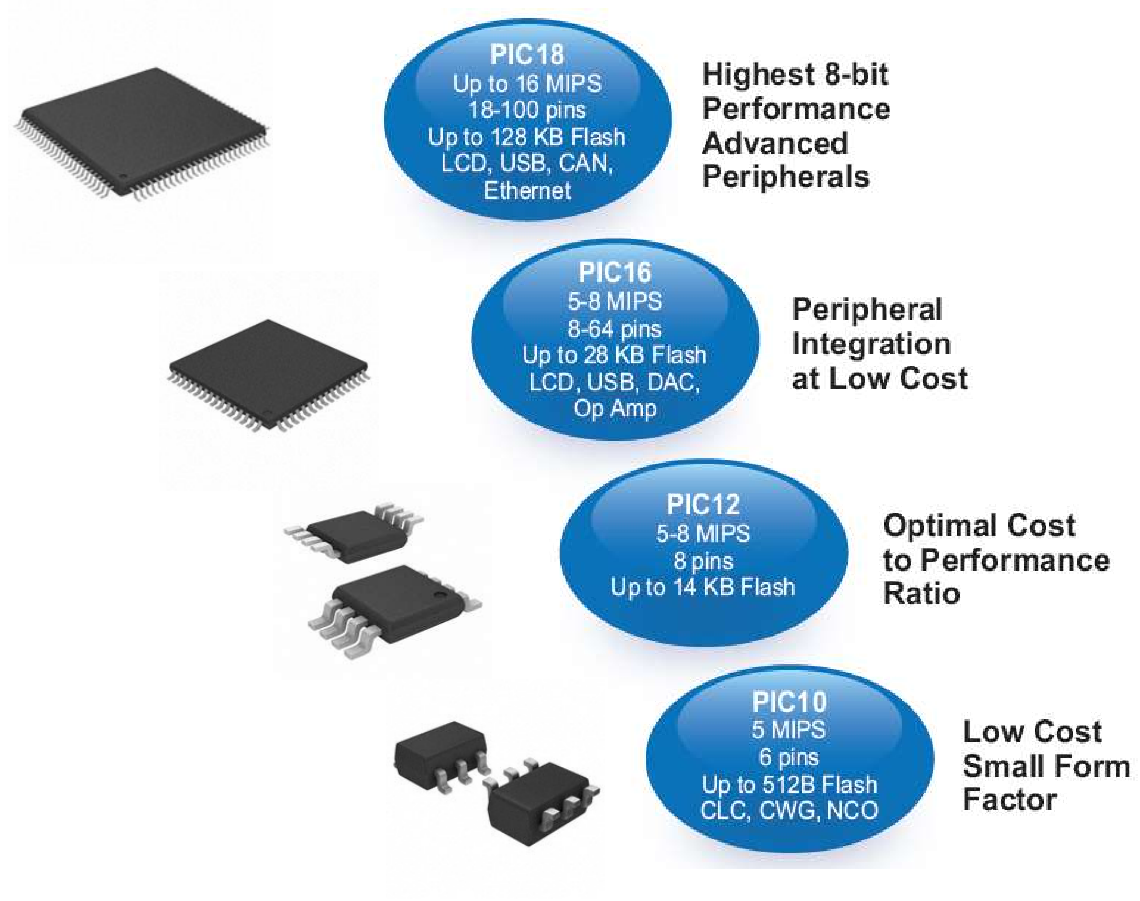
TECNOLÓGICO DE MONTERREY ®

Subject 02
THE PIC 18 ARCHITECTURE

Microchip 8 bits microcontrollers

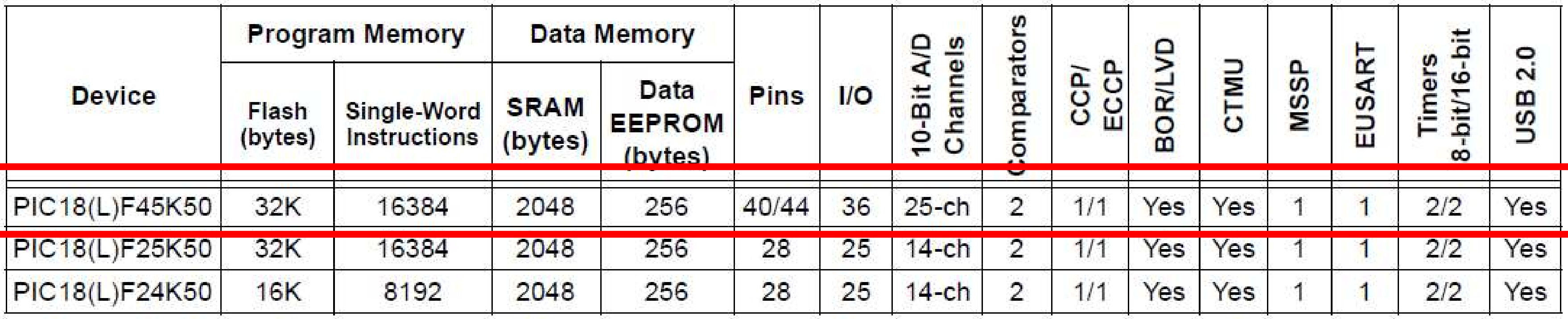


http://ww1.microchip.com/downloads/en/DeviceDoc/30010068F.pdf

General specs for the PIC18

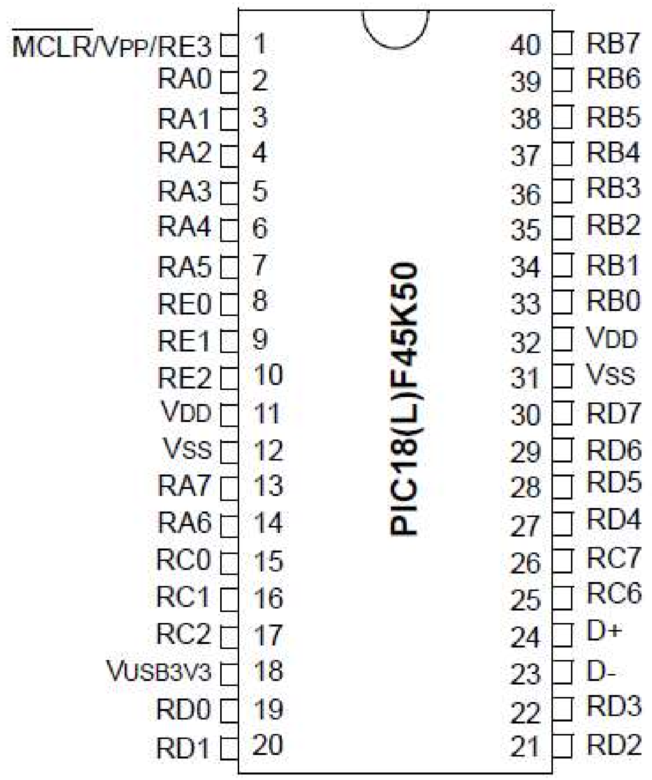
- Parallel ports (I/O)
- Timers
- PWM
- SPI e I2C
- USART
- A/D Converter
- Analog comparator
- Low power modes
- SRAM/EEPROM
- Flash/EPROM
- CAN/USB/Ethernet
- LCD
- More than 1000 part numbers available

