PART II

The PIC Microcontrollers

Programmable Interrupt Controller

Peripheral Interface Controller

EEC 343

Lecture_1

Microprocessors vs Microcontroller

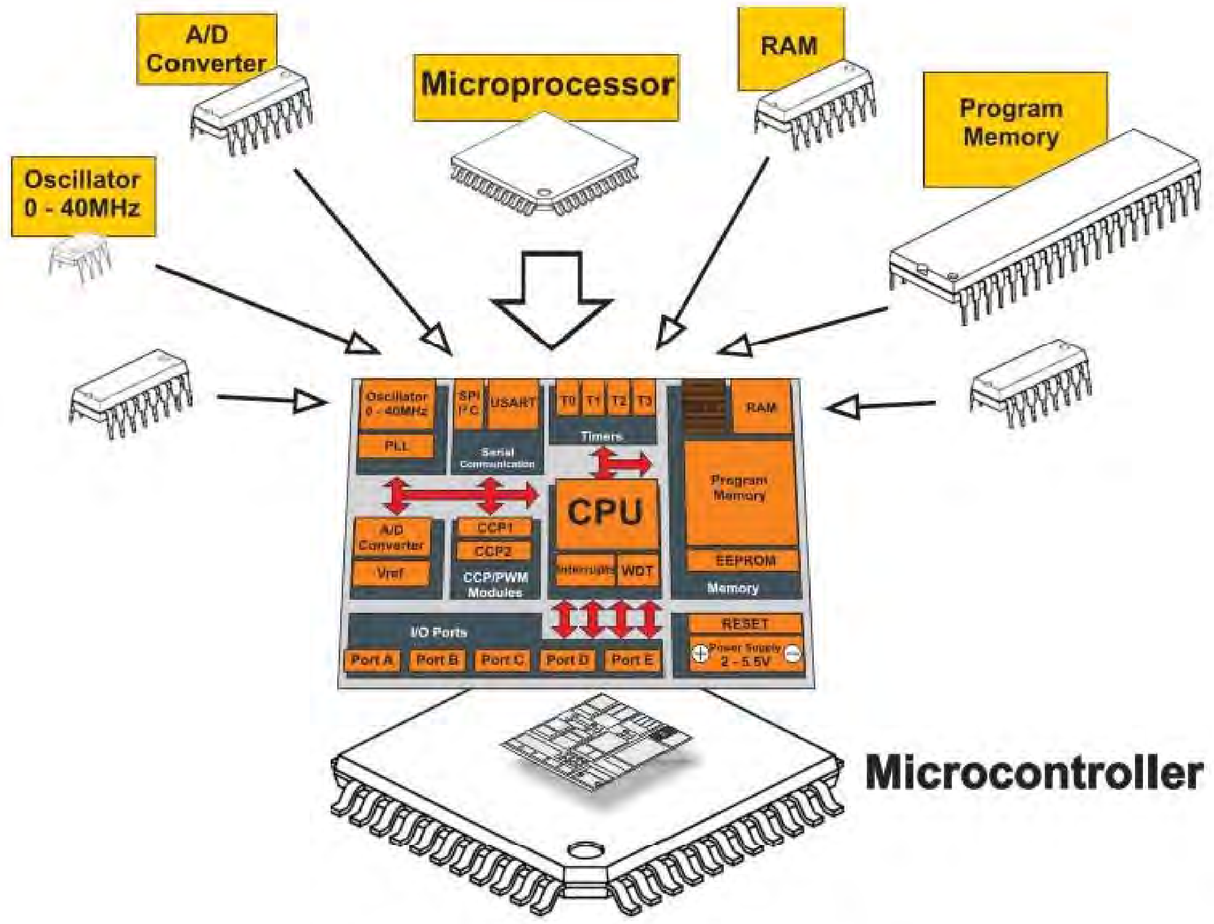


Fig. 0-1 Microcontroller versus Microprocessor

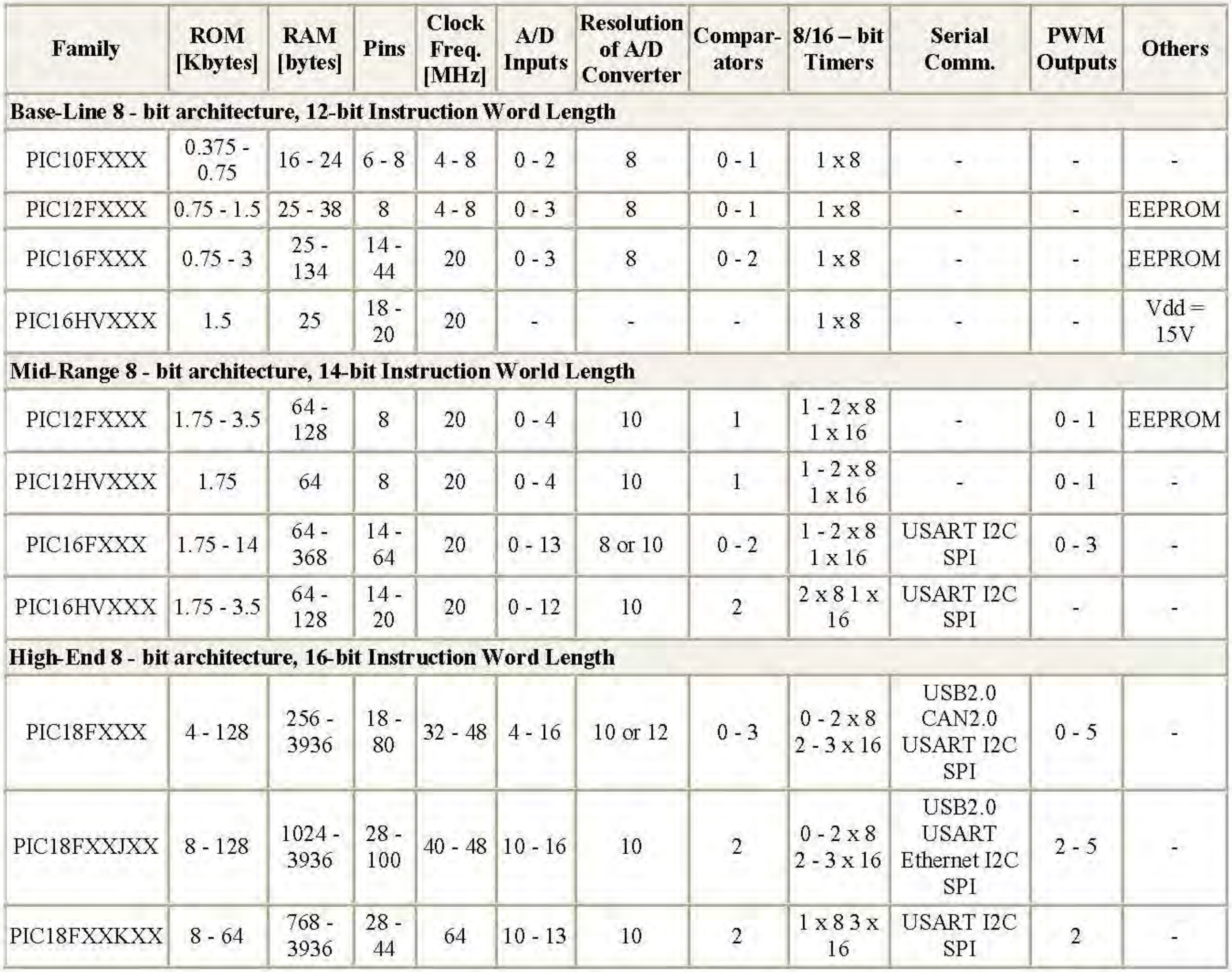
Microprocessors vs Microcontrollers

Feature	Microcontroller (µC)	Microprocessor (µP)
Purpose	Designed for specific embedded system applications	Designed for general-purpose computing applications
Architecture	Single-chip computer system with on-board memory, peripherals, and I/O interfaces	CPU with minimal on-board memory, peripherals and I/O interfaces
Integration level	Highly integrated	Less integrated
System architecture	Single-chip system	CPU + support chips
Processing power	Lower power	Higher power

Microprocessors vs Microcontroller

Feature	Microcontroller (μC)	Microprocessor (μP)
Instruction set	Fixed instruction set	More flexible
Clock speed	Lower clock speed, typically less than 100 MHz	Higher clock speed, typically greater than 1 GHz

Types of Microchip PIC µCs



Family	ROM [Kbytes]	RAM [bytes]	Pins	Clock Freq. [MHz]	A/D Inputs	Resolution of A/D Converter	Compar-ators	8/16 – bit Timers	Serial Comm.	PWM Outputs	Others
Base-Line 8 - bit architecture, 12-bit Instruction Word Length											
PIC10FXXX	0.375 - 0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	-	-	-
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	-	-	EEPROM
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	-	-	EEPROM
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	-	-	Vdd = 15V
Mid-Range 8 - bit architecture, 14-bit Instruction World Length											
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	EEPROM
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	-
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	1 - 2 x 8 1 x 16	USART I2C SPI	0 - 3	-
PIC16HVXXX	1.75 - 3.5	64 - 128	14 - 20	20	0 - 12	10	2	2 x 8 1 x 16	USART I2C SPI	-	-
High-End 8 - bit architecture, 16-bit Instruction Word Length											
PIC18FXXX	4 - 128	256 - 3936	18 - 80	32 - 48	4 - 16	10 or 12	0 - 3	0 - 2 x 8 2 - 3 x 16	USB2.0 CAN2.0 USART I2C SPI	0 - 5	-
PIC18FXXJXX	8 - 128	1024 - 3936	28 - 100	40 - 48	10 - 16	10	2	0 - 2 x 8 2 - 3 x 16	USB2.0 USART Ethernet I2C SPI	2 - 5	-
PIC18FXXKXX	8 - 64	768 - 3936	28 - 44	64	10 - 13	10	2	1 x 8 3 x 16	USART I2C SPI	2	-

Where we find PIC microcontrollers

- **Automotive systems**: Engine control units, dashboard displays, and body electronics.

- **Consumer electronics**: Remote controls, smart home devices, digital clocks, and appliances.

- **Industrial control**: Motor control, process automation, and programmable logic controller (PLC) systems.

- **Medical devices:** Patient monitoring equipment and portable diagnostic tools.

- **Power tools**: Used for motor control and battery management in cordless drills and other tools.

- **Vending machines**: Managing payment systems and product dispensing mechanisms.

PICmicro® MCU Microcontroller Overview

- Microchip uses a Harvard Architecture with separate address & data buses
  - Program bus: 12-, 14- & 16-bit wide instructions
  - Data bus: 8-bit wide data path
- Package sizes available from Microchip
    - 8 pins through 84 pins

Architecture (x14) 16F84A

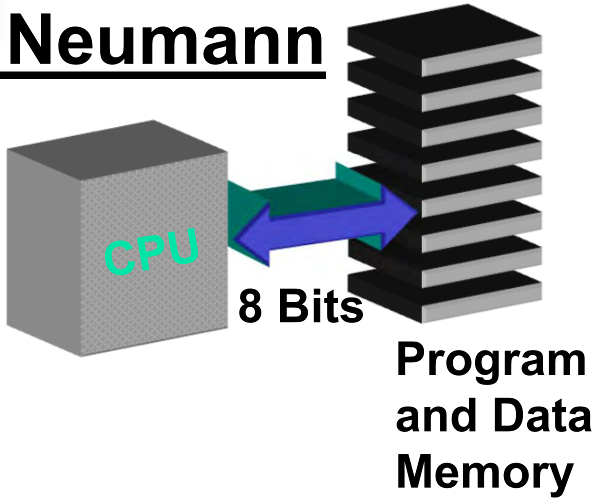
PICmicro® MCU Architecture

- RISC Microcontroller Features
- The high performance of the PICmicro® MCU can be attributed to the following:
  - Harvard Architecture
  - Instruction pipelining
  - Register file concept
  - Single cycle instructions
  - All single word instructions
  - Long word instruction
  - Reduced instruction set
  - Orthogonal instruction set

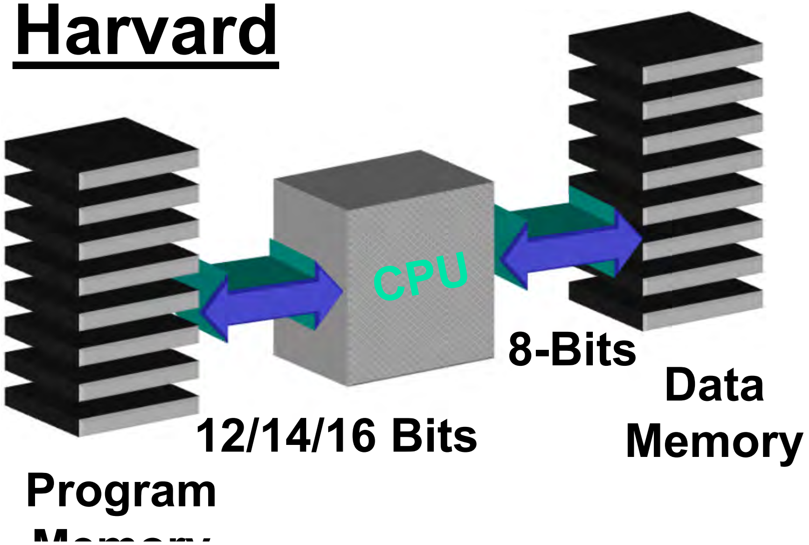
PICmicro® MCU Architecture

Harvard Architecture

Von Neumann



Harvard



- Fetches instructions and data from one memory
  - Limits operating bandwidth
- Two separate memory spaces for instructions and data
  - Increases throughput
  - Different program and data bus widths are possible

Pipelining

1. MOVLW 55h
2. MOVWF PORTB
3. CALL SUB_1
4. BSF PORTA, BIT3

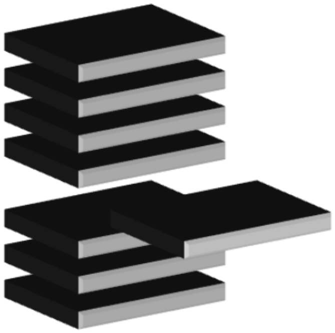
Pipeline

- In most microcontrollers, instructions are fetched and executed *sequentially*
- Allows overlap of fetch and execution
- Makes single cycle execution
- Program branches (e.g. GOTO, CALL or WRITE to PC) takes two cycles

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4
1. MOVLW 55h	**Fetch 1**	**Execute 1**			
2. MOVWF PORTB		**Fetch 2**	**Execute 2**		
3. CALL SUB_1			**Fetch 3**	**Execute 3**	
4. BSF PORTA, BIT3				**Fetch 4**	**Flush Fetch 4**
					Fetch SUB_1

