/clock out (ie, for counting events or using a different \( F_{\text{osc}} \))
- Warning: some peripherals share Timer resources
PIC Peripherals: CCP Modules

- Capture/Compare/PWM (CCP)
- 10bit PWM width within 8bit PWM period (frequency)
  - Enhanced 16bit cores have better bit widths
- Frequency/Duty cycle resolution tradeoff
  - 19.5KHz has 10bit resolution
  - 40KHz has 8bit resolution
  - 1MHz has 1bit resolution (makes a 1MHz clock!)
- Can use PWM to do DAC - See AN655
- Capture counts external pin changes
- Compare will interrupt on when the timer equals the value in a compare register
PIC Peripherals: Misc.

- Sleep Mode: PIC shuts down until external interrupt (or internal timer) wakes it up.
- Interrupt on pin change: Generate an interrupt when a digital input pin changes state (for example, interrupt on keypress).
- Watchdog timer: Resets chip if not cleared before overflow
- Brown out detect: Resets chip at a known voltage level
- LCD drivers: Drives simple LCD displays
- Future: CAN bus, 12bit ADC, better analog functions

VIRTUAL PERIPHERALS:
- Peripherals programmed in software. UARTS, timers, and more can be done in software (but it takes most of the resources of the machine)
• For example: PIC16F877 is a RISC microcontroller,
  – it has a reduced set of instructions,
  – more precisely 35 instructions
  – (ex.Intel's and Motorola's microcontrollers have over hundred instructions).
  – All of these instructions are executed in one cycle except for jump and branch instructions.
Characteristics

- RISC CPUs
  - 8-bit
  - 16-bit
- Number of I/O pins: 4-70
- Memory types and sizes:
  - Flash; OTP; ROM
  - 0.5k – 256k
Speeds

• All PICs require oscillators to execute instructions:
  – Internal* (low speeds, up to 8 MHz)
  – External (high speeds, up to 40 MHz)

• Instructions are executed at least at $\frac{1}{4}$ oscillator speed (4 clocks/instruction)

(*Note: not all PICs have internal oscillators)
A/D converters and C/C modules

- All PICs have between 0 and 16 A/D converters with 8/10-bit resolution
- 8-16 bit Timers/Counters
- Comparator Modules (0-2)
Example: PIC16F877A

5/6 Programming pins
8 A/D channels
2 Oscillator Inputs
2 RS-232 inputs
33 I/O ports
Note 1: The high order bits of the Direct Address for the RAM are from the STATUS register.
2: Not all devices have this feature, please refer to device data sheet.
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.
Core PIC
PIC Ports
Included I/O Units

- Timer0
- Timer1
- Timer2
- 10-bit A/D
- Data EEPROM
- CCP1,2
- Synchronous Serial Port
- USART
- Comparator
- Voltage Reference
FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK