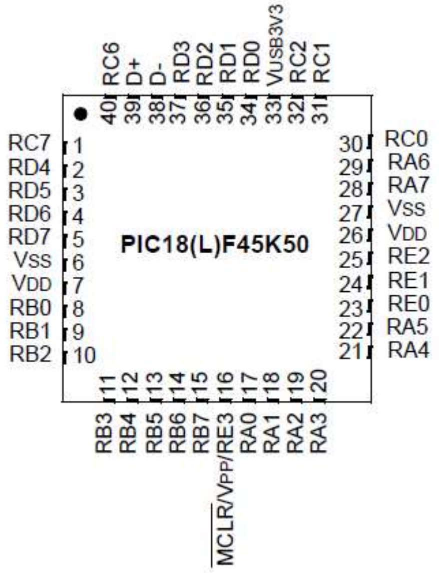
PIC18(L)F2X/45K50

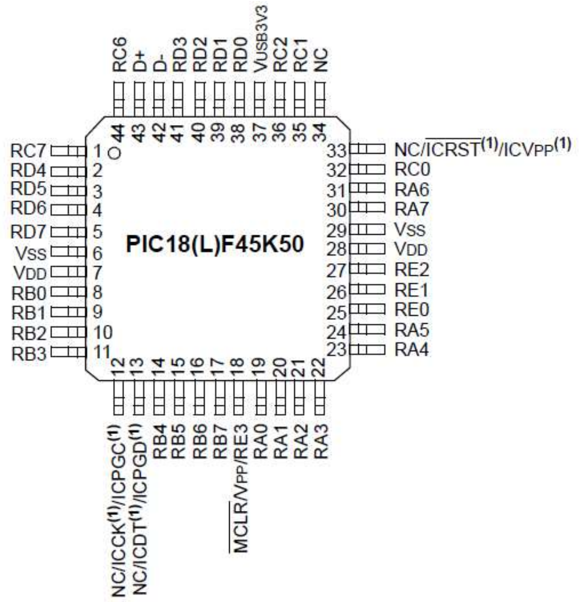


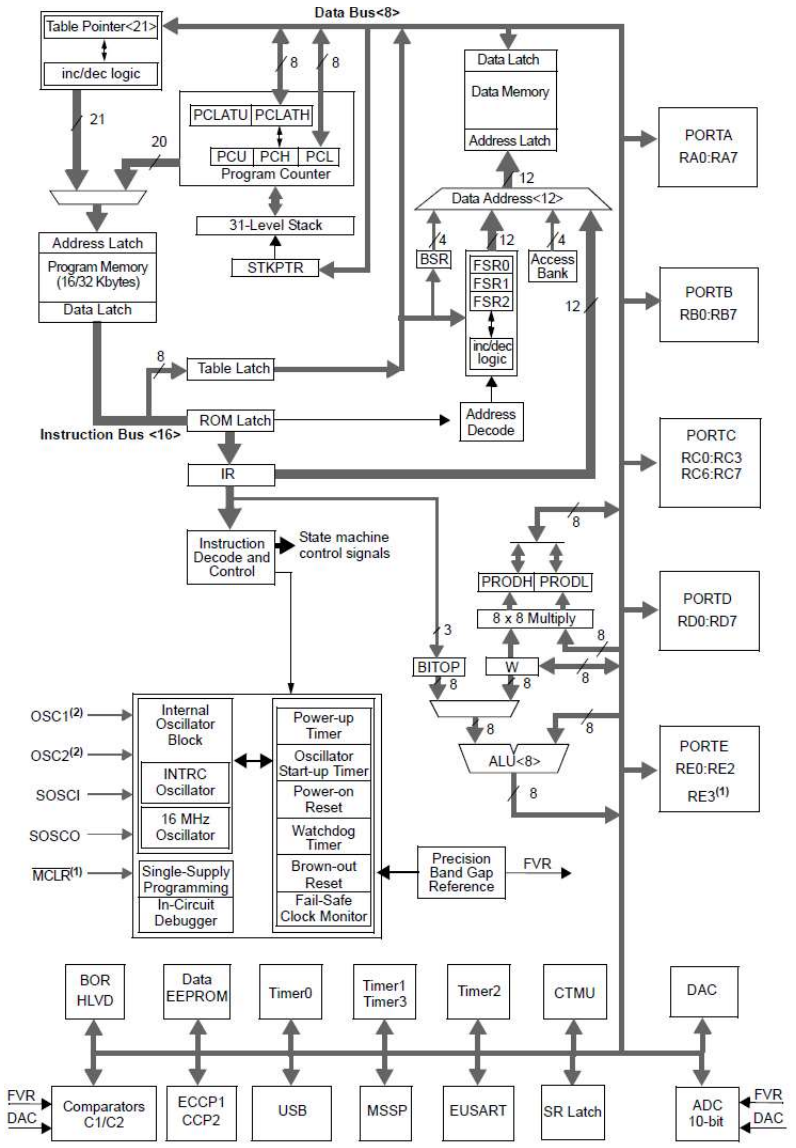
Device	Program Memory		Data Memory		Pins	I/O	10-Bit A/D Channels	Comparators	CCP/ ECCP	BOR/LVD	CTMU	MSSP	EUSART	Timers 8-bit/16-bit	USB 2.0
	Flash (bytes)	Single-Word Instructions	SRAM (bytes)	Data EEPROM (bytes)											
PIC18(L)F45K50	32K	16384	2048	256	40/44	36	25-ch	2	1/1	Yes	Yes	1	1	2/2	Yes
PIC18(L)F25K50	32K	16384	2048	256	28	25	14-ch	2	1/1	Yes	Yes	1	1	2/2	Yes
PIC18(L)F24K50	16K	8192	2048	256	28	25	14-ch	2	1/1	Yes	Yes	1	1	2/2	Yes