PICmicro® MCU Architecture

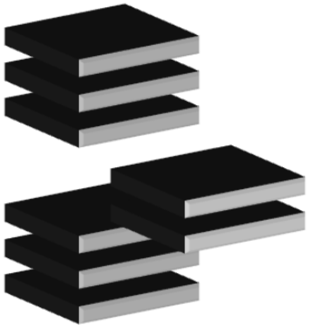
Long word Instructions



PICmicro® MCU

```
movlw     #imm<8>
```

1100XX k k k k k k k k



MC68HC05

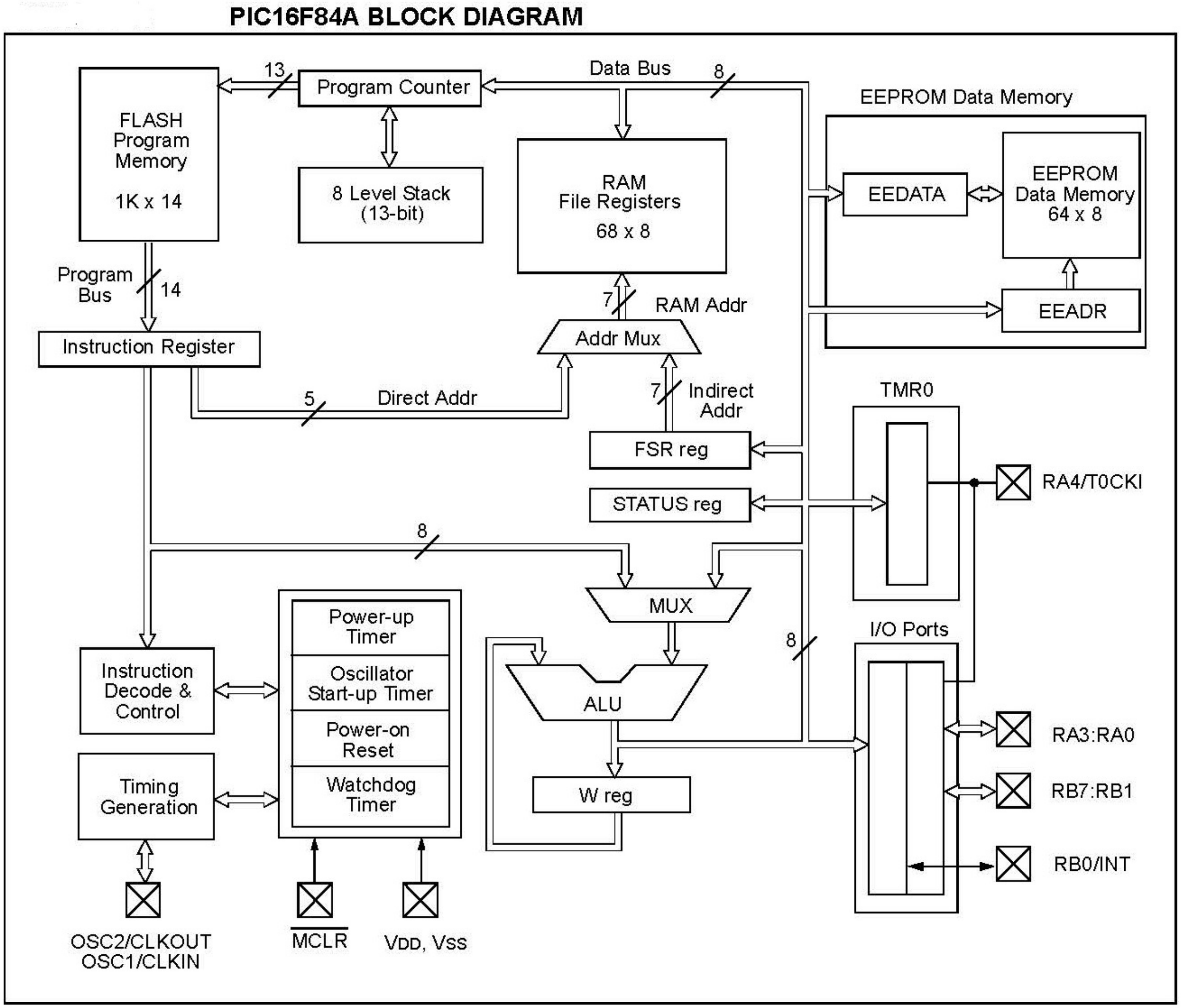
```
ldaa #imm<8>
```

1000 0110

k k k k k k k k

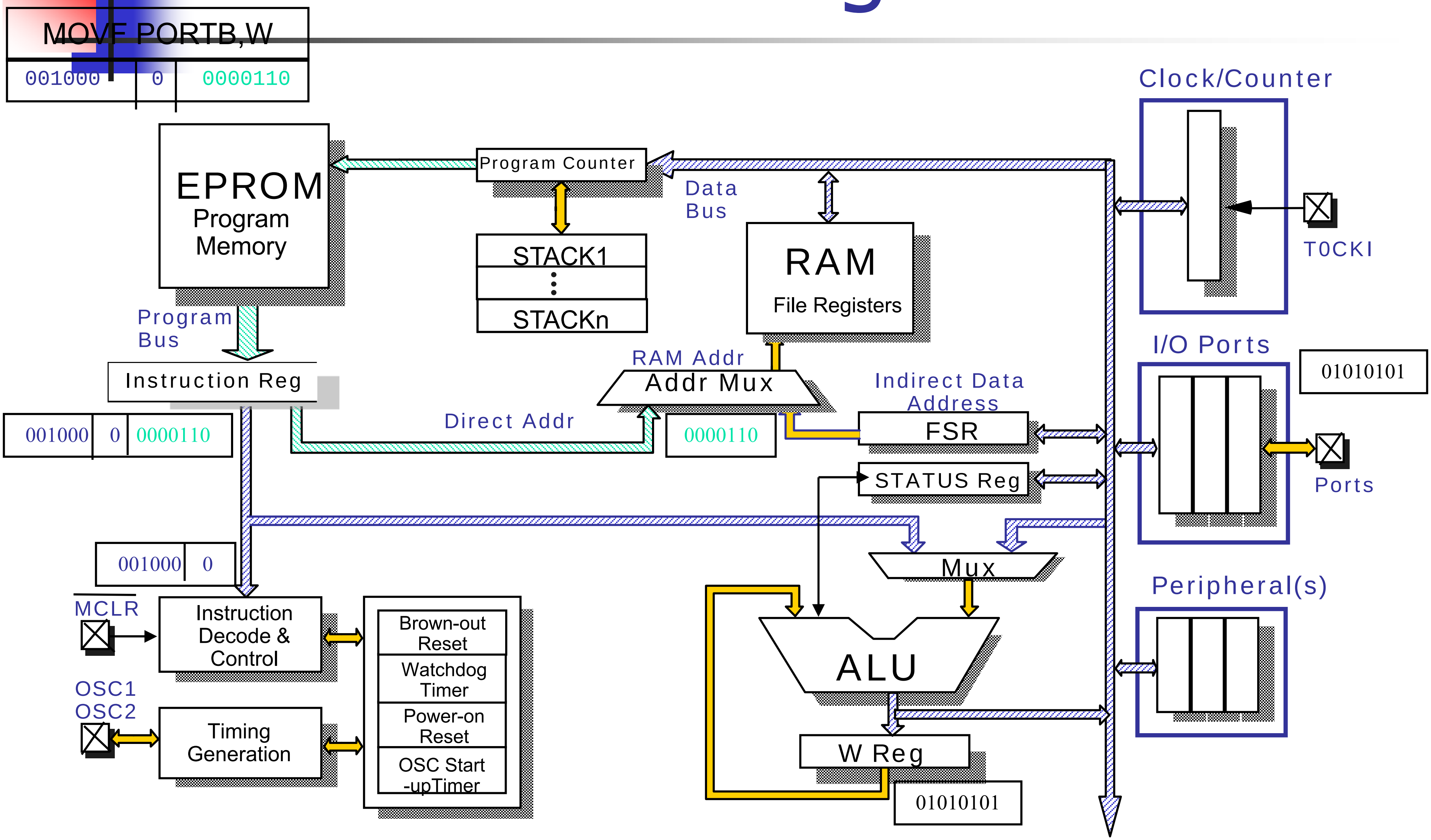
PICmicro® MCU Architecture

Block Diagram of PIC16F84A



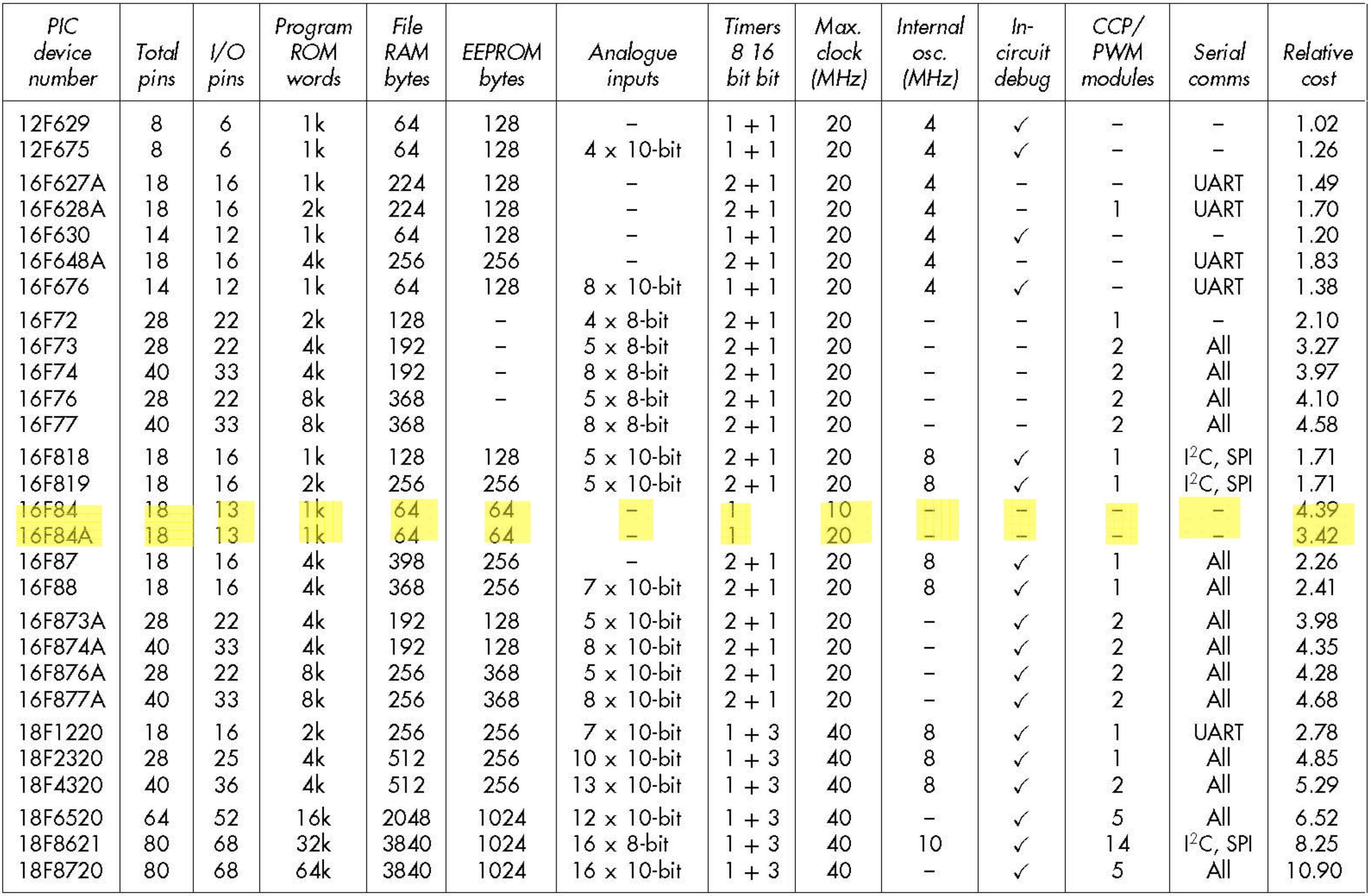
Lecture_2

PICmicro® MCU Architecture Block Diagram



PICmicro® MCU Architecture Flash-Type Microcontrollers

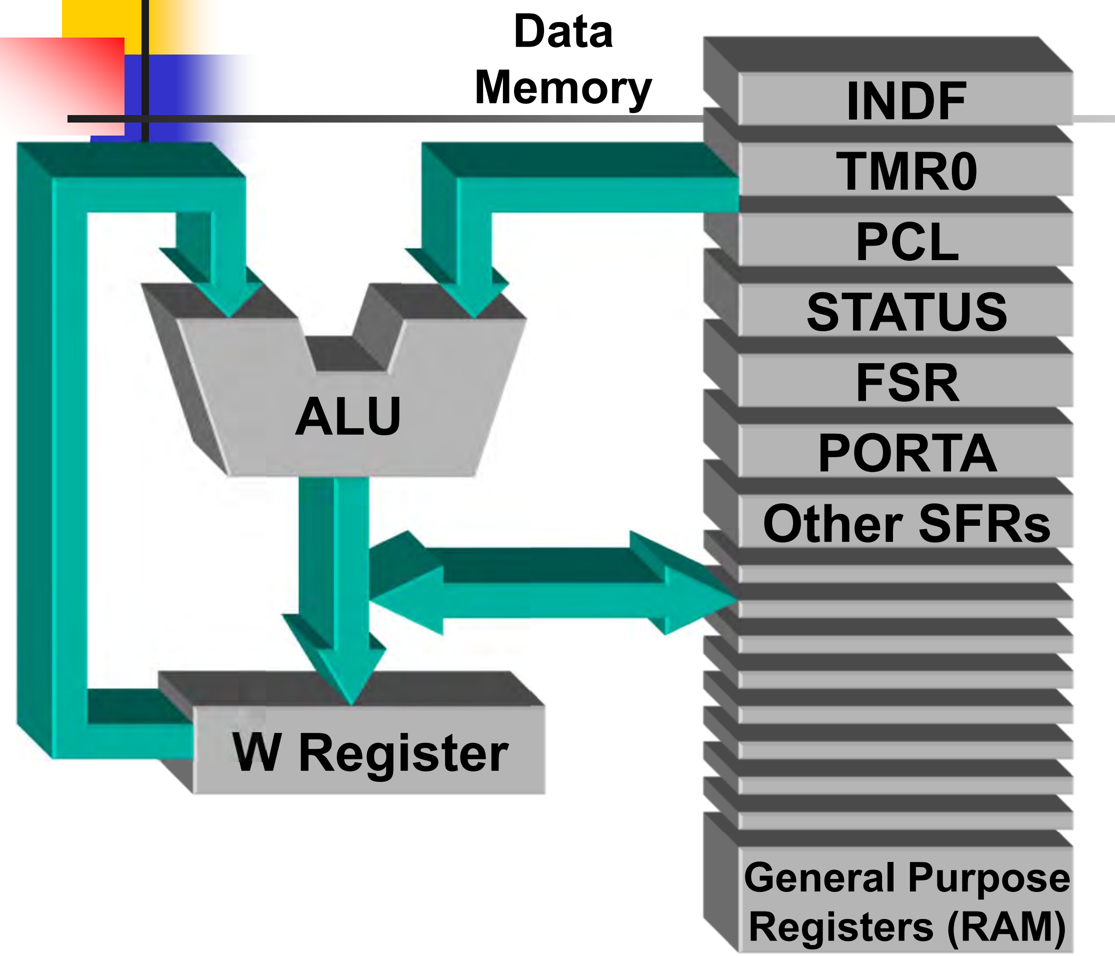
PIC flash microcontroller features



PIC device number	Total pins	I/O pins	Program ROM words	File RAM bytes	EEPROM bytes	Analogue inputs	Timers 8 16 bit bit	Max. clock (MHz)	Internal osc. (MHz)	In-circuit debug	CCP/ PWM modules	Serial comms	Relative cost
12F629	8	6	1k	64	128	–	1 + 1	20	4	✓	–	–	1.02
12F675	8	6	1k	64	128	4 x 10-bit	1 + 1	20	4	✓	–	–	1.26
16F627A	18	16	1k	224	128	–	2 + 1	20	4	–	–	UART	1.49
16F628A	18	16	2k	224	128	–	2 + 1	20	4	–	1	UART	1.70
16F630	14	12	1k	64	128	–	1 + 1	20	4	✓	–	–	1.20
16F648A	18	16	4k	256	256	–	2 + 1	20	4	–	–	UART	1.83
16F676	14	12	1k	64	128	8 x 10-bit	1 + 1	20	4	✓	–	UART	1.38
16F72	28	22	2k	128	–	4 x 8-bit	2 + 1	20	–	–	1	–	2.10
16F73	28	22	4k	192	–	5 x 8-bit	2 + 1	20	–	–	2	All	3.27
16F74	40	33	4k	192	–	8 x 8-bit	2 + 1	20	–	–	2	All	3.97
16F76	28	22	8k	368	–	5 x 8-bit	2 + 1	20	–	–	2	All	4.10
16F77	40	33	8k	368		8 x 8-bit	2 + 1	20	–	–	2	All	4.58
16F818	18	16	1k	128	128	5 x 10-bit	2 + 1	20	8	✓	1	I^{2}C, SPI	1.71
16F819	18	16	2k	256	256	5 x 10-bit	2 + 1	20	8	✓	1	I^{2}C, SPI	1.71
16F84	18	13	1k	64	64	–	1	10	–	–	–	–	4.39
16F84A	18	13	1k	64	64	–	1	20	–	–	–	–	3.42
16F87	18	16	4k	398	256	–	2 + 1	20	8	✓	1	All	2.26
16F88	18	16	4k	368	256	7 x 10-bit	2 + 1	20	8	✓	1	All	2.41
16F873A	28	22	4k	192	128	5 x 10-bit	2 + 1	20	–	✓	2	All	3.98
16F874A	40	33	4k	192	128	8 x 10-bit	2 + 1	20	–	✓	2	All	4.35
16F876A	28	22	8k	256	368	5 x 10-bit	2 + 1	20	–	✓	2	All	4.28
16F877A	40	33	8k	256	368	8 x 10-bit	2 + 1	20	–	✓	2	All	4.68
18F1220	18	16	2k	256	256	7 x 10-bit	1 + 3	40	8	✓	1	UART	2.78
18F2320	28	25	4k	512	256	10 x 10-bit	1 + 3	40	8	✓	1	All	4.85
18F4320	40	36	4k	512	256	13 x 10-bit	1 + 3	40	8	✓	2	All	5.29
18F6520	64	52	16k	2048	1024	12 x 10-bit	1 + 3	40	–	✓	5	All	6.52
18F8621	80	68	32k	3840	1024	16 x 8-bit	1 + 3	40	10	✓	14	I^{2}C, SPI	8.25
18F8720	80	68	64k	3840	1024	16 x 10-bit	1 + 3	40	–	✓	5	All	10.90

PICmicro® MCU Architecture

Register File Concept



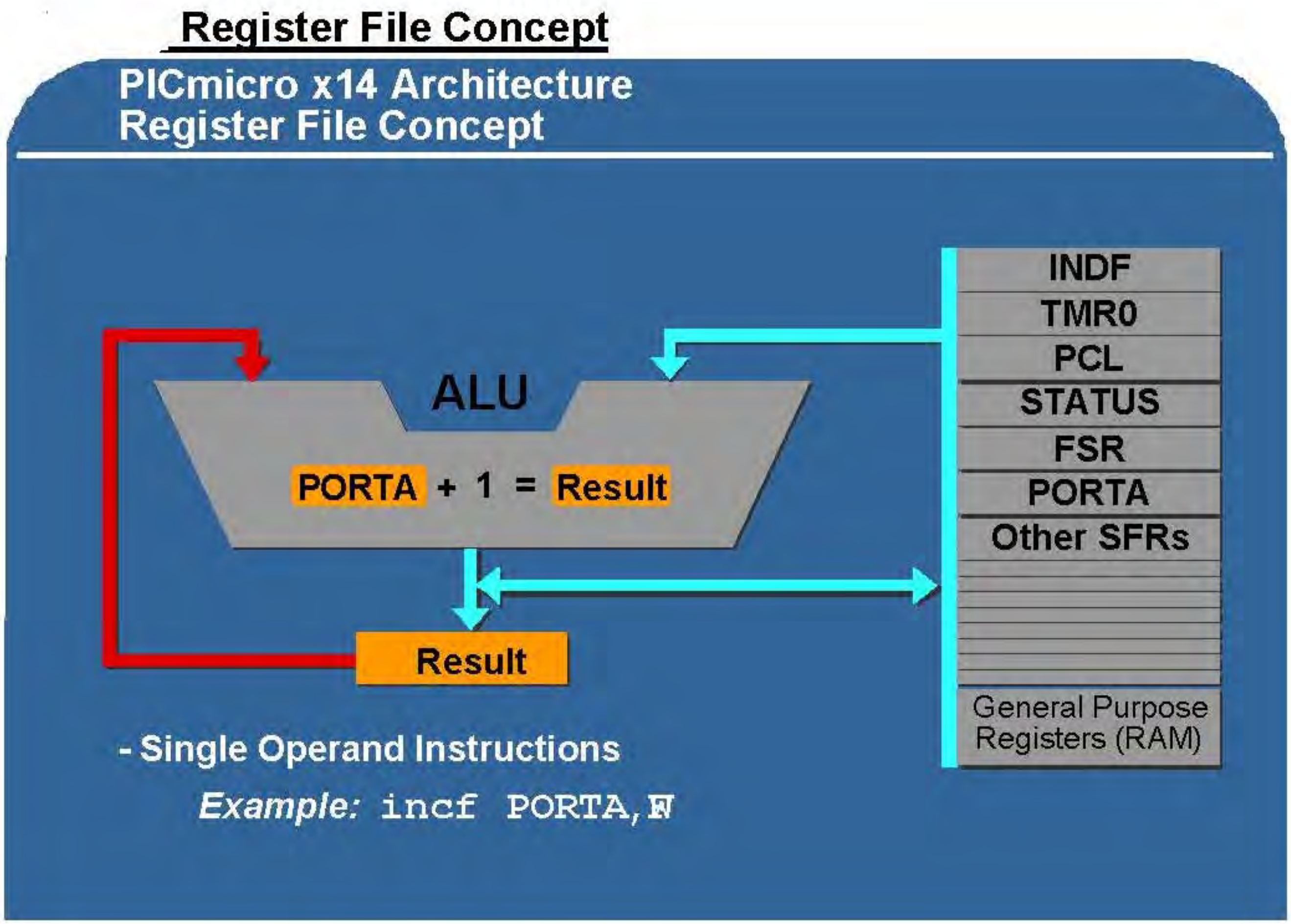
- RAM is a bank of general purpose registers
- Peripherals (I/O) are registers
- All instructions operate on any register
- Long word instruction allows direct addressing of registers

14-bit Instruction Format Example:

Op code <7>	**Direct data addr <7>**

PICmicro® MCU Architecture

Register File Concept



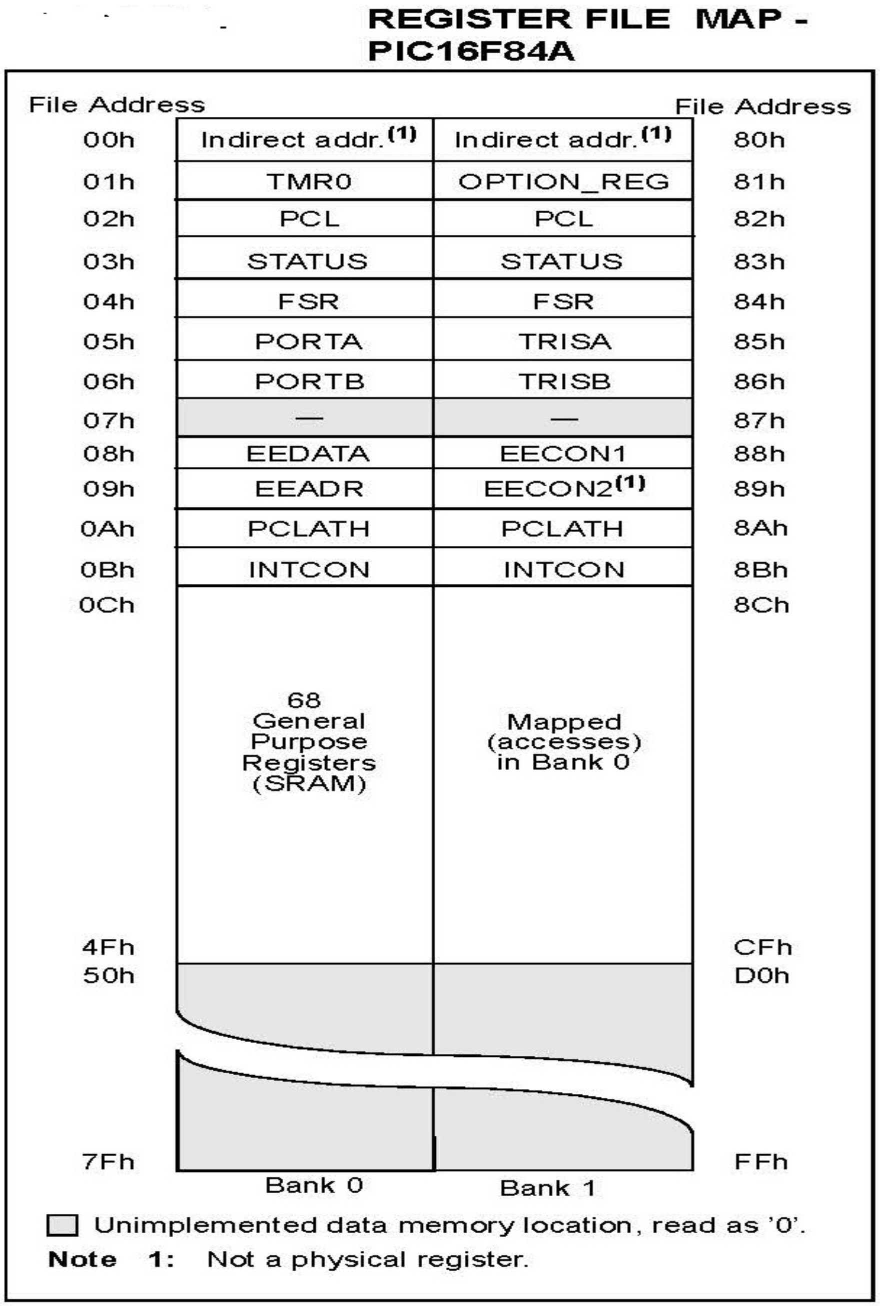
Register File Concept

PICmicro x14 Architecture
Register File Concept

- Single Operand Instructions

Example: `incf PORTA,F`

PIC 16F84 File Register Set



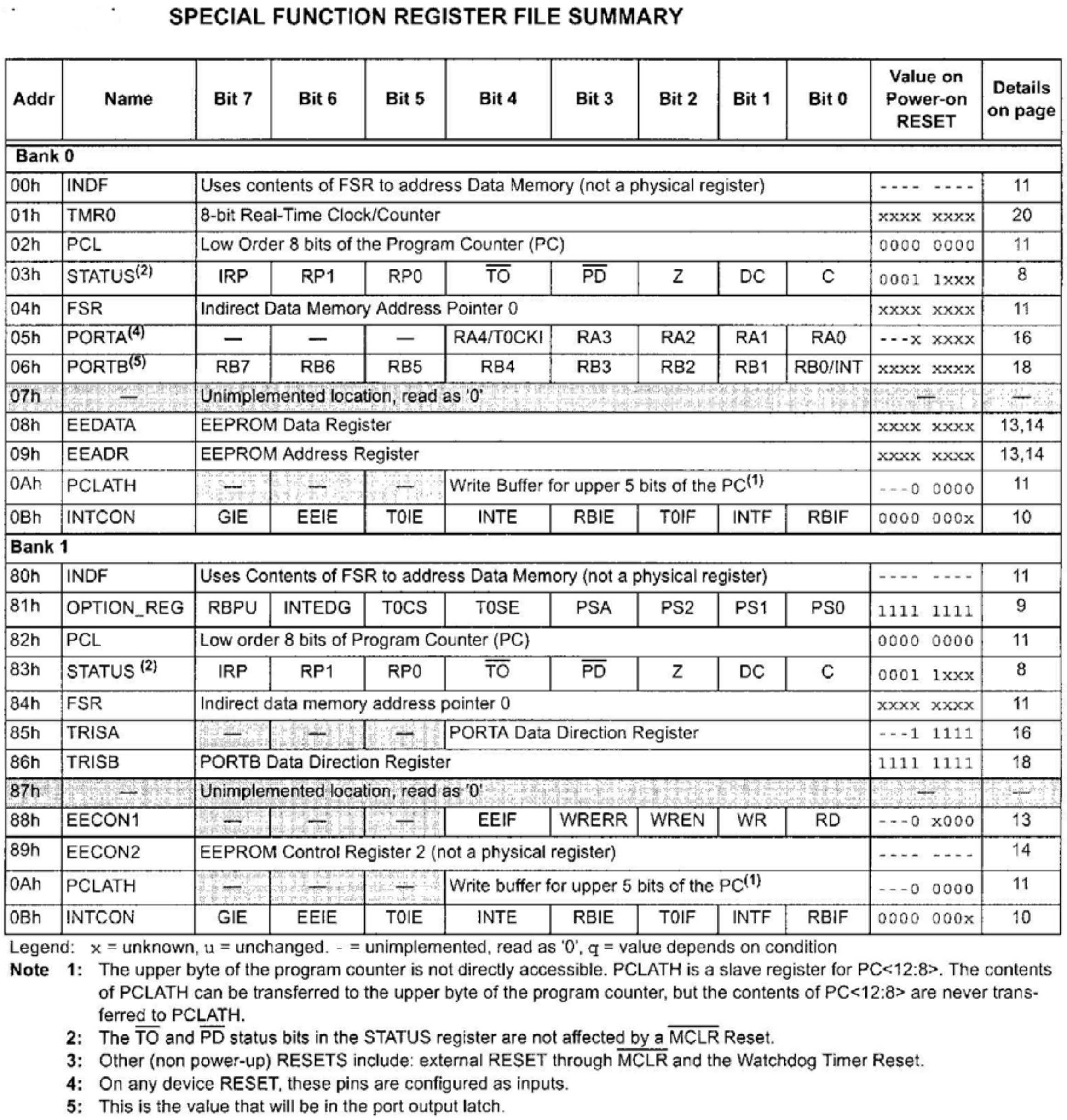
REGISTER FILE MAP - PIC16F84A

File Address			File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch 4Fh	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch CFh
50h 7Fh	▨	▨	D0h FFh
	Bank 0	Bank 1	

▨ Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

PIC 16F84 File Register Set



SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS[2]	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA[4]	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB[5]	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC[1]					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS [2]	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC[1]					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged. - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ status bits in the STATUS register are not affected by a $\overline{\text{MCLR}}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{\text{MCLR}}$ and the Watchdog Timer Reset.

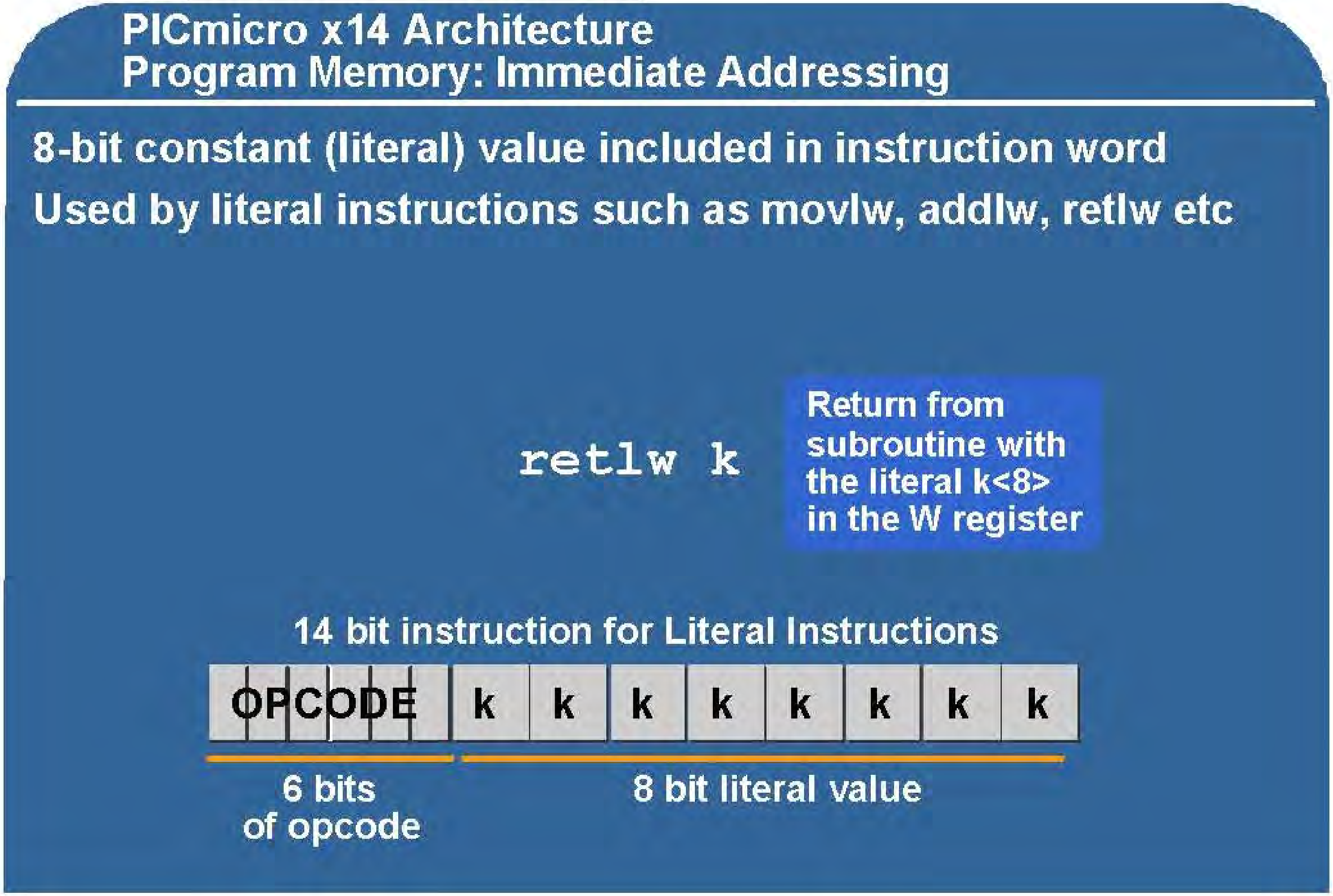
4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

PICmicro® MCU Architecture

Data Memory: Immediate Addressing

Immediate Addressing



PICmicro x14 Architecture
Program Memory: Immediate Addressing

8-bit constant (literal) value included in instruction word

Used by literal instructions such as movlw, addlw, retlw etc

```
retlw k
```

Return from subroutine with the literal k<8> in the W register

14 bit instruction for Literal Instructions

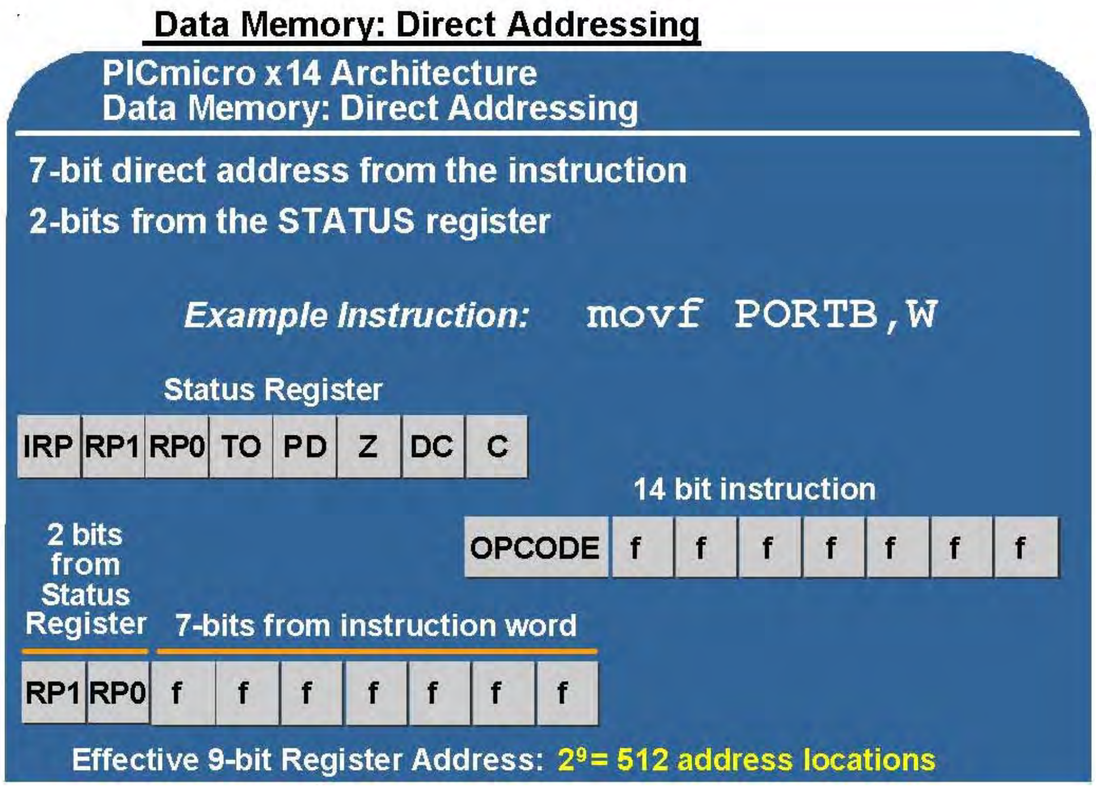
OPCODE	k	k	k	k	k	k	k	k

6 bits of opcode

8 bit literal value

PICmicro® MCU Architecture

Data Memory: Direct Addressing



Data Memory: Direct Addressing

PICmicro x14 Architecture
Data Memory: Direct Addressing

7-bit direct address from the instruction

2-bits from the STATUS register

Example Instruction: `movf PORTB,W`

Status Register

IRP	RP1	RP0	TO	PD	Z	DC	C

14 bit instruction

OPCODE	f	f	f	f	f	f	f

2 bits from Status Register

7-bits from instruction word

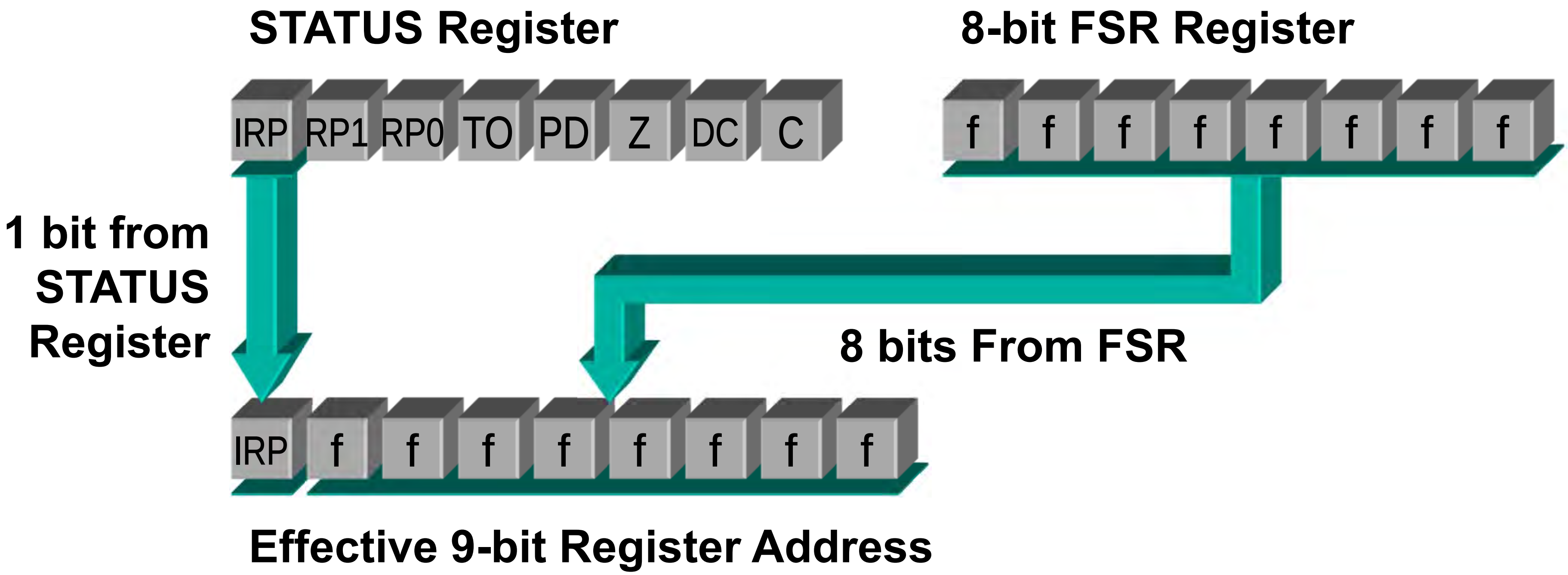
RP1	RP0	f	f	f	f	f	f	f

Effective 9-bit Register Address: 2^9 = 512 address locations

PICmicro® MCU Architecture

Data Memory: Indirect Addressing

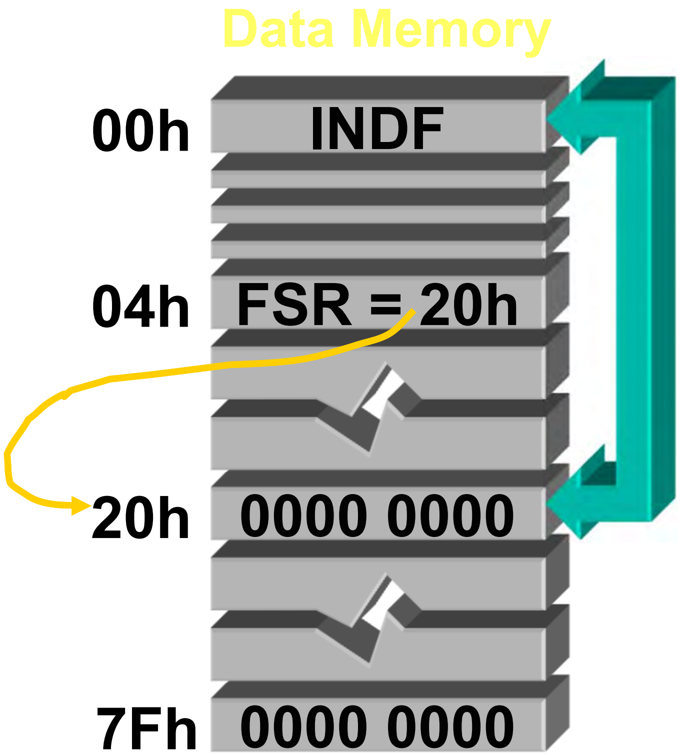
- 8-bit indirect address from the FSR (File Select Register)
- 1-bit from STATUS register



PICmicro® MCU Architecture

Data Memory: Indirect Addressing

- Clear all RAM locations from 0x20 to 0x7F
    - Indirect address is loaded into FSR
    - Every time INDF is used as operand, register pointed to by FSR is actually used



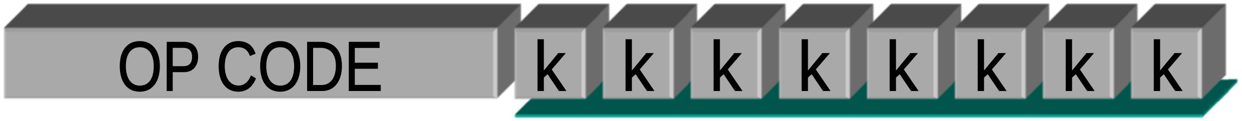
```
     movlw      0x20
     movwf      FSR
LOOP clrf       INDF
     incf       FSR,F
     btfss      FSR,7
     goto       LOOP
     <next instruction>
```

PICmicro® MCU Architecture

Data Memory: Immediate Addressing

- 8-bit constant (literal) value included in instruction word
- Used by literal instructions such as movlw, addlw, retlw, etc.

14-bit Instruction for Literal Instructions

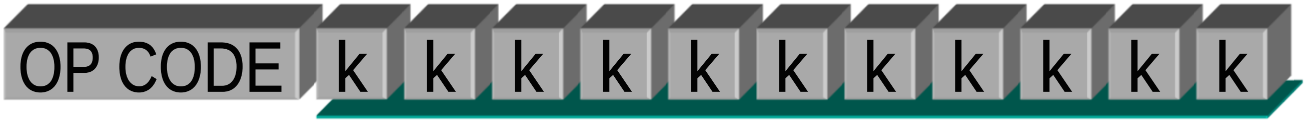


PICmicro® MCU Architecture

Data Memory: PC Absolute Addressing

- Used by control instructions CALL and GOTO to modify the PC (Program Counter)
- A constant value may also be written directly to the PC (next slide)

14-bit Instruction for CALL and GOTO



OP CODE k k k k k k k k k k k

This is for 2K addressing

PICmicro® MCU Architecture

Data Memory: PC Relative Addressing

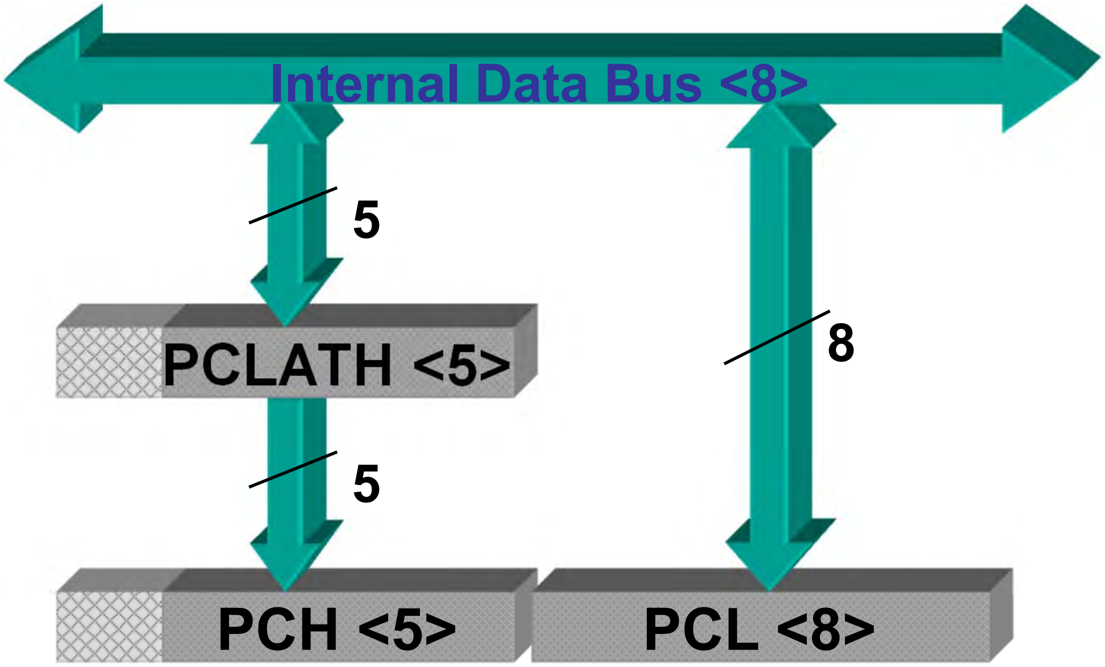
- PC Relative Addressing:
- Used to perform a computed goto by adding an offset directly to the 13-bit Program Counter (8K addressing)
- To write to PC - Applies when PC is the destination of an operation’s result

PICmicro® MCU Architecture

Data Memory: PC Relative Addressing

- First write high byte to PCLATH
- Next write low byte to PCL--this loads the entire 13-bit value to PC

- To read the PC
  - Read low byte from PCL
  - PCLATH is *NOT* loaded with value from PCH