PC<12:0>

CALL, RETURN
RETIE, RETLW

Stack Level 1
Stack Level 2

Stack Level 8

RESET Vector

Interrupt Vector

On-Chip Program Memory

Page 0

Page 1

Page 2

Page 3

0000h

0004h

0005h

07FFh

0800h

0FFFFh

1000h

17FFh

1800h

1FFFFh
FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

<table>
<thead>
<tr>
<th>File Address</th>
<th>File Address</th>
<th>File Address</th>
<th>File Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indirect addr(1) 00h</td>
<td>Indirect addr(1) 80h</td>
<td>Indirect addr(1) 100h</td>
<td>Indirect addr(1) 180h</td>
</tr>
<tr>
<td>TMR0 01h</td>
<td>OPTION_REG 81h</td>
<td>TMR0 101h</td>
<td>OPTION_REG 181h</td>
</tr>
<tr>
<td>PCL 02h</td>
<td>PCL 82h</td>
<td>PCL 102h</td>
<td>PCL 182h</td>
</tr>
<tr>
<td>STATUS 03h</td>
<td>STATUS 83h</td>
<td>STATUS 103h</td>
<td>STATUS 183h</td>
</tr>
<tr>
<td>FSR 04h</td>
<td>FSR 84h</td>
<td>FSR 104h</td>
<td>FSR 184h</td>
</tr>
<tr>
<td>PORTA 05h</td>
<td>TRISA 85h</td>
<td>PORTB 105h</td>
<td>TRISB 185h</td>
</tr>
<tr>
<td>PORTB 06h</td>
<td>TRISB 86h</td>
<td>PORTC 106h</td>
<td>TRISC 186h</td>
</tr>
<tr>
<td>PORTC 07h</td>
<td>TRISC 87h</td>
<td>PORTD(1) 107h</td>
<td>TRISD(1) 187h</td>
</tr>
<tr>
<td>PORTD(1) 08h</td>
<td>TRISD(1) 88h</td>
<td>PORTE(1) 108h</td>
<td>TRISE(1) 188h</td>
</tr>
<tr>
<td>PORTE(1) 09h</td>
<td>TRISE(1) 89h</td>
<td>PCLATH 109h</td>
<td>PCLATH 189h</td>
</tr>
<tr>
<td>PCLATH 0Ah</td>
<td>PCLATH 8Ah</td>
<td>INTCON 10Ah</td>
<td>INTCON 18Ah</td>
</tr>
<tr>
<td>INTCON 06h</td>
<td>INTCON 86h</td>
<td>EECON1 10Dh</td>
<td>EECON1 18Dh</td>
</tr>
<tr>
<td>PIR1 0Ch</td>
<td>PIE1 8Ch</td>
<td>EEADR 10Dh</td>
<td>EEADR 18Dh</td>
</tr>
<tr>
<td>PIR2 0Dh</td>
<td>PIE2 8Dh</td>
<td>EEADRH 10Fh</td>
<td>EEADRH 18Fh</td>
</tr>
<tr>
<td>TMR1L 0Eh</td>
<td>PCON 8Eh</td>
<td>Reserved(2) 18eh</td>
<td>Reserved(2) 18eh</td>
</tr>
<tr>
<td>TMR1H 0Fh</td>
<td>PCON 8Fh</td>
<td>Reserved(2) 18Fh</td>
<td>Reserved(2) 18Fh</td>
</tr>
<tr>
<td>T1CON 10h</td>
<td>T1CON 90h</td>
<td>General Purpose Register 16 Bytes</td>
<td>General Purpose Register 16 Bytes</td>
</tr>
<tr>
<td>T1CON 11h</td>
<td>SSMCON2 91h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2CON 12h</td>
<td>PR2 92h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2CON 13h</td>
<td>SSPADD 93h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPBUF 14h</td>
<td>SSPADD 94h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPBUF 15h</td>
<td>SSPSTAT 95h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR1L 16h</td>
<td>SSPSTAT 96h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR1L 17h</td>
<td>TXSTA 97h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP1CON 18h</td>
<td>TXSTA 98h</td>
<td>General Purpose Register 80 Bytes</td>
<td>General Purpose Register 80 Bytes</td>
</tr>
<tr>
<td>RCSTA 19h</td>
<td>SPBRG 99h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXREG 1Ah</td>
<td>SPBRG 9Ah</td>
<td>16Fh</td>
<td>16Fh</td>
</tr>
<tr>
<td>RCREG 1Bh</td>
<td>9Ah</td>
<td>170h</td>
<td>170h</td>
</tr>
<tr>
<td>CCP2L 1Ch</td>
<td>9Bh</td>
<td>accesses 70h-7Fh</td>
<td>accesses 70h-7Fh</td>
</tr>
<tr>
<td>CCP2H 1Dh</td>
<td>9Ch</td>
<td>1FFh</td>
<td>1FFh</td>
</tr>
<tr>
<td>CCP2CON 1Eh</td>
<td>9Dh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADRESH 1Fh</td>
<td>ADRESL 9Eh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADCON0 20h</td>
<td>ADCON1 9Fh</td>
<td>1F0h</td>
<td>1F0h</td>
</tr>
<tr>
<td>General Purpose Register 96 Bytes</td>
<td>accesses 70h-7Fh</td>
<td>17Fh</td>
<td>17Fh</td>
</tr>
<tr>
<td>Bank 0</td>
<td>Bank 1</td>
<td>Bank 2</td>
<td>Bank 3</td>
</tr>
<tr>
<td>7Fh</td>
<td>F0h</td>
<td>170h</td>
<td>1FFh</td>
</tr>
</tbody>
</table>

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

Unimplemented data memory locations, read as 0.
* Not a physical register.
<table>
<thead>
<tr>
<th>Pin</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>IRP</td>
<td>RP1</td>
<td>RP0</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
</tr>
</tbody>
</table>

**STATUS (0x03, 0x83, 0x103, 0x183)**
Instruction Set

- 35 instructions of 6 types:
  - Set and clear bits
  - Move (memory / working register)
  - Arithmetic ops (add subtract)
  - Logical ops (and, or, exor)
  - Shift
  - Goto, call, return, skip next instr on condition
### Single Bit Operations