PIC18(L)F45K50



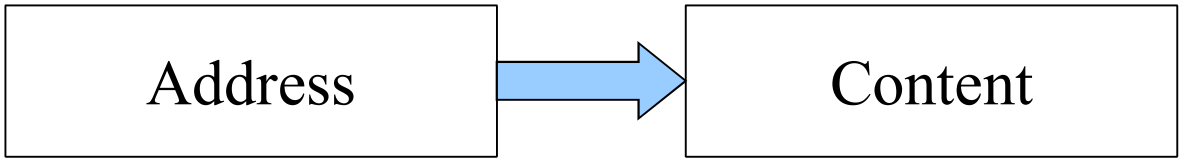






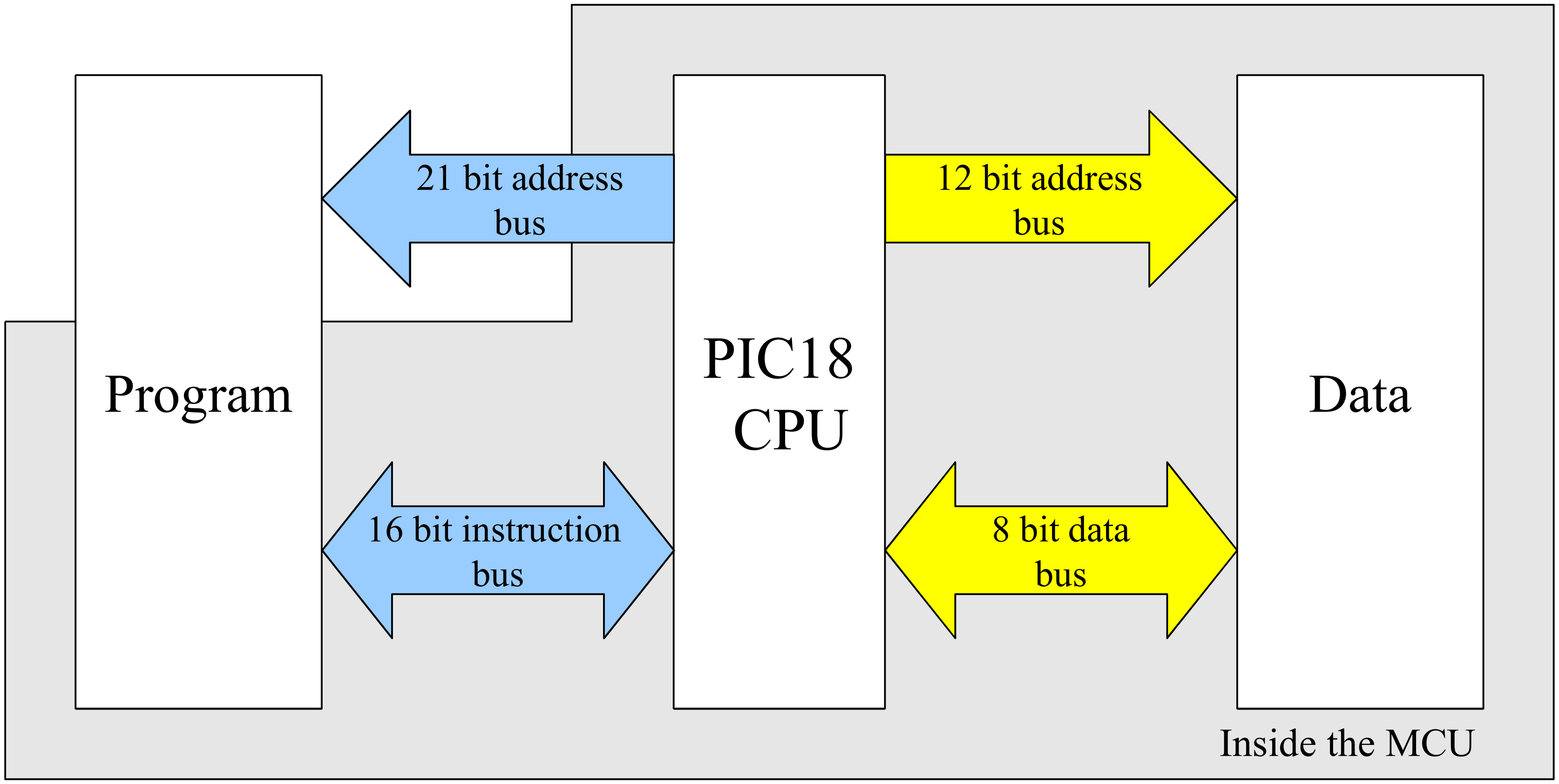
PIC18(L)F2X/45K50 FAMILY BLOCK DIAGRAM

Memory Organization



Address	→ Memory address to access
[Address]	→ Content of the memory address

Memory spaces



Memory spaces

- Program memory maximum addressing

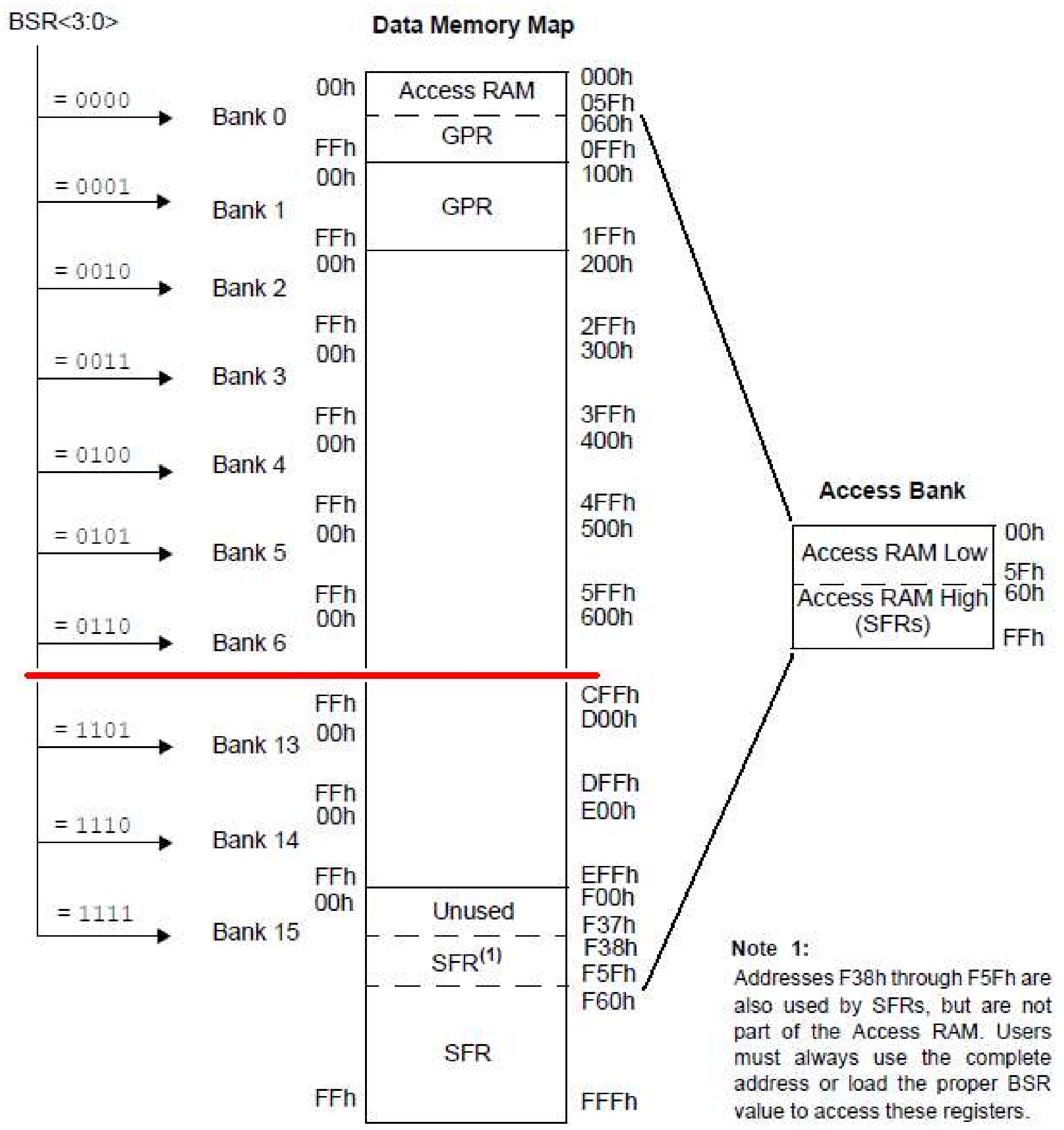
  2^{21}= 2097152 = 2Mbytes

- Data memory maximum addressing

  2^{12}= 4096 =4Kbytes

The data memory space terminology

- Microchip calls the also register file to the data memory.
- A register is an element of the file (byte) but can also be called a file register



Note 1: Addresses F38h through F5Fh are also used by SFRs, but are not part of the Access RAM. Users must always use the complete address or load the proper BSR value to access these registers.