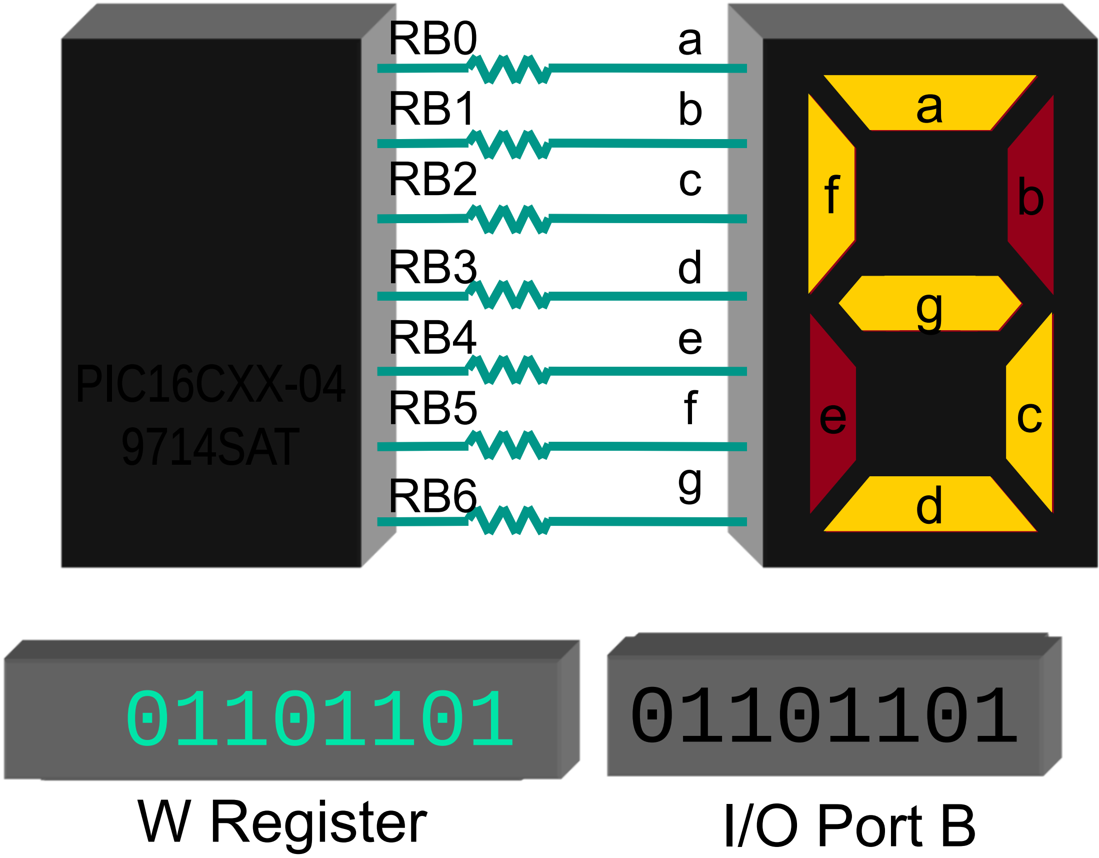
13-Bit Program Counter

PICmicro® MCU Architecture

Data Memory: PC Relative Addressing

```
    org     0x10
    clrf    PCLATH
    movf    DisplayValue,W
    call    SevenSegmentDecode
    movwf   PORTB
    goto    Continue
SevenSegmentDecode
    addwf   PCL,F
    retlw   B'00111111' ;decode 0
    retlw   B'00000110' ;decode 1
    retlw   B'01011011' ;decode 2
    retlw   B'01001111' ;decode 3
    retlw   B'01100110' ;decode 4
    retlw   B'01101101' ;decode 5
    retlw   B'01111101' ;decode 6
    retlw   B'00000111' ;decode 7
    retlw   B'01111111' ;decode 8
    retlw   B'01101111' ;decode 9
Continue
```

- Look-up Table Example



Lecture_3

Instruction Set
(x14)

PICmicro® MCU Instruction Set

PICmicro® x14 Instruction Set

- 12-bit core → 33 instructions
- 14-bit core → 35 instructions
- 16-bit enhanced core → 77 instructions
  - Easy to learn
  - High compaction
  - Very powerful single-word instructions
  - All single- cycle except program branches
  - Upward compatibility of instructions

PICmicro® MCU Instruction Set

Summary

Byte-Oriented Operations

NOP	-	No Operation
MOVWF	f	Move W to f
MOVF	f,d	Move f
CLRW	-	Clear W
CLRF	f	Clear f
INCF	f,d	Increment f
DECF	f,d	Decrement f
ADDWF	f,d	Add W and f
SUBWF	f,d	Subtract W from f
ANDWF	f,d	AND W and f
IORWF	f,d	Inclusive OR W and f
XORWF	f,d	Exclusive OR W and f
COMF	f,d	Complement f
RRF	f,d	Rotate right f through carry
RLF	f,d	Rotate left f through carry
INCFSZ	f,d	Increment f, skip if zero
DECFSZ	f,d	Decrement f, skip if zero
SWAPF	f,d	Swap nibbles of f

Bit-Oriented Operations

BCF	f,b	Bit clear f
BSF	f,b	Bit set f
BTFSC	f,b	Bit test f, skip if clear
BTFSS	f,b	Bit test f, skip if set

Literal and Control Operations

SLEEP	-	Go into standby mode
CLRWDT	-	Clear watchdog timer
RETLW	k	Return, place literal in W
RETFIE	-	Return from interrupt
RETURN	-	Return from subroutine
CALL	k	Call subroutine
GOTO	k	Go to address (k is 9bit)
MOVLW	k	Move literal to W
IORLW	k	Inclusive OR literal with W
ADDLW	k	Add literal with W
SUBLW	k	Subtract W from literal
ANDLW	k	AND literal with W
XORLW	k	Exclusive OR literal with W

f = File Register, k = literal value (8 bit), b = bit address <0,7>, d = destination (0=W, 1=f)

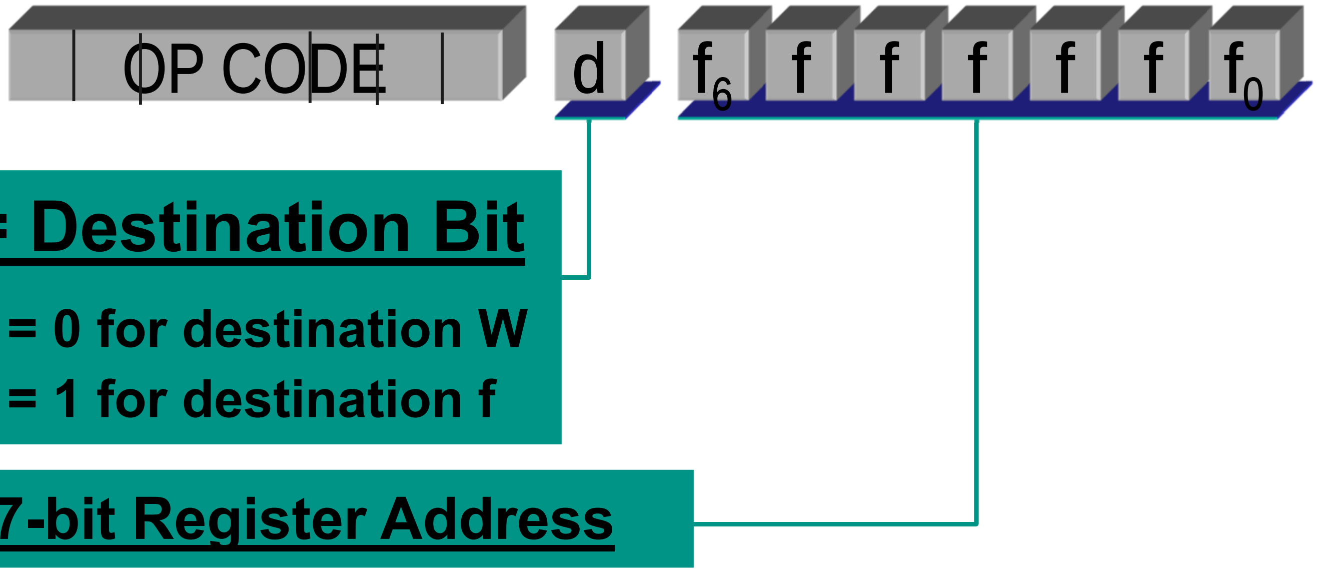
PICmicro® MCU Instruction Set

Byte-Oriented Operations

Byte-Oriented Operations

NOP	-
MOVWF	f
CLRW	-
CLRF	f
SUBWF	f,d
DECF	f,d
IORWF	f,d
ANDWF	f,d
XORWF	f,d
ADDWF	f,d
MOVF	f,d
COMF	f,d
INCF	f,d
DECFSZ	f,d
RRF	f,d
RLF	f,d
SWAPF	f,d
INCFSZ	f,d

14-bit Instruction for Byte Oriented Operations



d = Destination Bit

d = 0 for destination W

d = 1 for destination f

f = 7-bit Register Address

Example:

ADDWF REG, W

ADDWF f, d

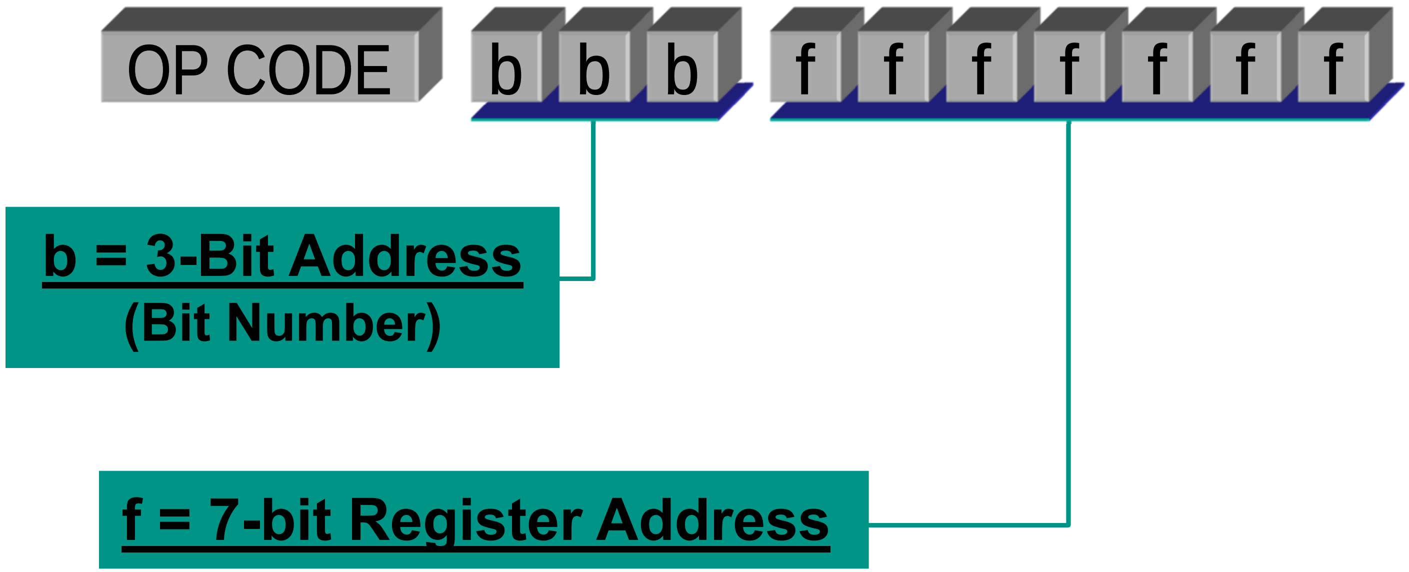
PICmicro® MCU Instruction Set

Bit-Oriented Operations

Bit-Oriented Operations

BCF	f,b
BSF	f,b
BTFSC	f,b
BTFSS	f,b

14-bit Instruction for Bit Oriented Operations



b = 3-Bit Address
(Bit Number)

f = 7-bit Register Address

Example:

BTFSC STATUS, C

BTFSC f, b

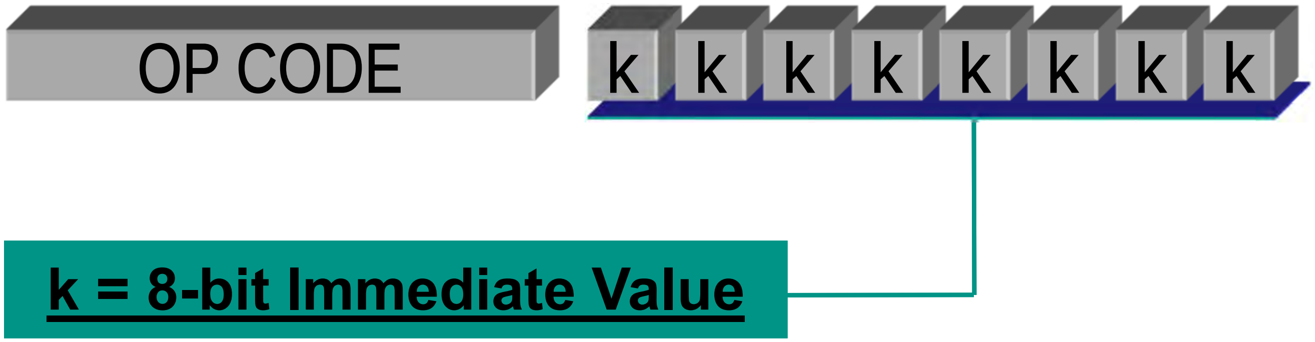
PICmicro® MCU Instruction Set

Literal and Control Operations

Literal and Control Operations

SLEEP	-
CLRWDT	-
RETLW	k
RETFIE	-
RETURN	-
CALL	k
GOTO	k
MOVLW	k
IORLW	k
ADDLW	k
SUBLW	k
ANDLW	k
XORLW	k

14-bit Instruction for Literal Operations



k = 8-bit Immediate Value

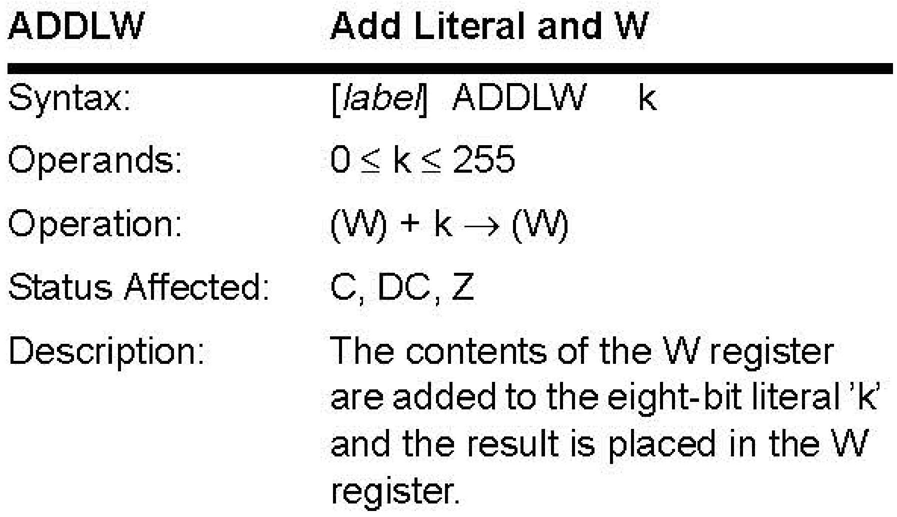
Example:

MOVLW 0x2F

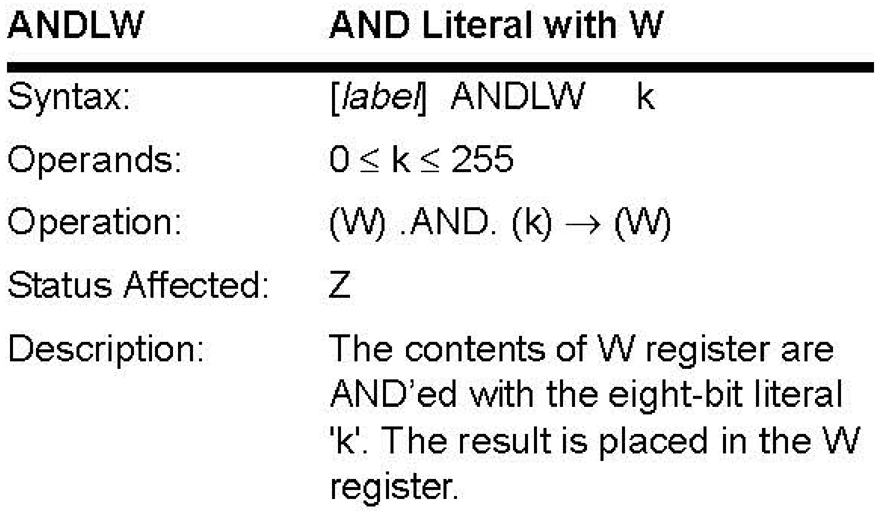
MOVLW k

14-Bit Core Instruction Set

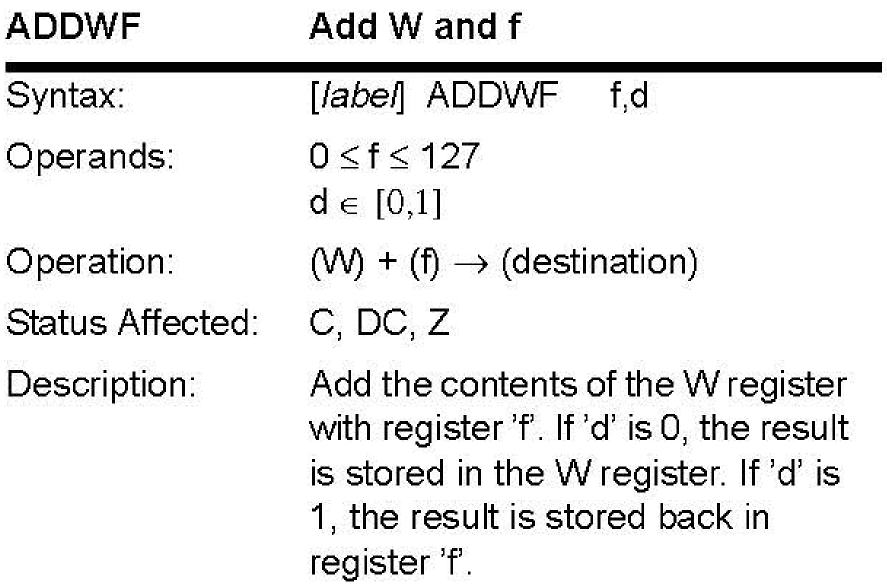
Examples



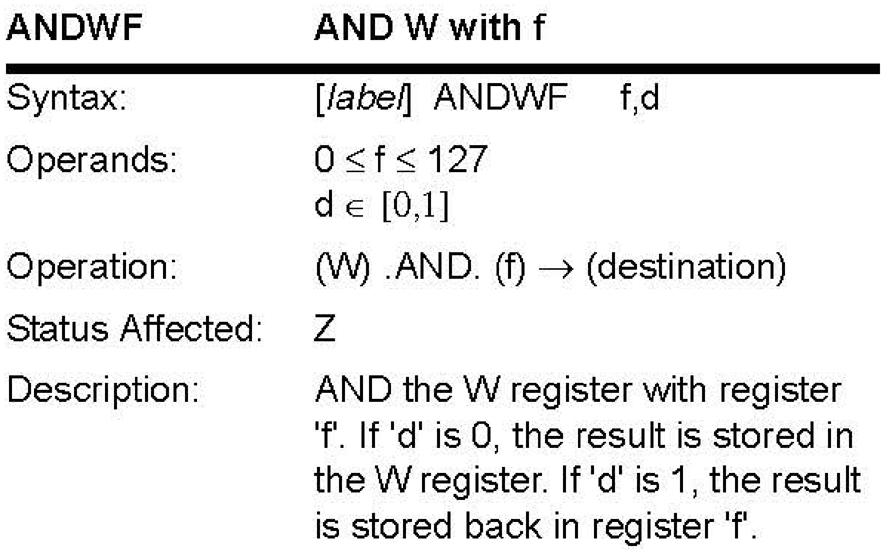
ADDLW	**Add Literal and W**
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.



ANDLW	**AND Literal with W**
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.