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bcf</strong></td>
<td>PORTB.0</td>
<td>; clear bit 0 of Port B (pin #21)</td>
</tr>
<tr>
<td><strong>bsf</strong></td>
<td>STATUS.C</td>
<td>; set the carry bit</td>
</tr>
</tbody>
</table>

### Clear & Move

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clrw</strong></td>
<td></td>
<td>; Clear the W register</td>
</tr>
<tr>
<td><strong>clrf</strong></td>
<td>TEMP1</td>
<td>; Clear the register with label TEMP1</td>
</tr>
<tr>
<td><strong>movlw</strong></td>
<td>5</td>
<td>; load #5 into W</td>
</tr>
<tr>
<td><strong>movwf</strong></td>
<td>TEMP1</td>
<td>; move W into TEMP1</td>
</tr>
<tr>
<td><strong>movf</strong></td>
<td>TEMP1,W</td>
<td>; move TEMP1 into W</td>
</tr>
<tr>
<td><strong>swapf</strong></td>
<td>TEMP1,F</td>
<td>; swap nibbles of TEMP1. Place the result in TEMP1</td>
</tr>
<tr>
<td><strong>swapf</strong></td>
<td>TEMP1,W</td>
<td>; swap nibbles of TEMP1. Place the result in W</td>
</tr>
</tbody>
</table>

### Increment & Decrement

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>incf</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 + 1</td>
</tr>
<tr>
<td><strong>incf</strong></td>
<td>TEMP1,W</td>
<td>; W = TEMP1 + 1</td>
</tr>
<tr>
<td><strong>decf</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 - 1</td>
</tr>
<tr>
<td><strong>decf</strong></td>
<td>TEMP1,W</td>
<td>; W = TEMP1 - 1</td>
</tr>
</tbody>
</table>

### Bit Operations

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>andlw</strong></td>
<td>b'000000111'</td>
<td>; clear bits 7-4 of W</td>
</tr>
<tr>
<td><strong>movlw</strong></td>
<td>b'000000111'</td>
<td>; clear bits 7-4 of TEMP1</td>
</tr>
<tr>
<td><strong>andwf</strong></td>
<td>TEMP1,F</td>
<td>; clear bits 7-4 of TEMP1</td>
</tr>
<tr>
<td><strong>movlw</strong></td>
<td>b'000000111'</td>
<td>; set bits 0-3 of TEMP1</td>
</tr>
<tr>
<td><strong>iorwf</strong></td>
<td>TEMP1,F</td>
<td>; (set bits 0-3 of TEMP1)</td>
</tr>
<tr>
<td><strong>movlw</strong></td>
<td>b'000000111'</td>
<td>; (toggle bits 0-3 of TEMP1)</td>
</tr>
</tbody>
</table>

### Addition / Subtraction

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>addlw</strong></td>
<td>5</td>
<td>; W = W + 5</td>
</tr>
<tr>
<td><strong>addwf</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 + W</td>
</tr>
<tr>
<td><strong>sublw</strong></td>
<td>5</td>
<td>; W = 5 - W</td>
</tr>
<tr>
<td><strong>subwf</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 - W</td>
</tr>
</tbody>
</table>

### Rotate

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rlf</strong></td>
<td>TEMP1,F</td>
<td>; nine-bit rotate left (carry = 9th bit)</td>
</tr>
<tr>
<td><strong>rrf</strong></td>
<td>TEMP1,W</td>
<td>; nine-bit rotate right (carry = 9th bit)</td>
</tr>
</tbody>
</table>

### Conditional Branch

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>btfsc</strong></td>
<td>TEMP1,S</td>
<td>; test bit #0 of TEMP1</td>
</tr>
<tr>
<td><strong>btfss</strong></td>
<td>STATUS.C</td>
<td>; test if a carry resulted</td>
</tr>
<tr>
<td><strong>incfsz</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 + 1. Skip next inst if zero.</td>
</tr>
<tr>
<td><strong>decfsz</strong></td>
<td>TEMP1,F</td>
<td>; TEMP1 = TEMP1 - 1. Skip next inst if zero</td>
</tr>
</tbody>
</table>

### Program Control

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>goto</strong></td>
<td>There</td>
<td>; jump to label There</td>
</tr>
<tr>
<td><strong>call</strong></td>
<td>There</td>
<td>; save the return address on the stack</td>
</tr>
<tr>
<td><strong>return</strong></td>
<td></td>
<td>; pull the return address off the stack</td>
</tr>
<tr>
<td><strong>nosp</strong></td>
<td></td>
<td>; do nothing</td>
</tr>
</tbody>
</table>