Instruction basic construction

$$W = A + B$$

Operation Code | Operand 1 | Operand n

Hypothetic instruction

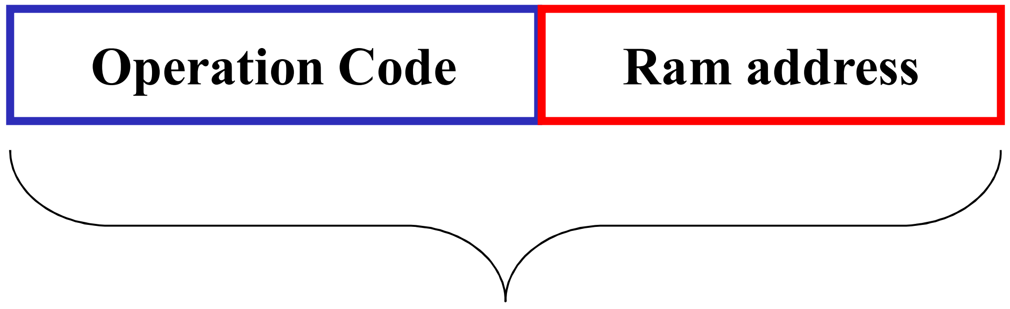
Y = [ram location] + Y

Operation Code	**Operand 1**

Y = An internal working 8 bit register
[ram location] = is the content RAM address specified by the user

In the PIC architecture

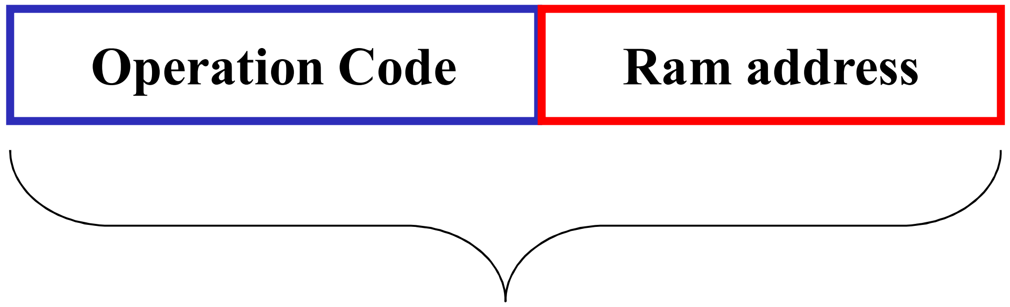
- The optimal instruction length In bits is 16
- To Access the entire ram you need a 12 bit address
- Given the later, the hypothetical instruction would require:



16 (instruction) + 12 (address of the operand) = 28 bits

Instruction modification ?

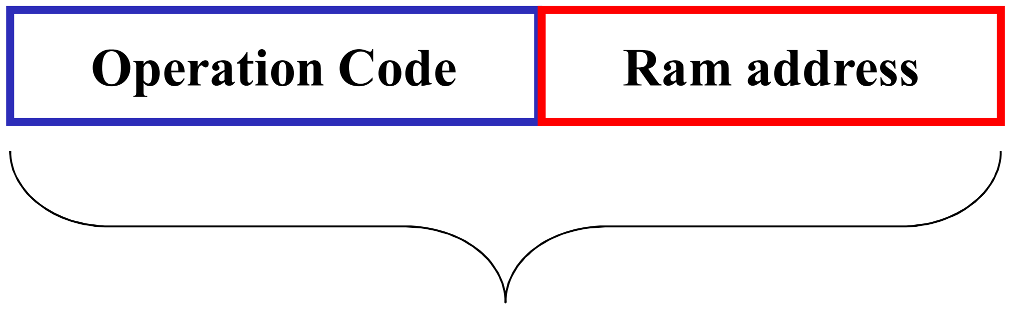
- If we want to be optimal and use only 16 bits for the instruction what could we do ?
- Reduce operation code length to 4 bits and express the complete address



4 (instruction) + 12 (address of the operand) = 16 bits

Instruction modification ?

- If we reduce the section that tell the instruction what to do to only 4 bits
- With 4 bits how many “different” things we could do with an instruction of that nature

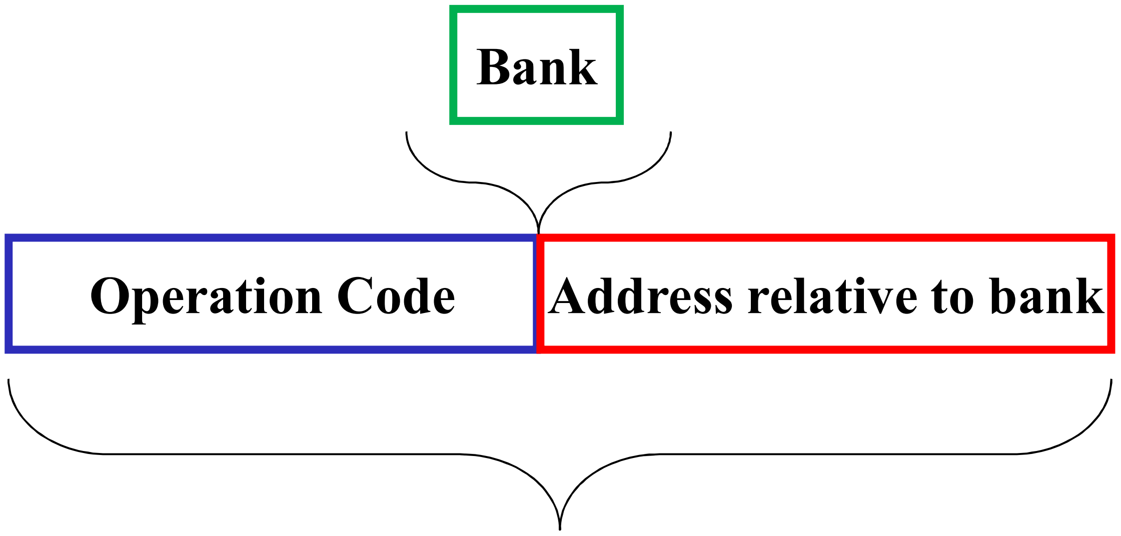


4 (instruction) + 12 (address of the operand) = 16 bits

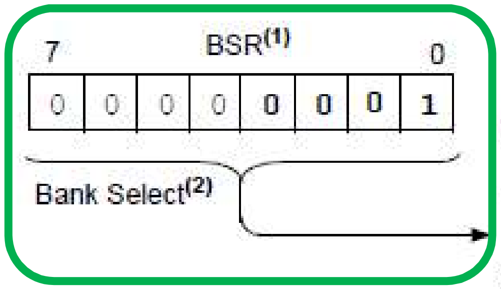
What was the solution?