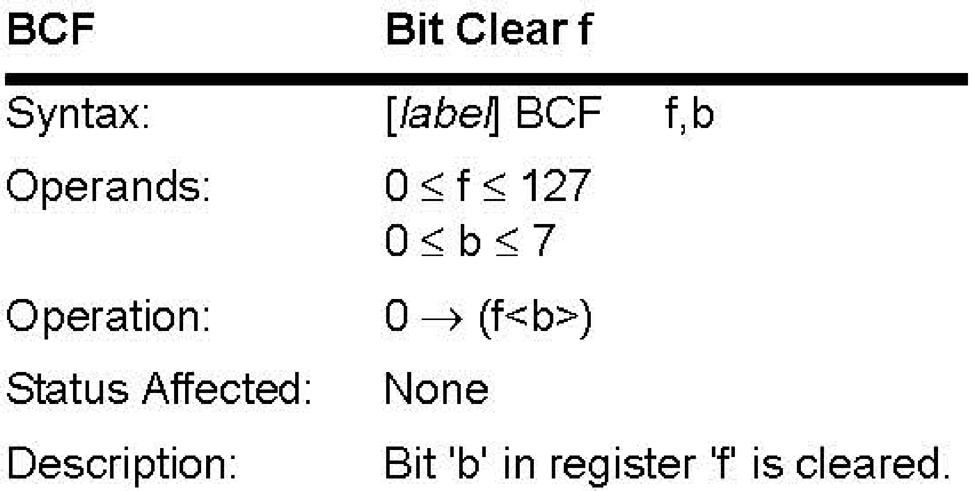
ADDWF	**Add W and f**
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.



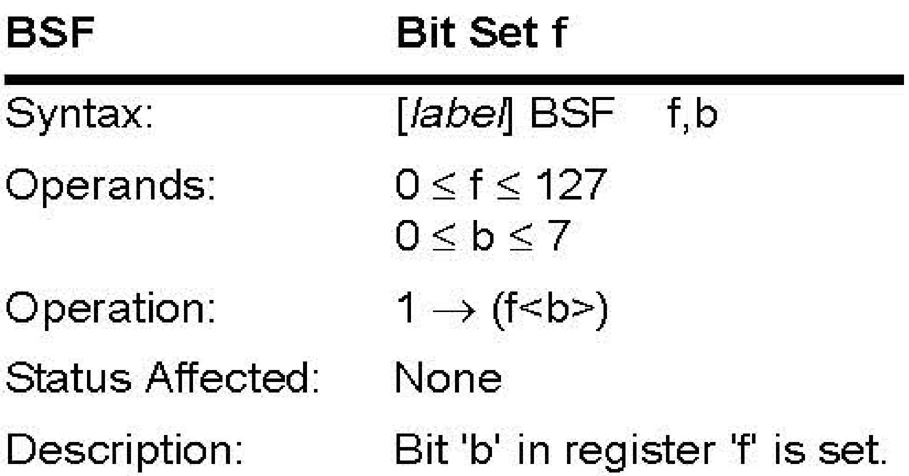
ANDWF	**AND W with f**
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

14-Bit Core Instruction Set

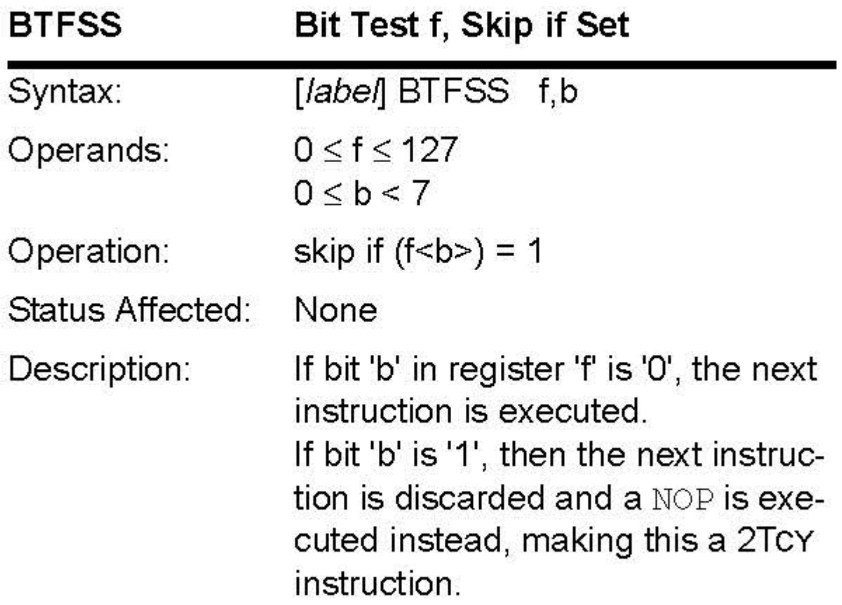
Examples



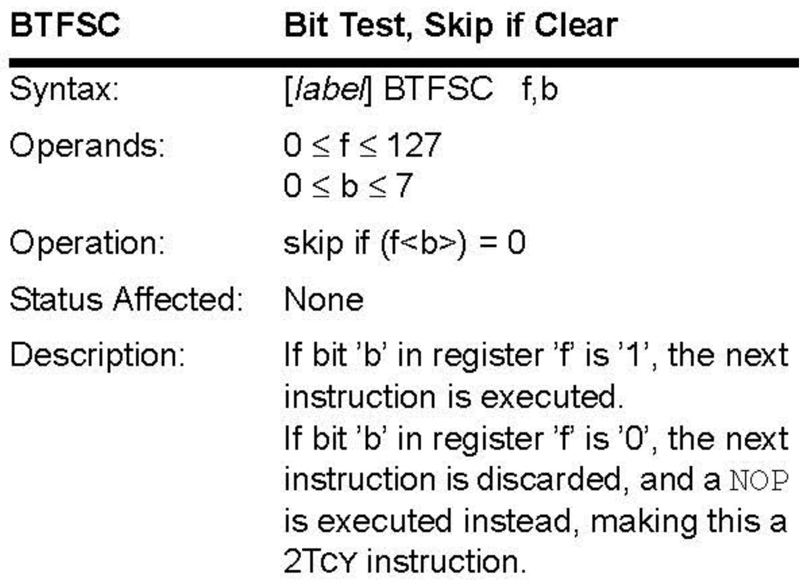
BCF	**Bit Clear f**
Syntax:	[*label*] BCF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.



BSF	**Bit Set f**
Syntax:	[*label*] BSF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.



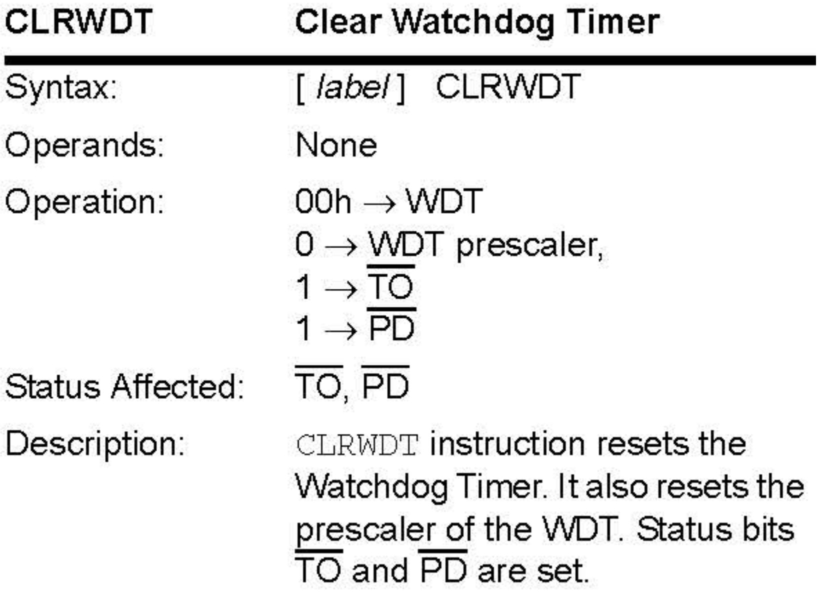
BTFSS	**Bit Test f, Skip if Set**
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b < 7$
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.



BTFSC	**Bit Test, Skip if Clear**
Syntax:	[*label*] BTFSC f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b' in register 'f' is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

14-Bit Core Instruction Set

Examples



CLRWDT **Clear Watchdog Timer**

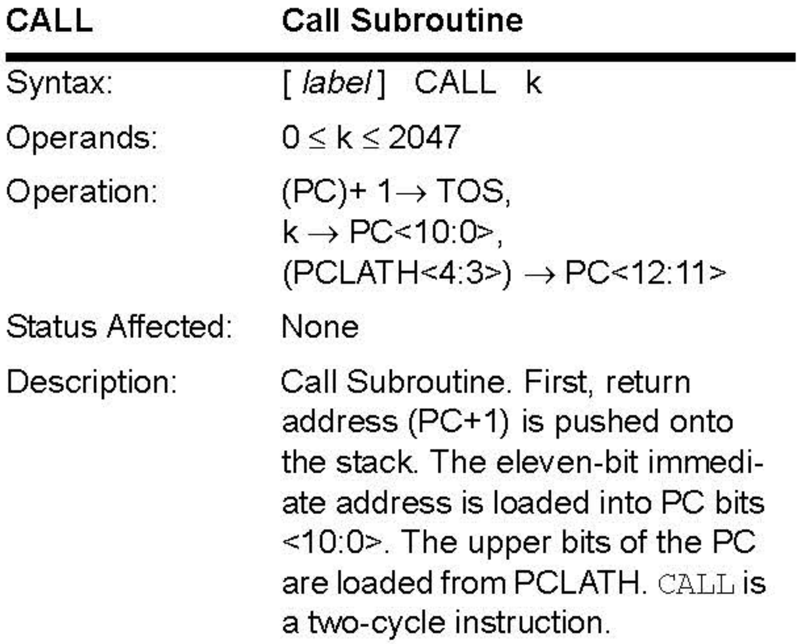
Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
0 → WDT prescaler,
1 → $\overline{TO}$
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.



CALL **Call Subroutine**

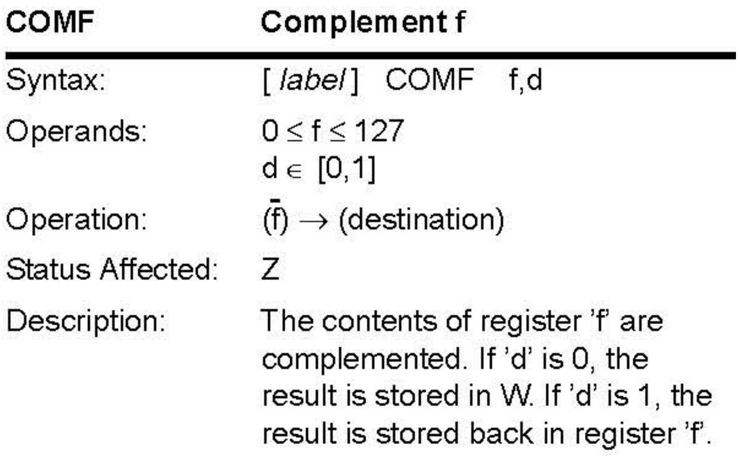
Syntax: [*label*] CALL k

Operands: 0 ≤ k ≤ 2047

Operation: (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

Status Affected: None

Description: Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.



COMF **Complement f**

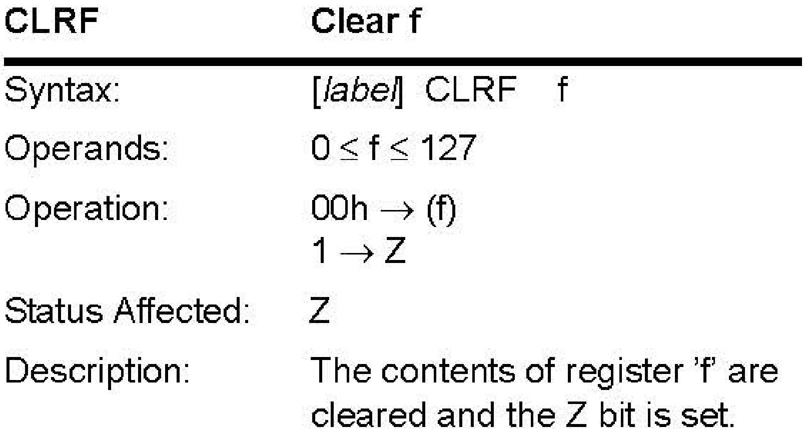
Syntax: [*label*] COMF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: $(\overline{f})$ → (destination)

Status Affected: Z

Description: The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.



CLRF **Clear f**

Syntax: [*label*] CLRF f

Operands: 0 ≤ f ≤ 127

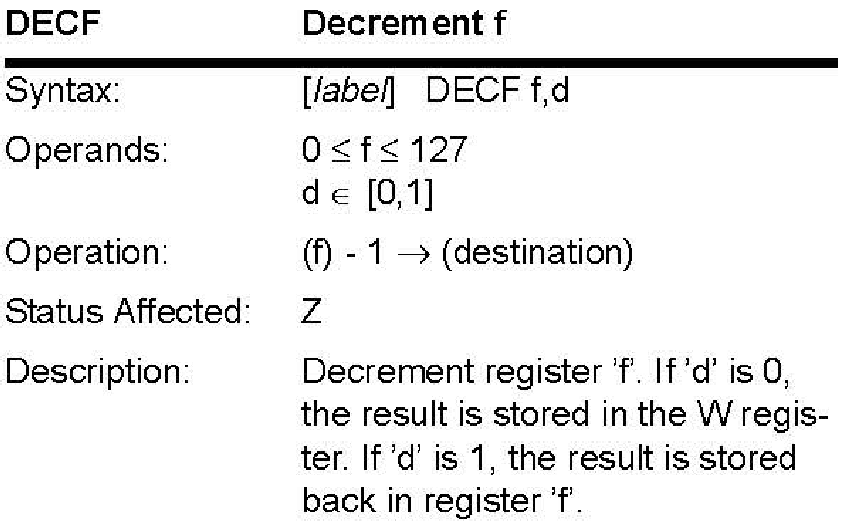
Operation: 00h → (f)
1 → Z

Status Affected: Z

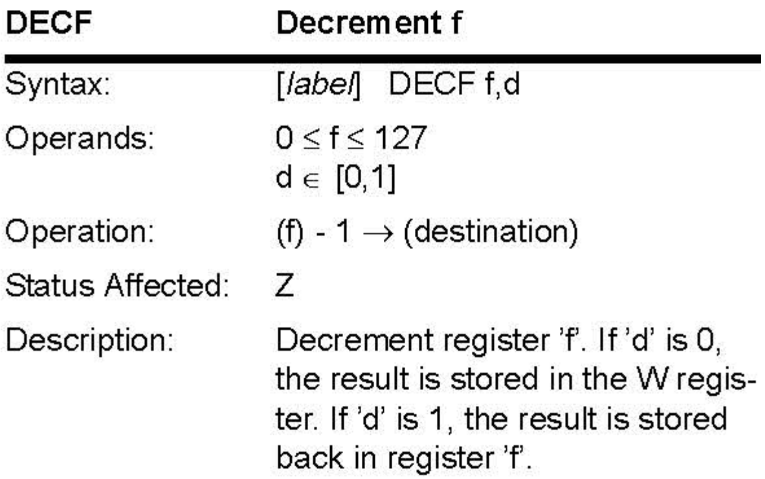
Description: The contents of register 'f' are cleared and the Z bit is set.

14-Bit Core Instruction Set

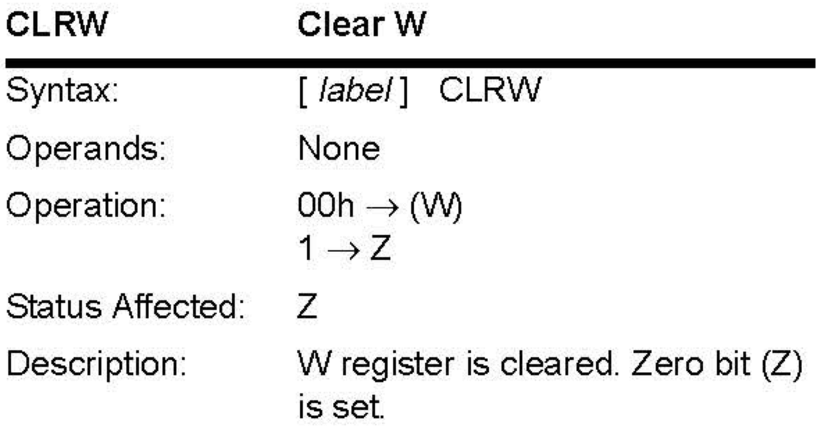
Examples



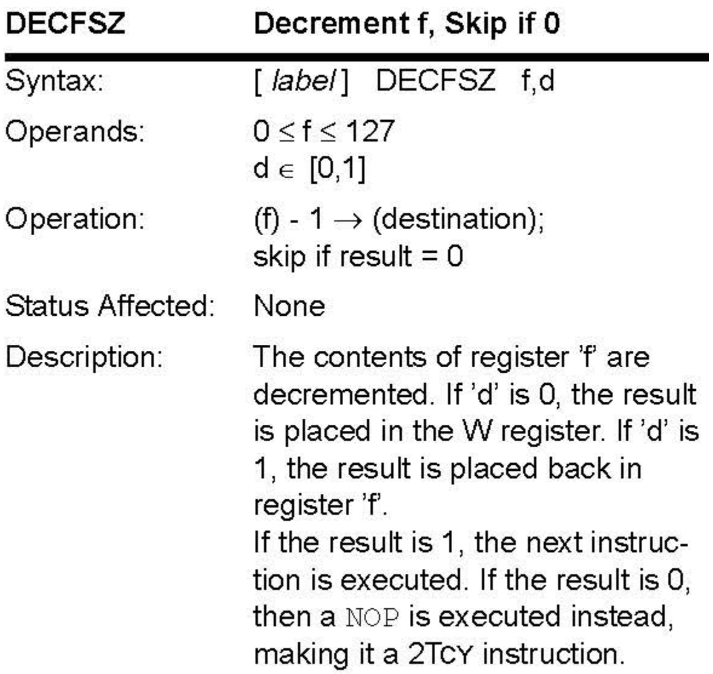
DECF	**Decrement f**
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register ’f’. If ’d’ is 0, the result is stored in the W register. If ’d’ is 1, the result is stored back in register ’f’.



DECF	**Decrement f**
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement 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’.



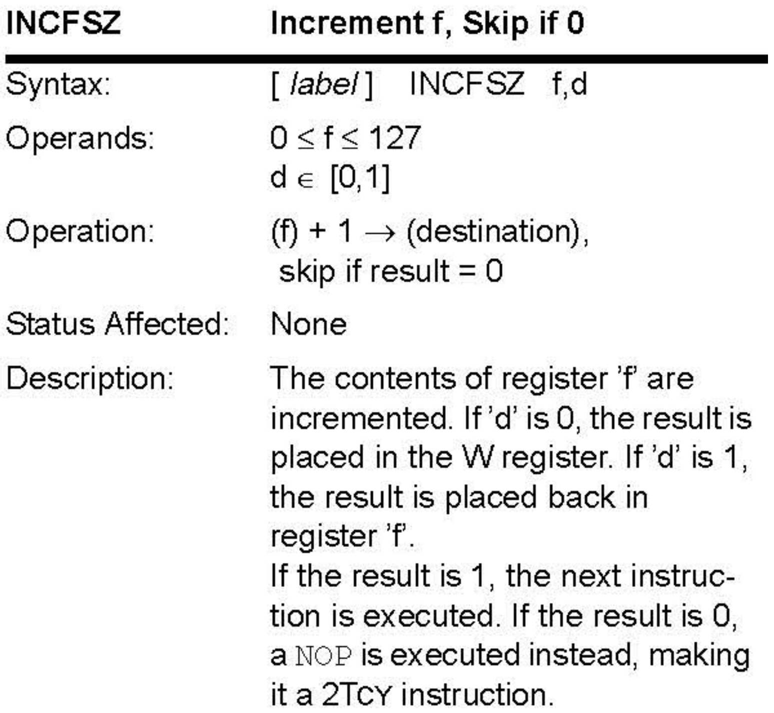
CLRW	**Clear W**
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.



DECFSZ	**Decrement f, Skip if 0**
Syntax:	[*label*] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register ’f’ are decremented. If ’d’ is 0, the result is placed in the W register. If ’d’ is 1, the result is placed back in register ’f’. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead, making it a 2TCY instruction.

14-Bit Core Instruction Set

Examples



INCFSZ **Increment f, Skip if 0**

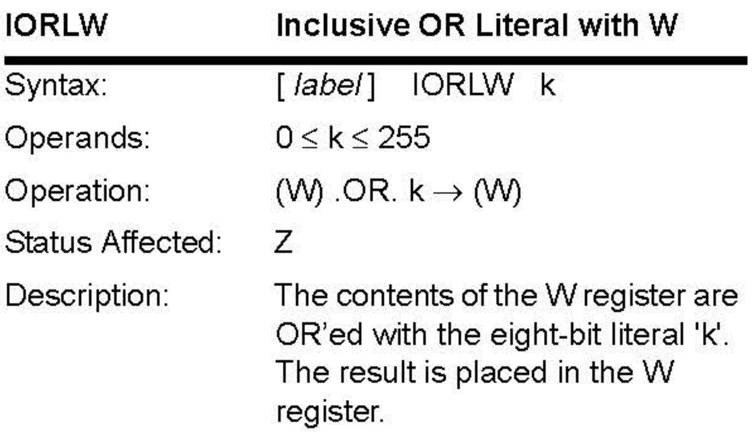
Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (destination),
skip if result = 0

Status Affected: None

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 1, the next instruction is executed. If the result is 0, a `NOP` is executed instead, making it a 2TCY instruction.