- Instead of using a direct address in the instruction microchip used and “offset” type of addressing.
- The instructions work relative to the offset (Bank) that is defined by another instruction that is executed previously
- The instructions sets the working bank sets the 4 most significant bits of physical address of the ram, this way:

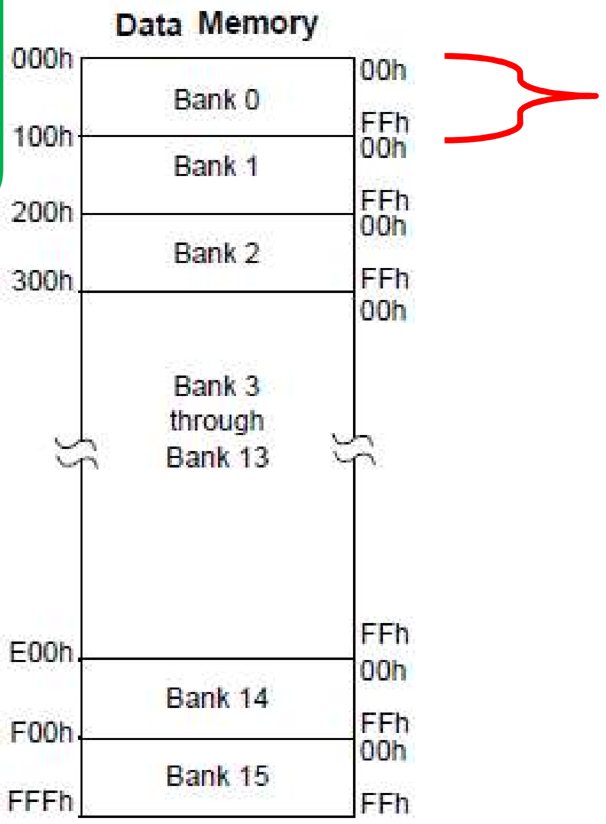
What was the solution?



8 (instruction) + 8(address of the operand) = 16 bits



Bank (offset)