IORLW **Inclusive OR Literal with W**

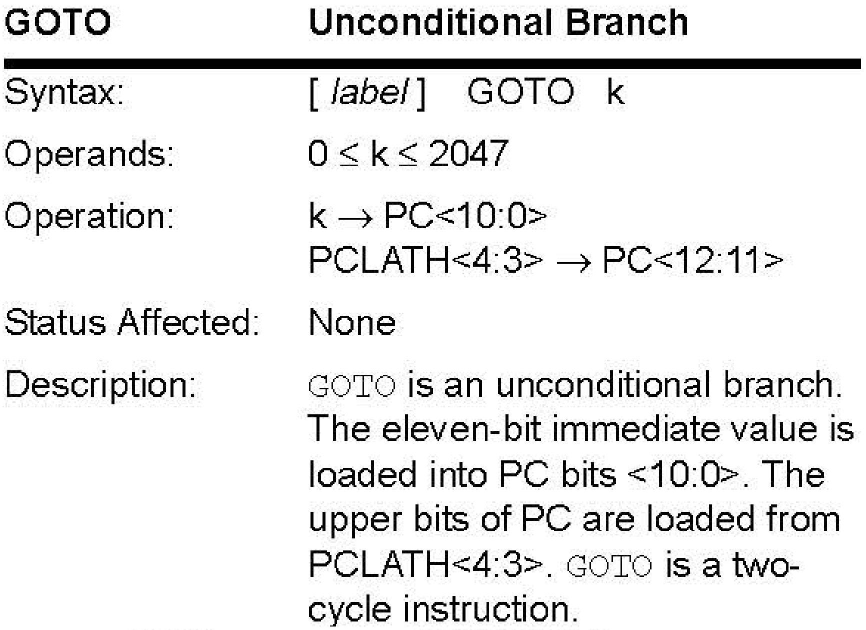
Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → (W)

Status Affected: Z

Description: The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register.



GOTO **Unconditional Branch**

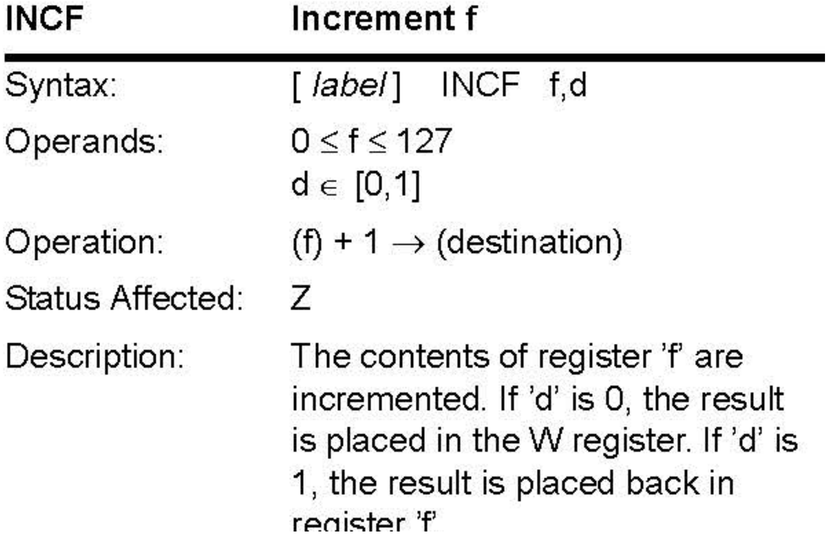
Syntax: [*label*] GOTO k

Operands: $0 \le k \le 2047$

Operation: k → PC<10:0>
PCLATH<4:3> → PC<12:11>

Status Affected: None

Description: `GOTO` is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. `GOTO` is a two-cycle instruction.



INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

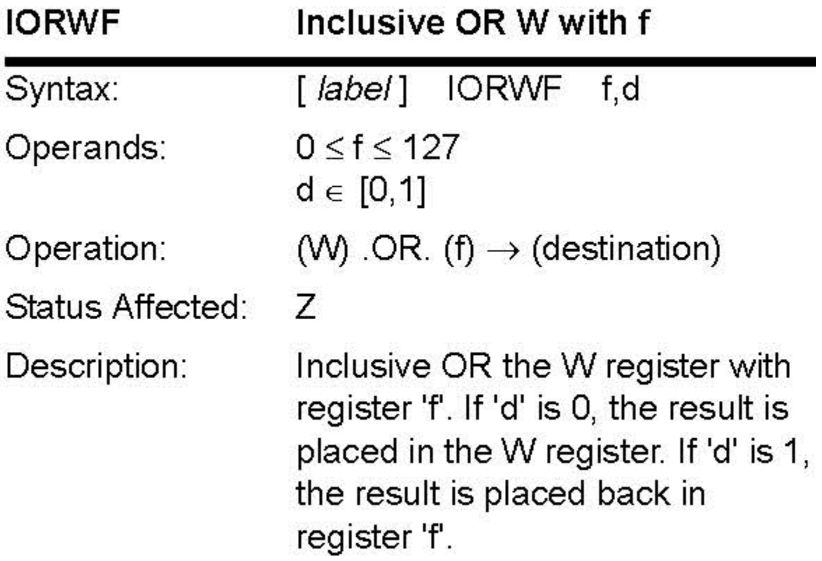
Operation: (f) + 1 → (destination)

Status Affected: Z

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

14-Bit Core Instruction Set

Examples



IORWF **Inclusive OR W with f**

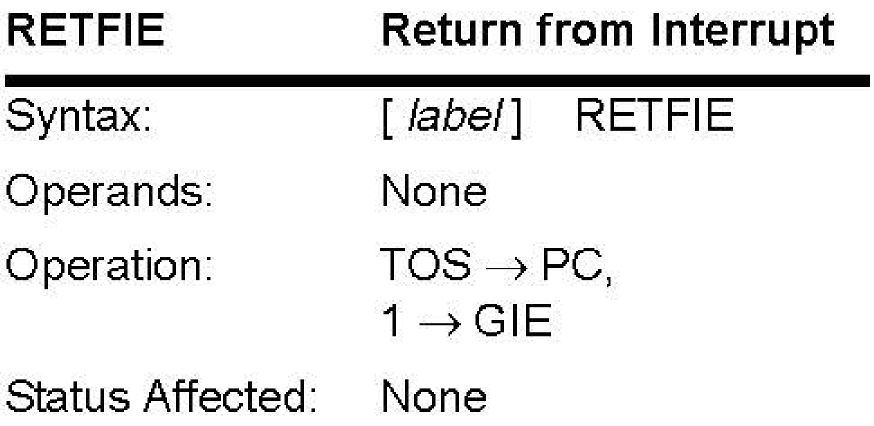
Syntax: [*label*] IORWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (W) .OR. (f) → (destination)

Status Affected: Z

Description: Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



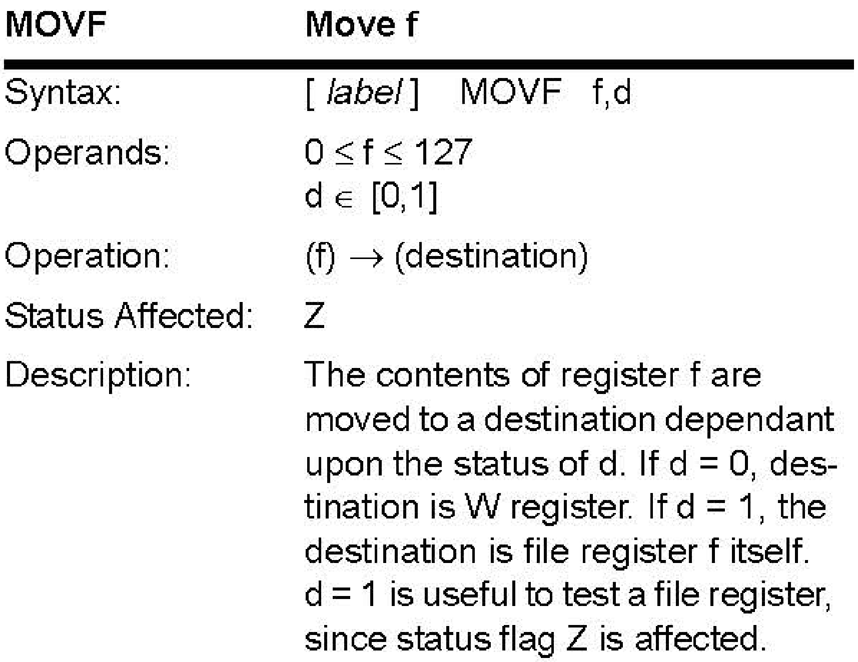
RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE

Operands: None

Operation: TOS → PC,
1 → GIE

Status Affected: None



MOVF **Move f**

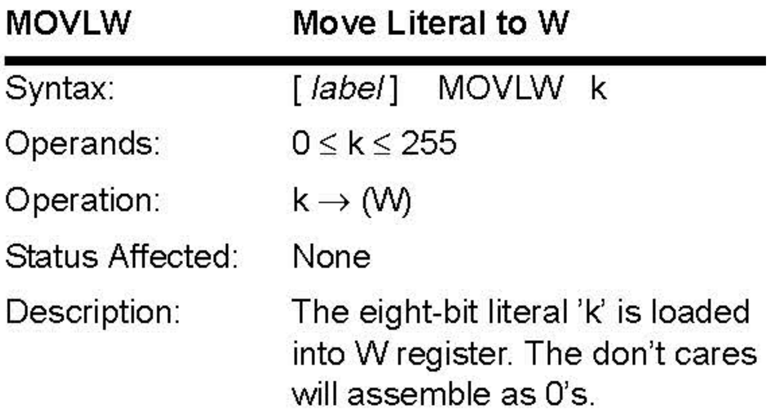
Syntax: [*label*] MOVF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) → (destination)

Status Affected: Z

Description: The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.



MOVLW **Move Literal to W**

Syntax: [*label*] MOVLW k

Operands: 0 ≤ k ≤ 255

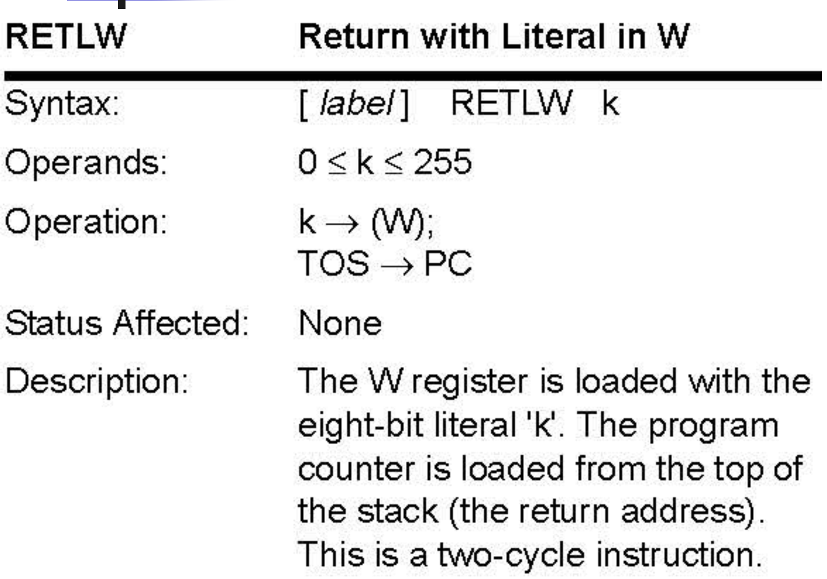
Operation: k → (W)

Status Affected: None

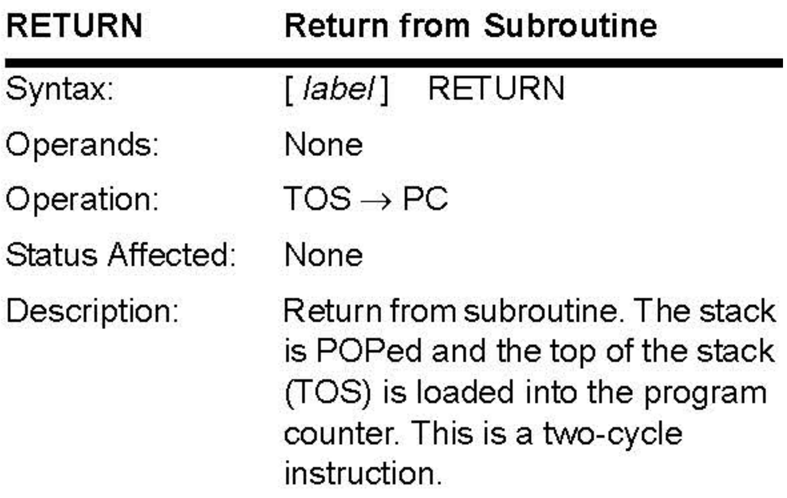
Description: The eight-bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

14-Bit Core Instruction Set

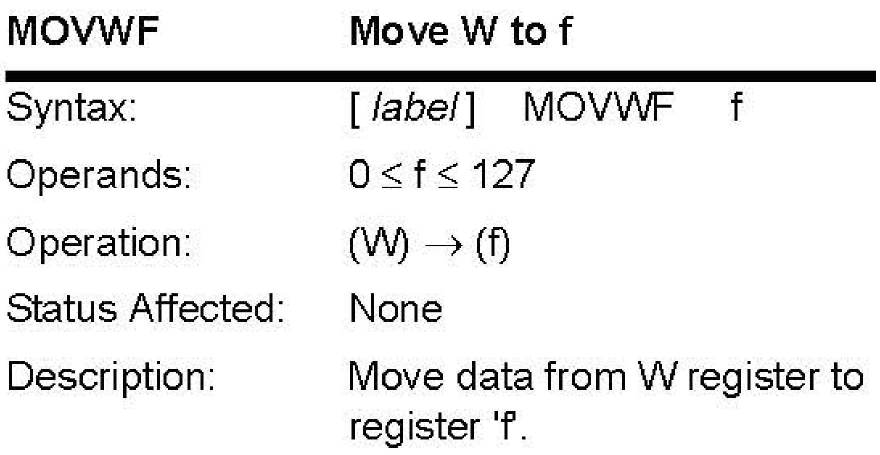
Examples



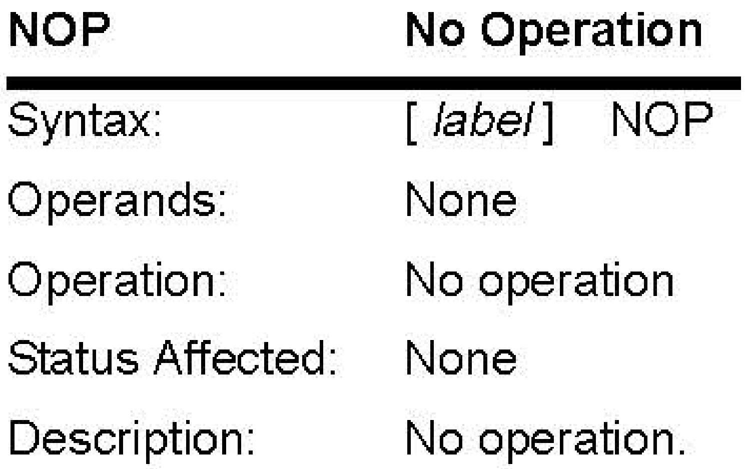
RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.



RETURN	Return from Subroutine
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.



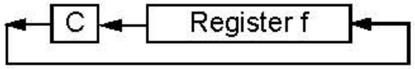
MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.



NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

14-Bit Core Instruction Set
Examples

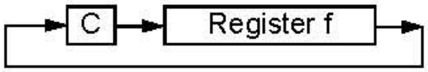
RLF	Rotate Left f through Carry
Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register ’f’ are rotated one bit to the left through the Carry Flag. If ’d’ is 0, the result is placed in the W register. If ’d’ is 1, the result is stored back in register ’f’.



SUBWF	Subtract W from f
Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) → (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2’s complement method) W register from 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'.

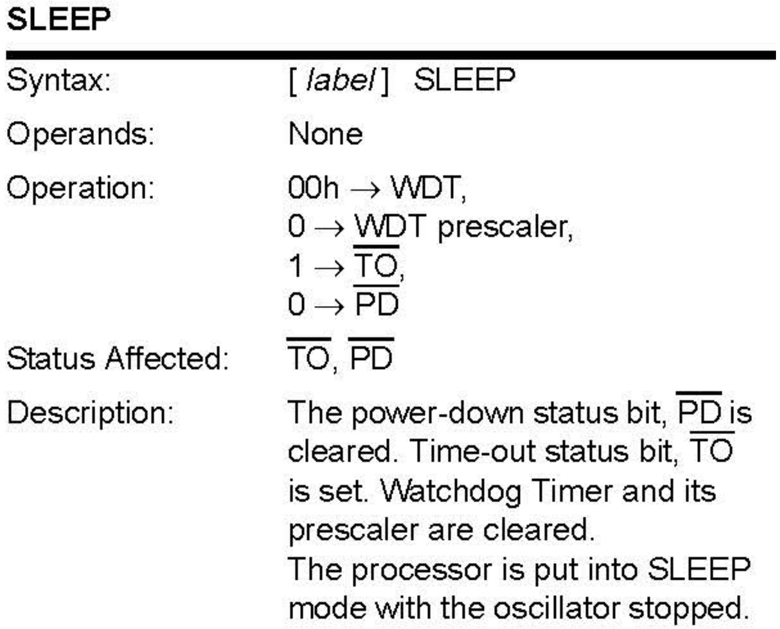
SUBLW	Subtract W from Literal
Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2’s complement method) from the eight-bit literal 'k'. The result is placed in the W register.

RRF	Rotate Right f through Carry
Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register ’f’ are rotated one bit to the right through the Carry Flag. If ’d’ is 0, the result is placed in the W register. If ’d’ is 1, the result is placed back in register ’f’.



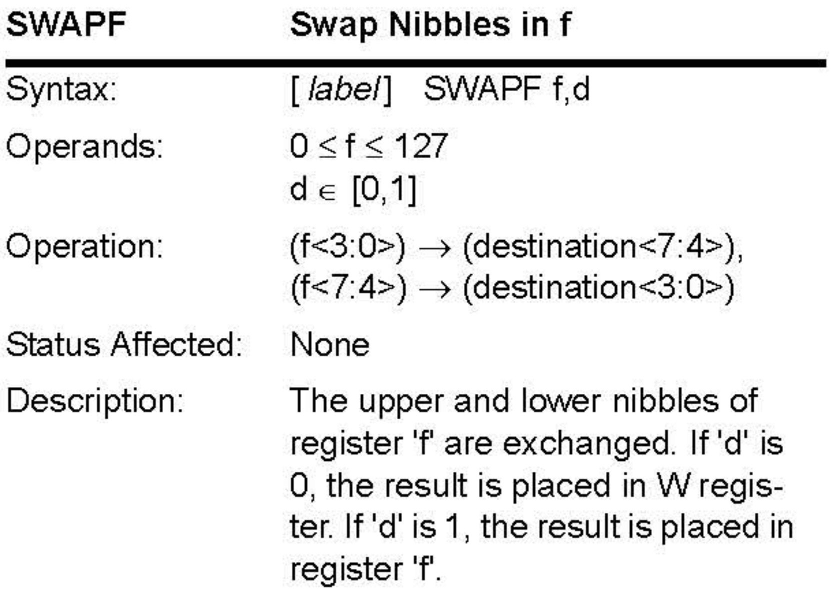
14-Bit Core Instruction Set

Examples



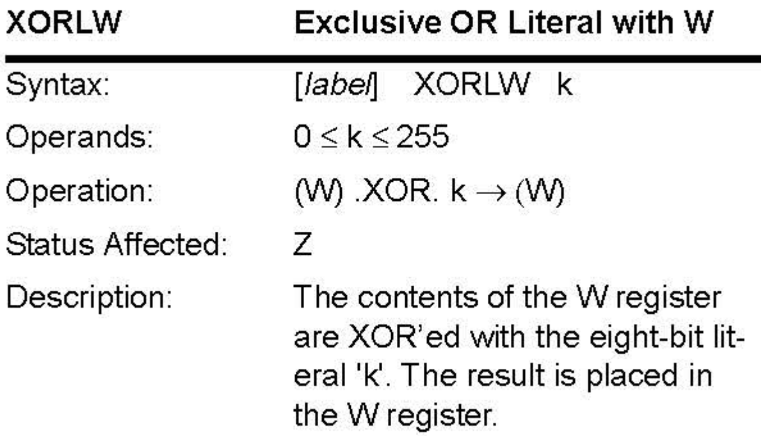
SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{\text{TO}}$, 0 → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Description:	The power-down status bit, $\overline{\text{PD}}$ is cleared. Time-out status bit, $\overline{\text{TO}}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.



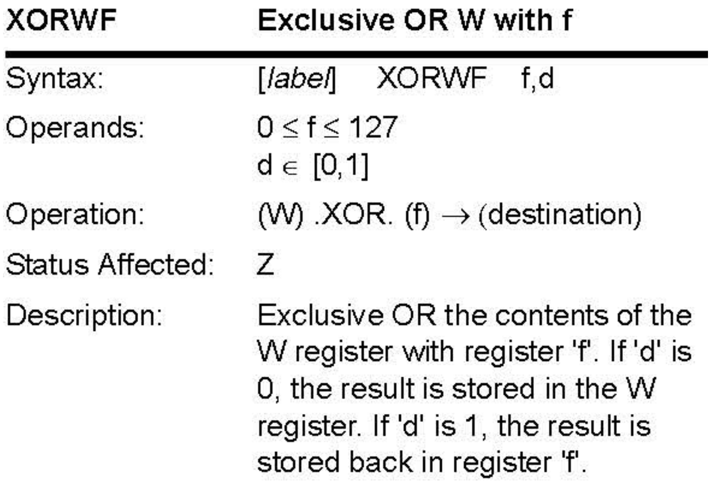
SWAPF Swap Nibbles in f

Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd' is 1, the result is placed in register 'f'.



XORLW Exclusive OR Literal with W

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.



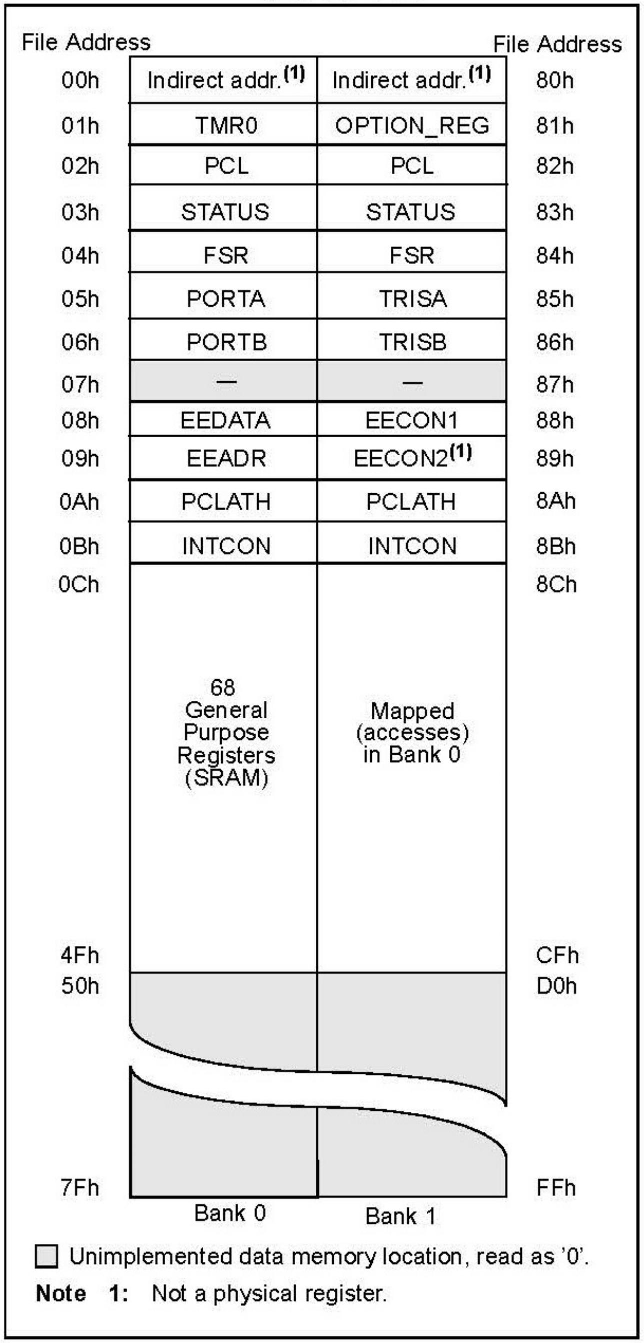
XORWF Exclusive OR W with f

Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of 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'.

Lecture_4

Special Function Registers
PIC SFRs

REGISTER FILE MAP - PIC16F84A

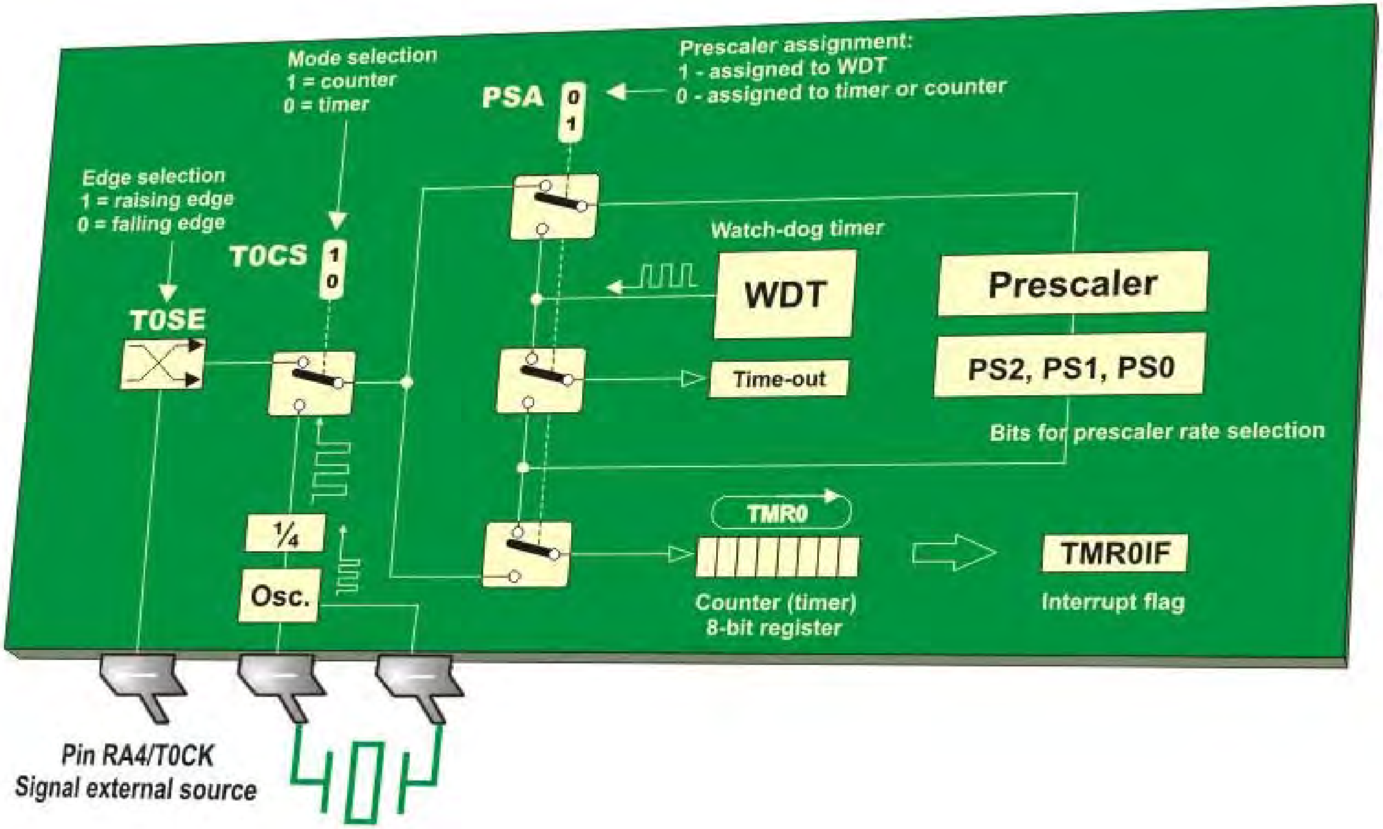


File Address	Bank 0	Bank 1	File Address
00h	Indirect addr.[1]	Indirect addr.[1]	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2[1]	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h	▨	▨	D0h
7Fh			FFh

▨ Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

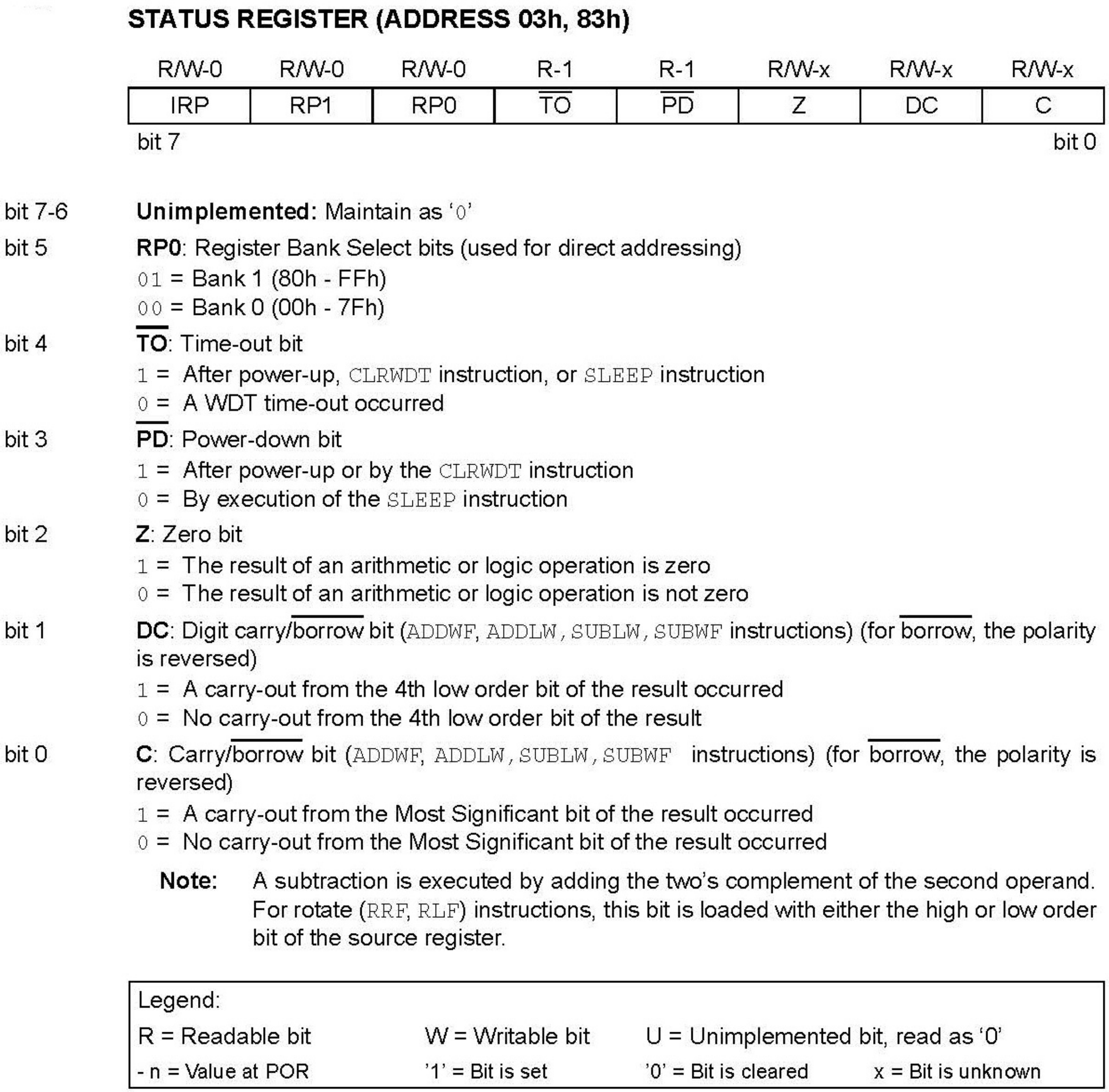
Special Function Registers
PIC SFRs



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

Special Function Registers PIC SFRs



STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C

bit 7 bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 **$\overline{\text{TO}}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{\text{PD}}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

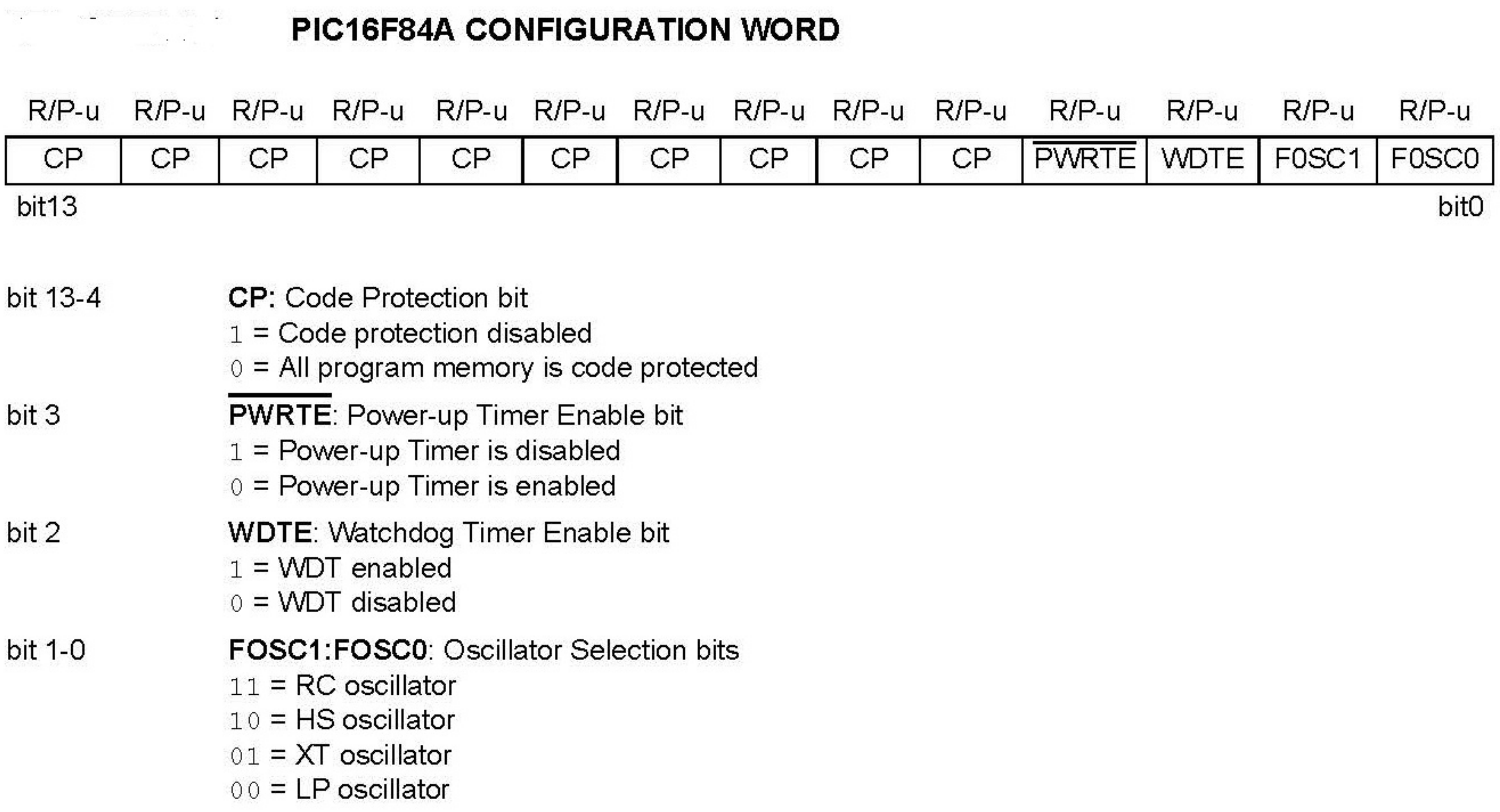
bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:
R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
- n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

Special Function Registers
PIC SFRs



PIC16F84A CONFIGURATION WORD

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit13													bit0

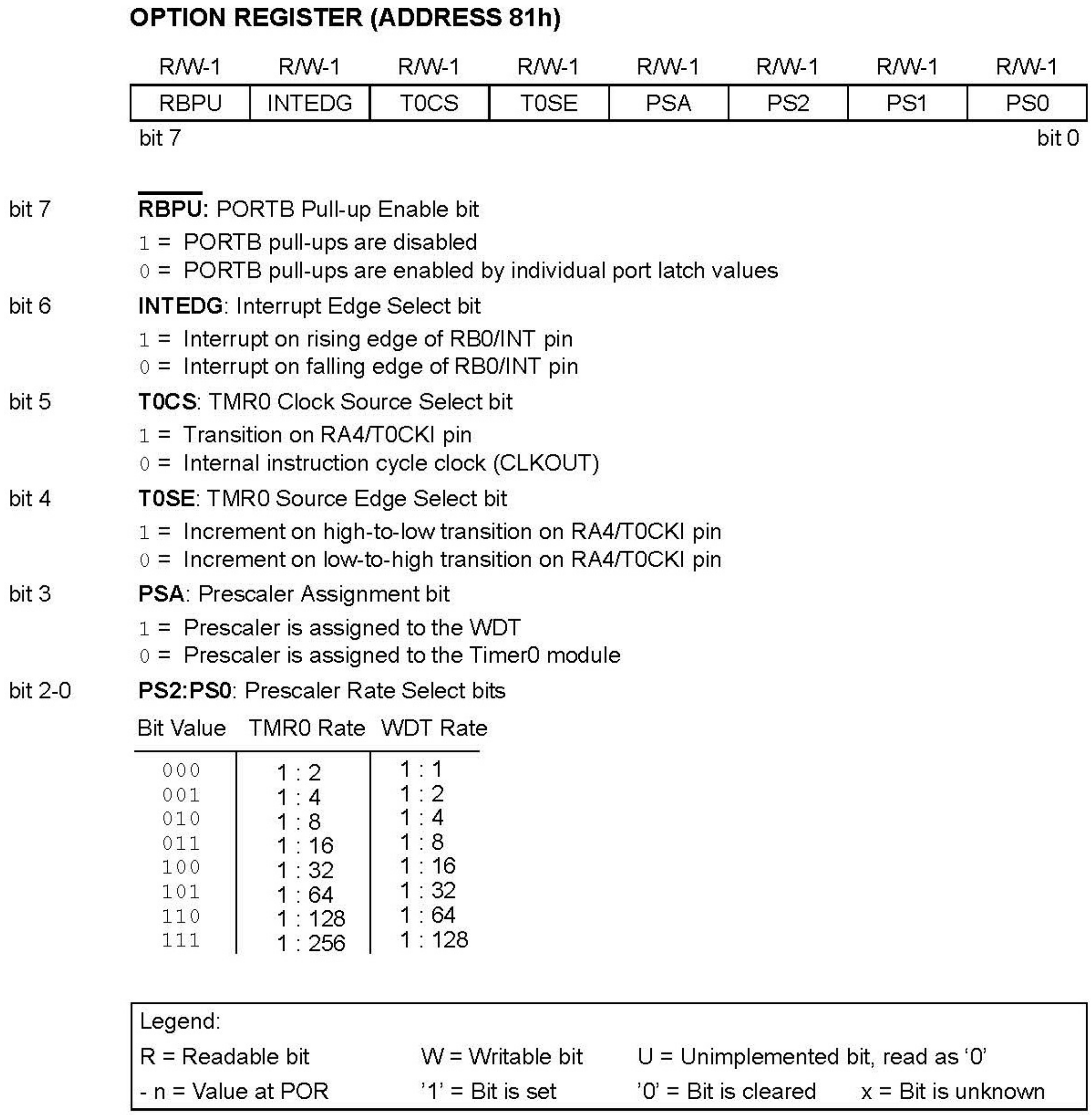
bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

bit 3 **$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Special Function Registers PIC SFRs



OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

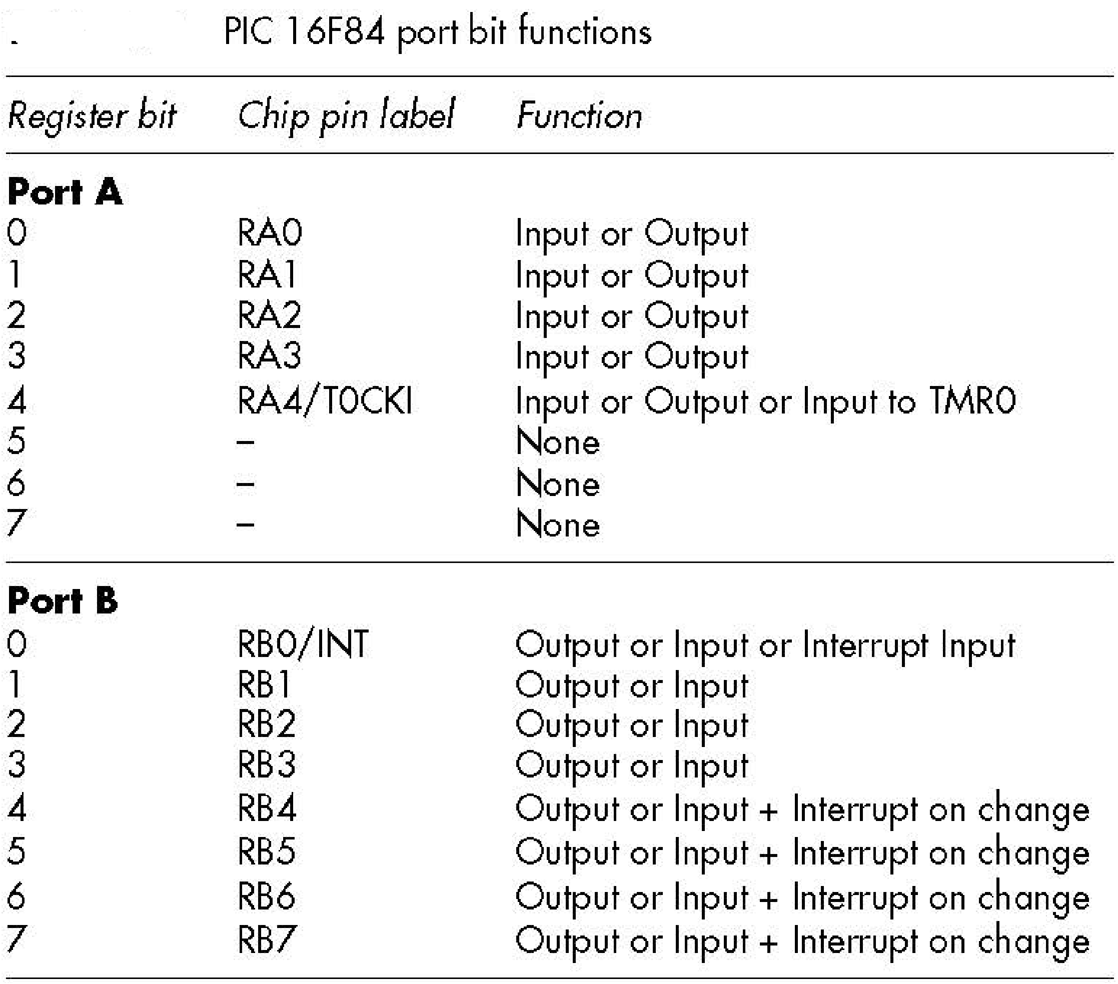
bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:
R = Readable bit W = Writable bit U = Unimplemented bit, read as ‘0’
- n = Value at POR ’1’ = Bit is set ’0’ = Bit is cleared x = Bit is unknown

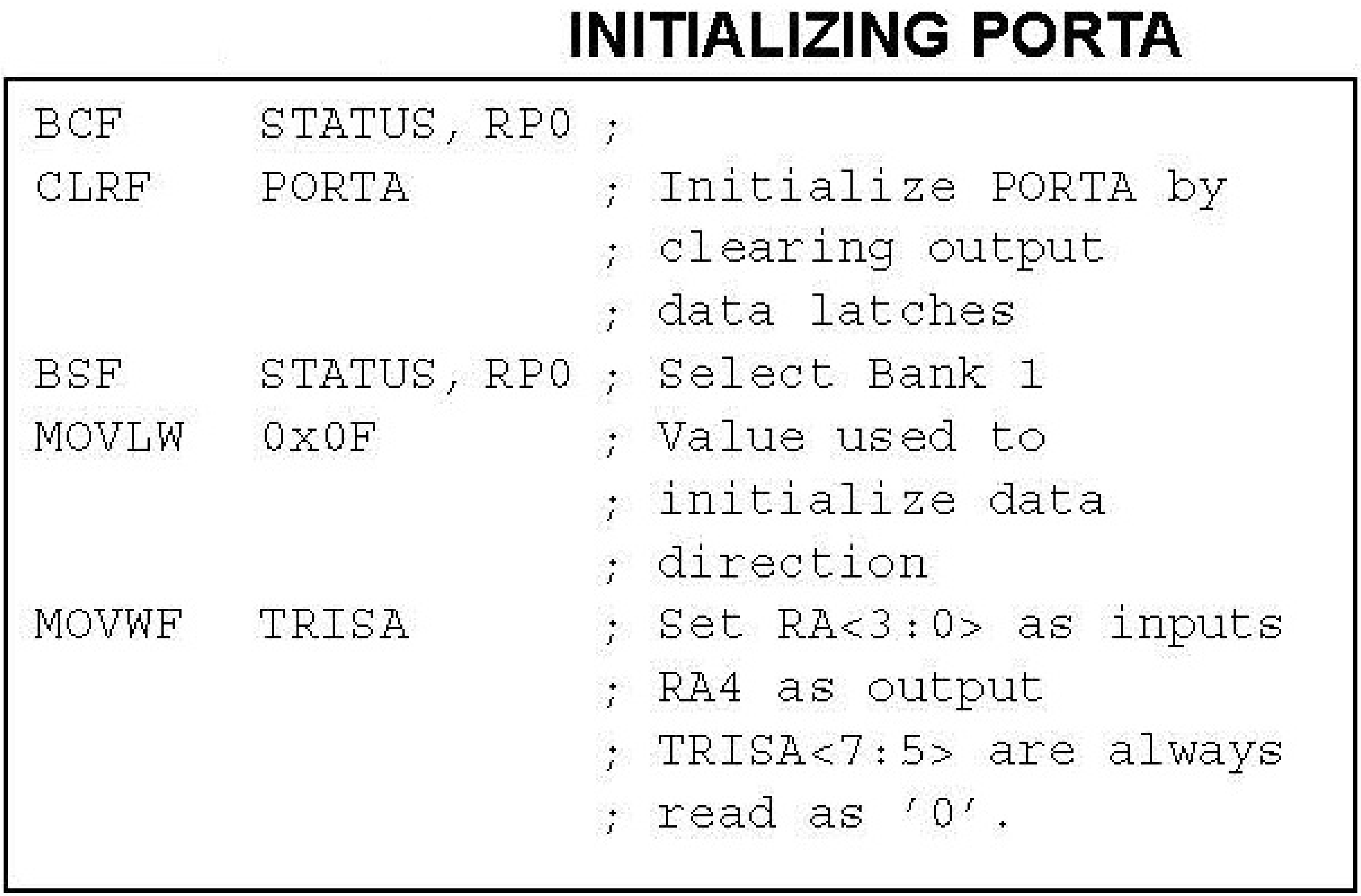
Special Function Registers PIC SFRs



PIC 16F84 port bit functions

Register bit	*Chip pin label*	*Function*
Port A		
0	RA0	Input or Output
1	RA1	Input or Output
2	RA2	Input or Output
3	RA3	Input or Output
4	RA4/T0CKI	Input or Output or Input to TMR0
5	–	None
6	–	None
7	–	None
Port B		
0	RB0/INT	Output or Input or Interrupt Input
1	RB1	Output or Input
2	RB2	Output or Input
3	RB3	Output or Input
4	RB4	Output or Input + Interrupt on change
5	RB5	Output or Input + Interrupt on change
6	RB6	Output or Input + Interrupt on change
7	RB7	Output or Input + Interrupt on change

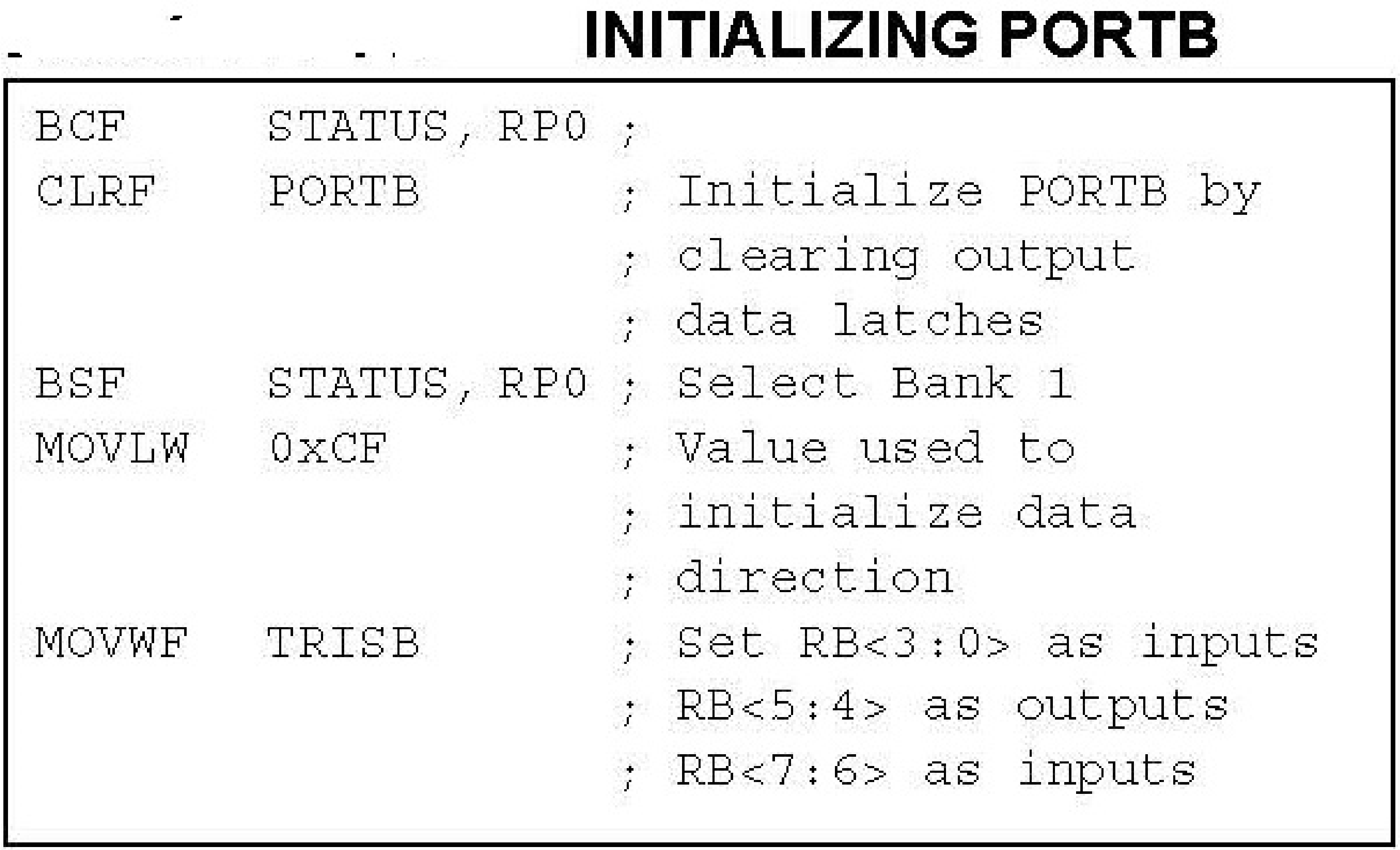
Special Function Registers
PIC SFRs



INITIALIZING PORTA

```
BCF    STATUS, RP0 ;
CLRF   PORTA       ; Initialize PORTA by
                   ; clearing output
                   ; data latches
BSF    STATUS, RP0 ; Select Bank 1
MOVLW  0x0F        ; Value used to
                   ; initialize data
                   ; direction
MOVWF  TRISA       ; Set RA<3:0> as inputs
                   ; RA4 as output
                   ; TRISA<7:5> are always
                   ; read as ‘0’.
```

Special Function Registers
PIC SFRs



INITIALIZING PORTB

```
BCF     STATUS, RP0 ;
CLRF    PORTB       ; Initialize PORTB by
                    ; clearing output
                    ; data latches
BSF     STATUS, RP0 ; Select Bank 1
MOVLW   0xCF        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISB       ; Set RB<3:0> as inputs
                    ; RB<5:4> as outputs
                    ; RB<7:6> as inputs
```

Lecture_5

Oscillators

SPECIAL FEATURES OF THE CPU

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection

SPECIAL FEATURES OF THE CPU

OSC Selection

	FOSC1 FOSC0
LP Low Power Crystal	00
XT Crystal/Resonator	01
HS High Speed	10
RC Resistor/Capacitor	11

SPECIAL FEATURES OF THE CPU

• LP	32.768kHz - 200kHz
• XT	1MHz - 4MHz
• HS	4MHz - 10 MHz
• RC	DC - 4MHz

SPECIAL FEATURES OF THE CPU

Quartz Crystals
LP, XT, and HS

Ceramic Resonator
XT and HS

SPECIAL FEATURES OF THE CPU

REST

The PIC16F84A differentiates among various kinds of RESET:

- Power-on Reset (POR)
- MCLR during normal operation
- MCLR during SLEEP
- WDT Reset (during normal operation)
- WDT Wake-up (during SLEEP)

SPECIAL FEATURES OF THE CPU

REST

A Power-on Reset pulse is generated on-chip when VDD rise is detected.

The Power-up Timer (PWRT) provides a fixed 72 ms nominal time-out.

The chip is kept in RESET as long as the PWRT is active.
A configuration bit, PWRTE , can enable/disable the PWRT.

SPECIAL FEATURES OF THE CPU

REST

A device may be powered down (SLEEP) and later powered up (wake-up from SLEEP).

The Power-down mode is entered by executing the SLEEP instruction.

the PD bit (STATUS – bit3) is cleared

the TO bit (STATUS – bit4) is set

the oscillator driver is turned off.

The I/O ports maintain the status they had before the SLEEP instruction was executed

SPECIAL FEATURES OF THE CPU

REST

The device can wake-up from SLEEP through one of the following events:

1. External RESET input on $\overline{\text{MCLR}}$ pin.
2. WDT wake-up (if WDT was enabled).
3. Interrupt from RB0/INT pin, RB port change, or data EEPROM write complete.

The first event ($\overline{\text{MCLR}}$ Reset) will cause a device RESET.
The two latter events are considered a continuation of program execution.

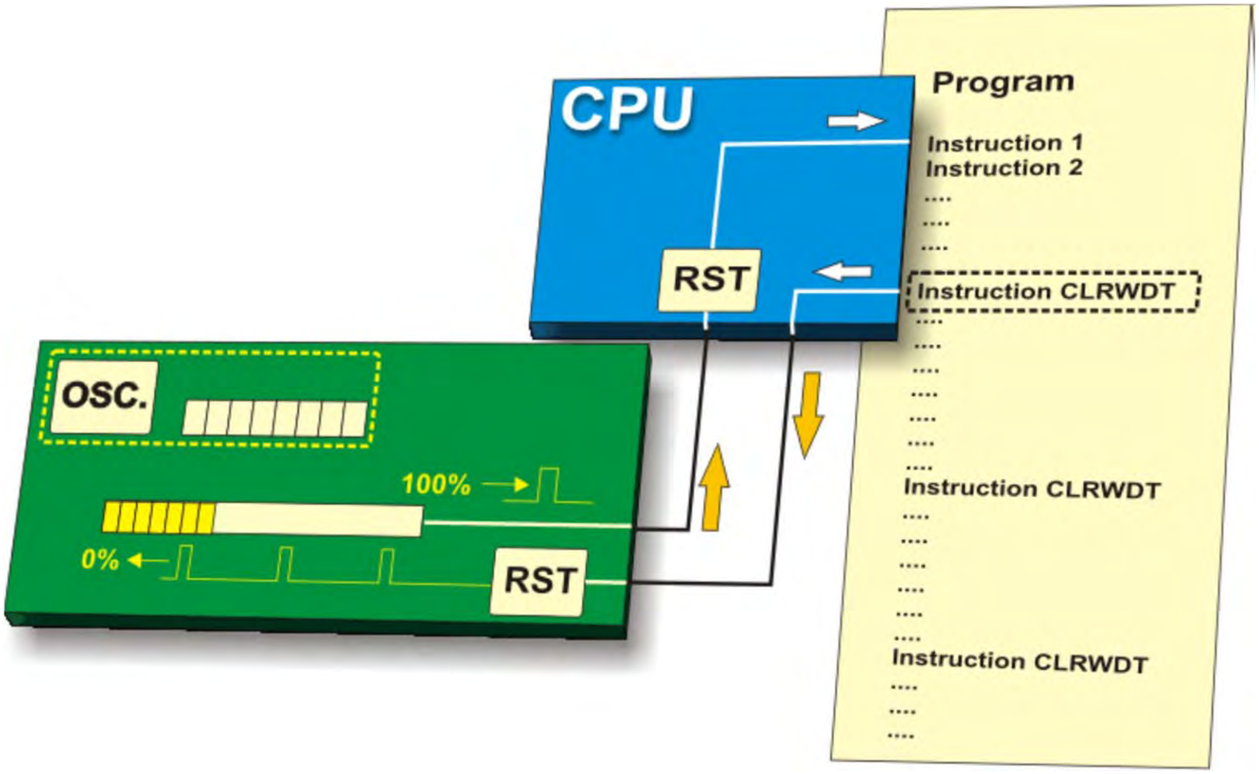
The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits can be used to determine the cause of a device RESET.

The $\overline{\text{PD}}$ bit, which is set on power-up, is cleared when SLEEP is invoked.

The $\overline{\text{TO}}$ bit is cleared if a WDT time-out occurred (and caused wake-up).

SPECIAL FEATURES OF THE CPU

REST



SPECIAL FEATURES OF THE CPU

Interrupts

The PIC16F84A has 4 sources of interrupt:

- External interrupt RB0/INT pin
- TMR0 overflow interrupt
- PORTB change interrupts (pins RB7:RB4)
- Data EEPROM write complete interrupt

SPECIAL FEATURES OF THE CPU

Interrupts

External interrupt on RB0/INT pin is edge triggered

rising if INTEDG bit (OPTION_REG) is set

falling if INTEDG bit is clear

When a valid edge appears on the RB0/INT pin, the INTF bit (INTCON - bit1) is set.

This interrupt can be disabled by clearing control bit INTE (INTCON – bit4).

SPECIAL FEATURES OF THE CPU

Interrupts

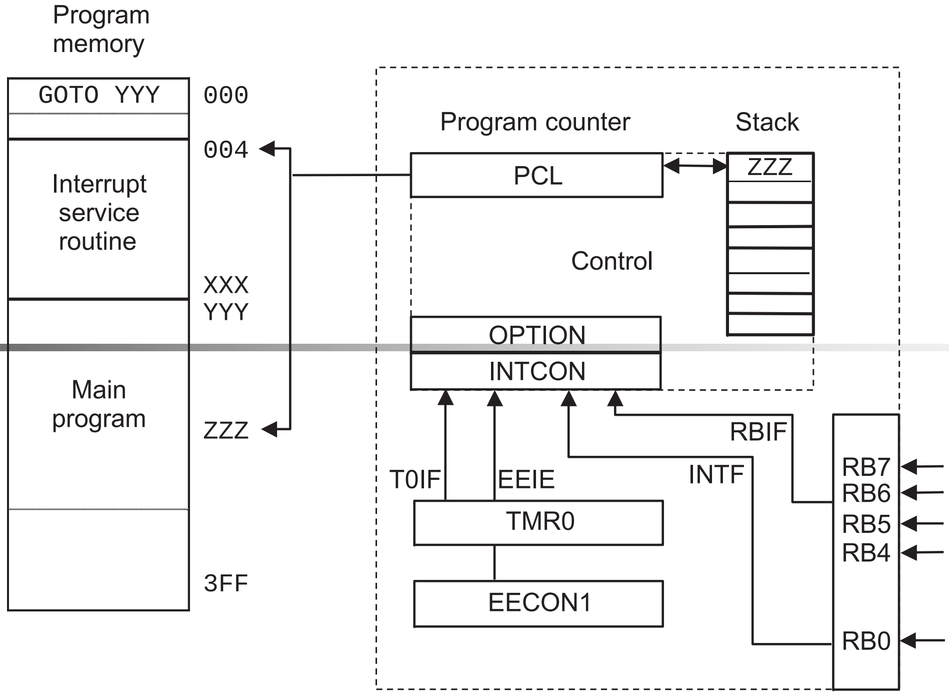
An overflow (FFh → 00h) in TMR0 will set flag bit T0IF (INTCON – bit2).
The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON – bit5)

An input change on PORTB bit 7:4 sets flag bit RBIF (INTCON – bit0)

The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON - bit3)

At the completion of a data EEPROM write cycle, flag bit EEIF (EECON1 - bit4) will be set.

The interrupt can be enabled/disabled by setting/clearing enable bit EEIE (INTCON - bit6)



Interrupt control bit functions

	Bit	Label	Function	Settings
INTCON	0	RBIF	Port B (4:7) Interrupt flag	0 = No change 1 = Bit change detected
	1	INTF	RB0 Interrupt flag	0 = No interrupt 1 = Interrupt detected
	2	T0IF	TMR0 overflow Interrupt flag	0 = No overflow 1 = Overflow detected
	3	RBIE	Port B (4:7) Interrupt enable	0 = Disabled 1 = Enabled
	4	INTE	RB0 Interrupt enable	0 = Disabled 1 = Enabled
	5	T0IE	TMR0 overflow Interrupt enable	0 = Disabled 1 = Enabled
	6	EEIE	EEPROM write complete interrupt enable flag	0 = Disabled 1 = Enabled
	7	GIE	Global interrupt enable	0 = Disabled 1 = Enabled

OPTION	6	INTEDG	RB0 interrupt Active edge select	0 = Falling edge 1 = Rising edge