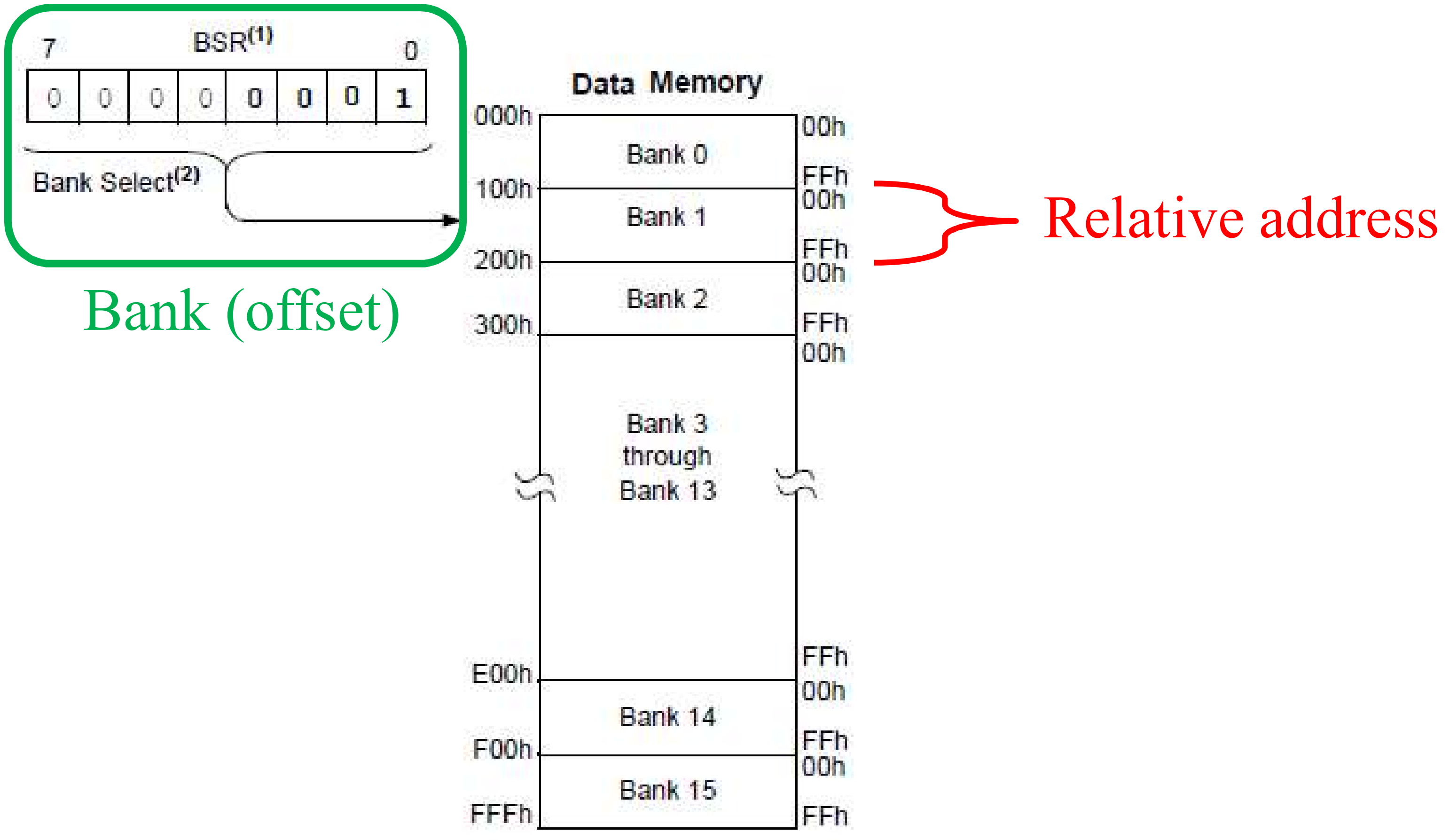
Bank 0

Bank 1

Bank 2

******

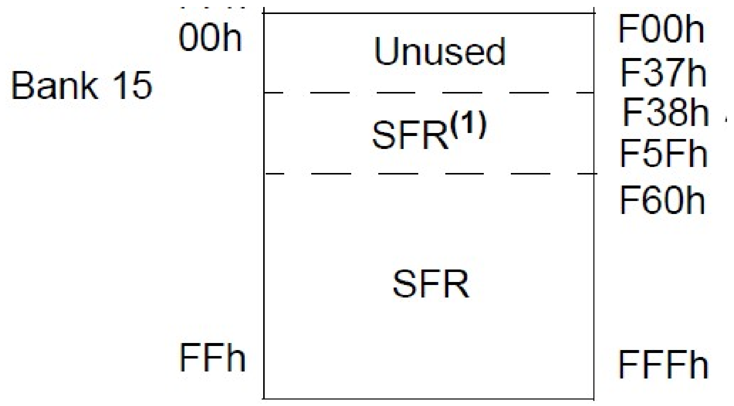
Bank 15



Bank	Address relative to bank
0→F	00→FF

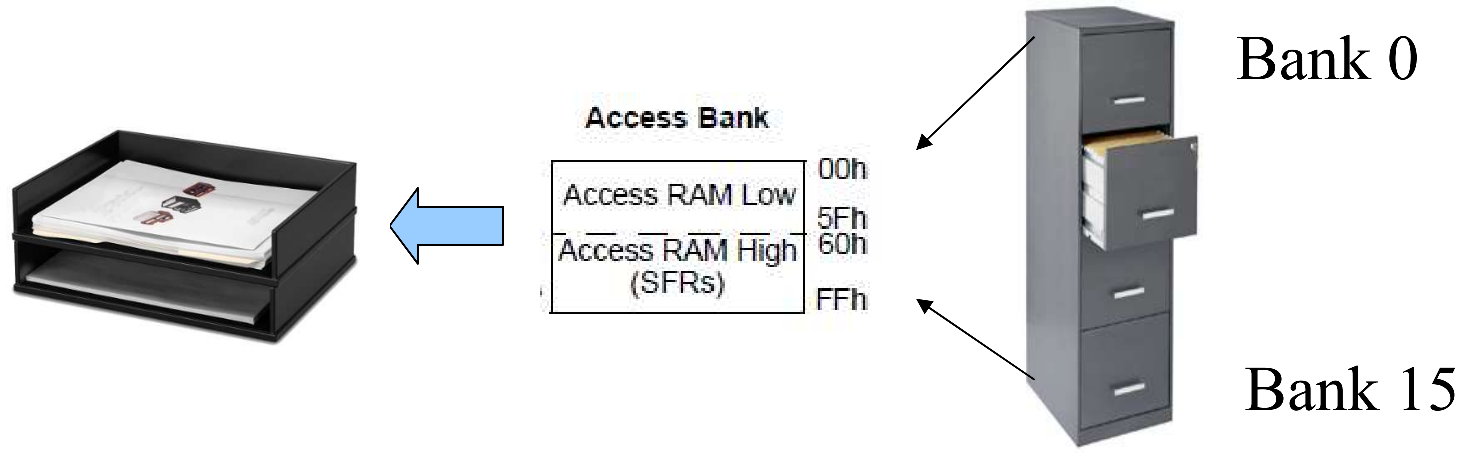
The SFR

Special Function Registers: Are registers employed to interact with the CPU and the peripherals, they are mapped in the data memory space



The Acces Bank

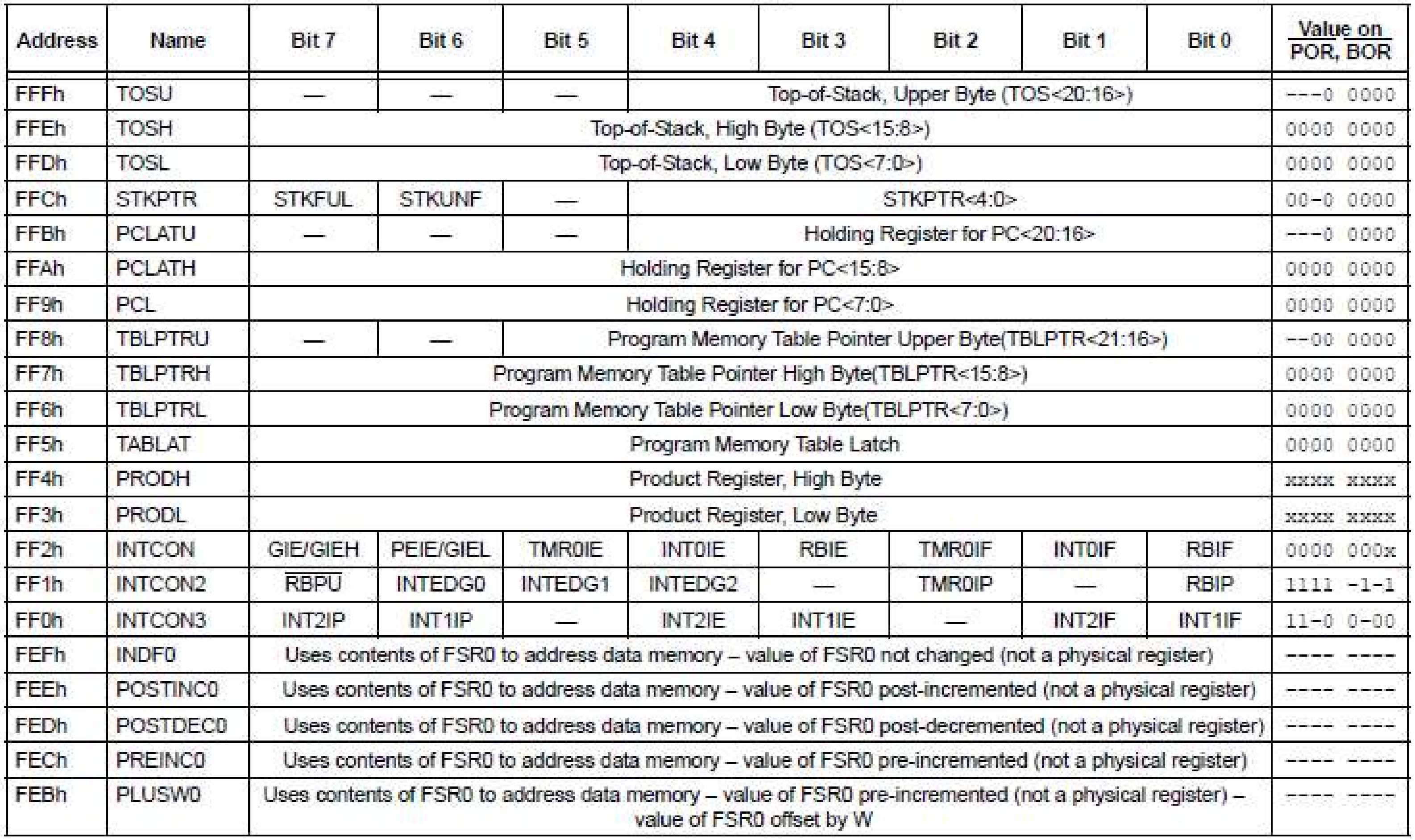
It is as if from drawer 0 and drawer 15 (Bank 0 and Bank 15), we separated in a pair of trays a group of folders (bytes) of each to work and not have to be opening the drawers. Each folder would contain 8 sheets (bits).



EEPROM

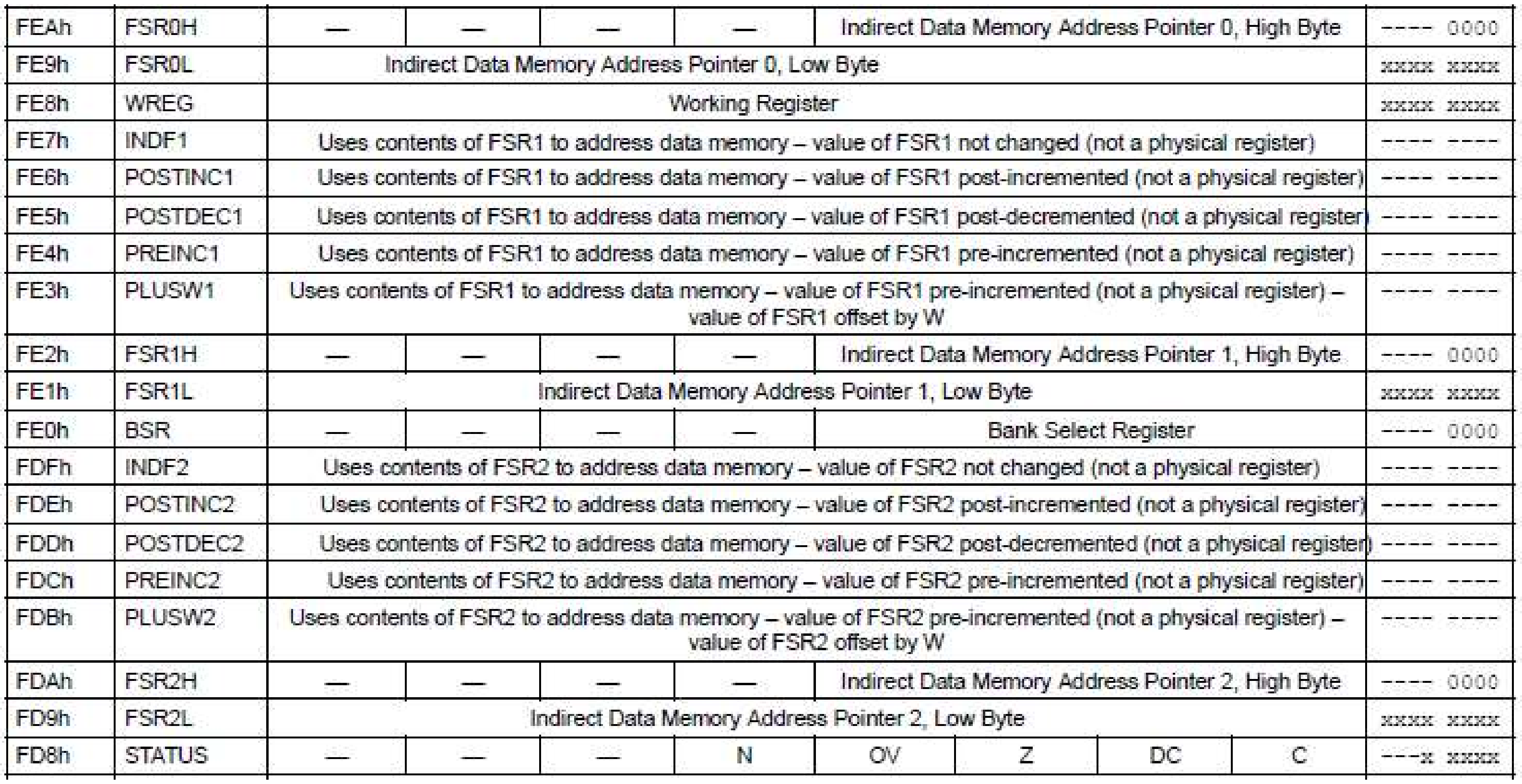
- The PIC18 contains up to 256 bytes of read-write non volatile memory
- This memory is not mapped to the data memory space
- To Access a location on the EEPROM and writing to it you must use the SFR's (indirect access)

The CPU registers



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FFFh	TOSU	—	—	—	Top-of-Stack, Upper Byte (TOS<20:16>)					---0 0000
FFEh	TOSH	Top-of-Stack, High Byte (TOS<15:8>)								0000 0000
FFDh	TOSL	Top-of-Stack, Low Byte (TOS<7:0>)								0000 0000
FFCh	STKPTR	STKFUL	STKUNF	—	STKPTR<4:0>					00-0 0000
FFBh	PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000
FFAh	PCLATH	Holding Register for PC<15:8>								0000 0000
FF9h	PCL	Holding Register for PC<7:0>								0000 0000
FF8h	TBLPTRU	—	—	Program Memory Table Pointer Upper Byte(TBLPTR<21:16>)						--00 0000
FF7h	TBLPTRH	Program Memory Table Pointer High Byte(TBLPTR<15:8>)								0000 0000
FF6h	TBLPTRL	Program Memory Table Pointer Low Byte(TBLPTR<7:0>)								0000 0000
FF5h	TABLAT	Program Memory Table Latch								0000 0000
FF4h	PRODH	Product Register, High Byte								xxxx xxxx
FF3h	PRODL	Product Register, Low Byte								xxxx xxxx
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x
FF1h	INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1
FF0h	INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00
FEFh	INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								---- ----
FEEh	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								---- ----
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								---- ----
FECh	PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								---- ----
FEBh	PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								---- ----

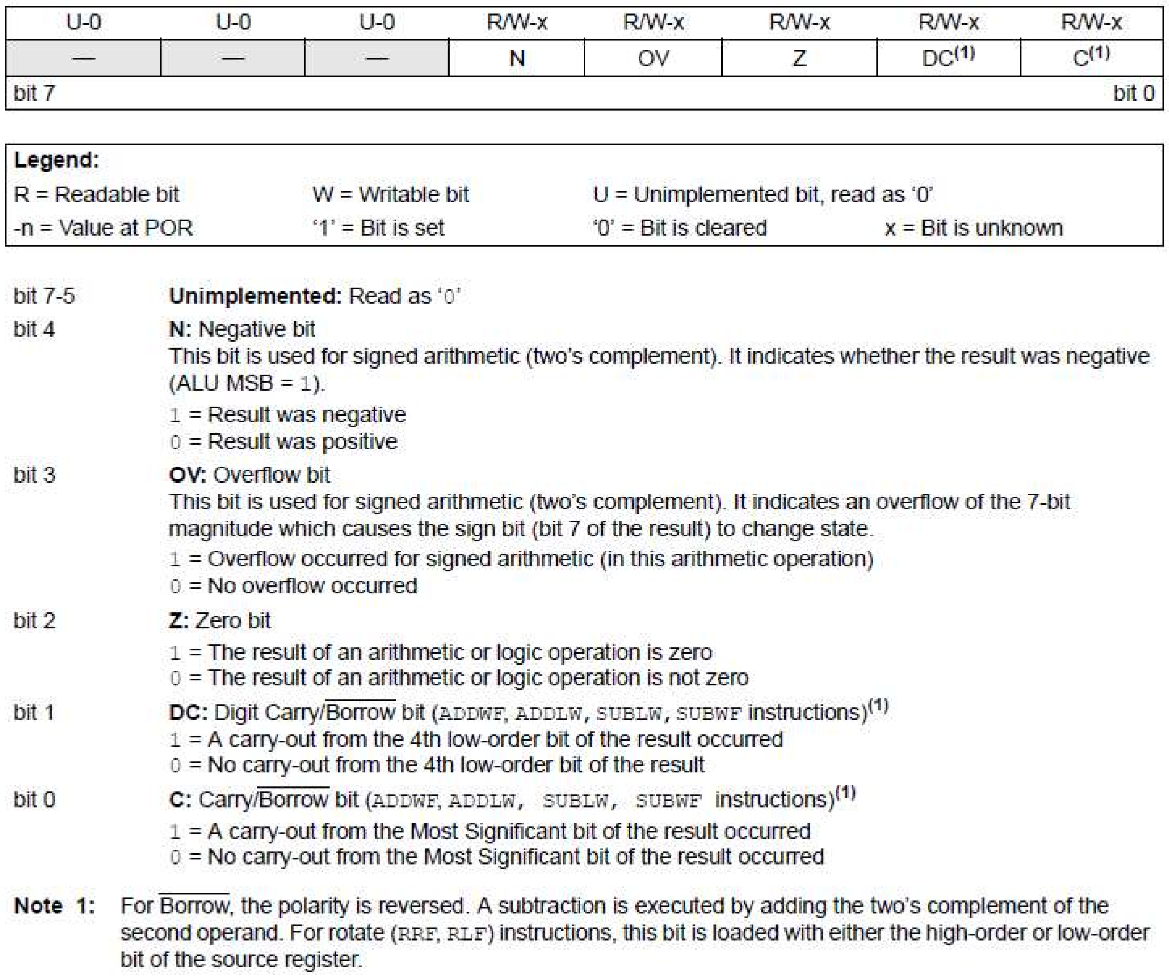
The CPU registers cont…



FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0, High Byte				---- 0000
FE9h	FSR0L	Indirect Data Memory Address Pointer 0, Low Byte								xxxx xxxx
FE8h	WREG	Working Register								xxxx xxxx
FE7h	INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								---- ----
FE6h	POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								---- ----
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								---- ----
FE4h	PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								---- ----
FE3h	PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								---- ----
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1, High Byte				---- 0000
FE1h	FSR1L	Indirect Data Memory Address Pointer 1, Low Byte								xxxx xxxx
FE0h	BSR	—	—	—	—	Bank Select Register				---- 0000
FDFh	INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								---- ----
FDEh	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								---- ----
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								---- ----
FDCh	PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								---- ----
FDBh	PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								---- ----
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2, High Byte				---- 0000
FD9h	FSR2L	Indirect Data Memory Address Pointer 2, Low Byte								xxxx xxxx
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx

The STATUS registers

Contains the status of the last ALU operation



U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC[(1)]	C[(1)]
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (two's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (two's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

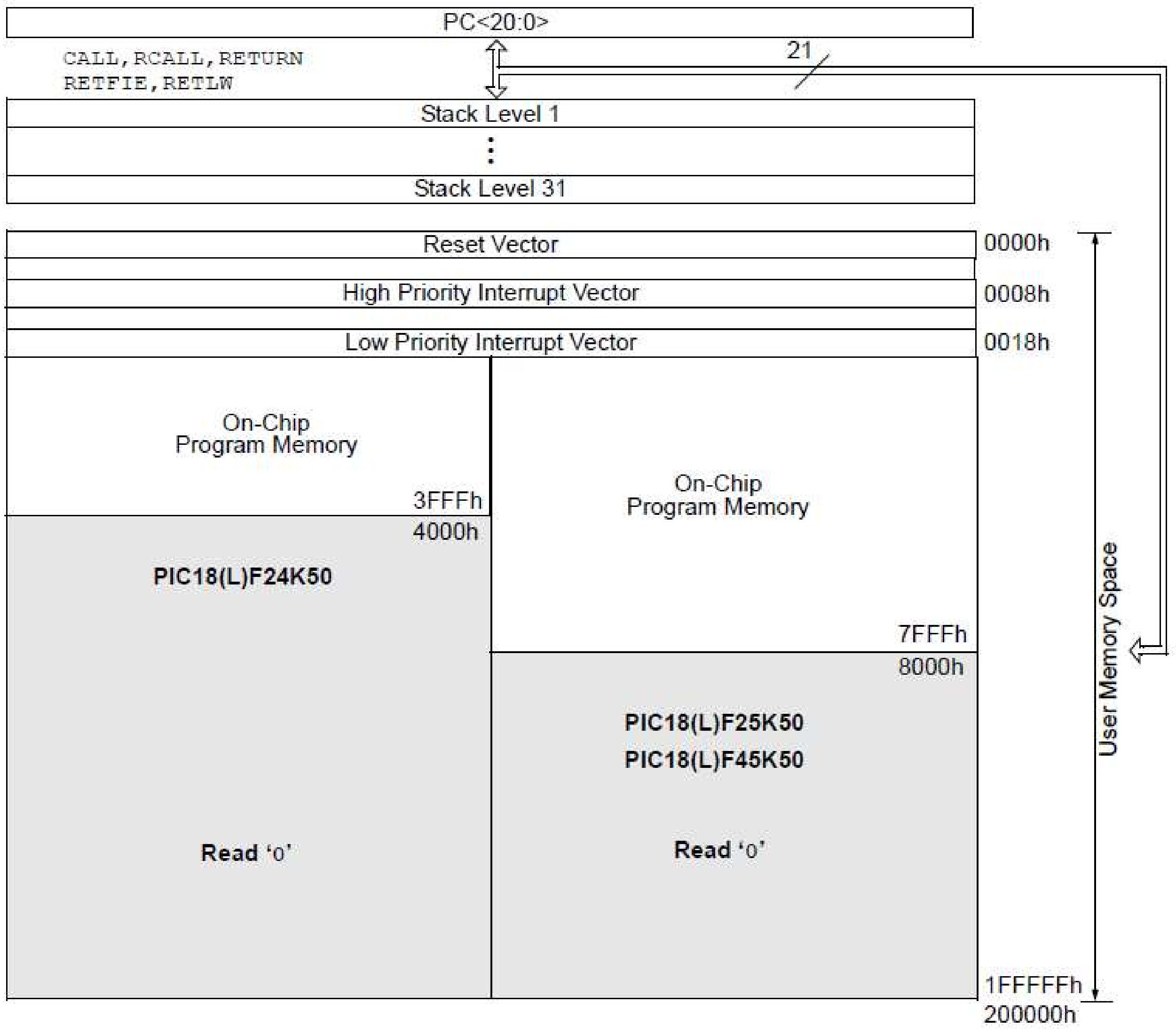
bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{Borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)[(1)]
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

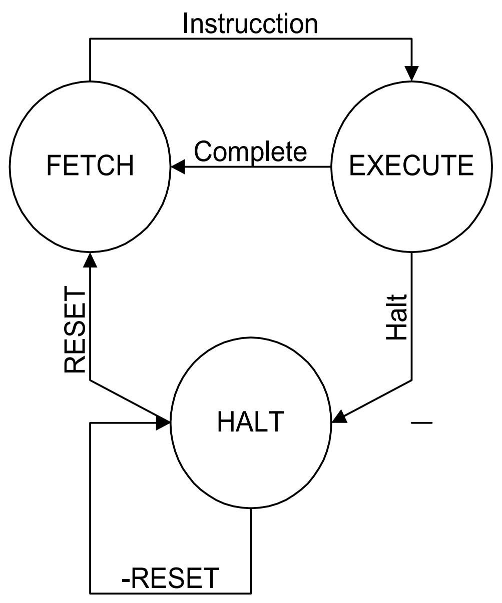
bit 0 **C:** Carry/$\overline{\text{Borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)[(1)]
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note 1: For $\overline{\text{Borrow}}$, the polarity is reversed. 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-order or low-order bit of the source register.

PROGRAM MEMORY MAP AND STACK FOR PIC18(L)F2X/45K50 DEVICES



Sequential logic system



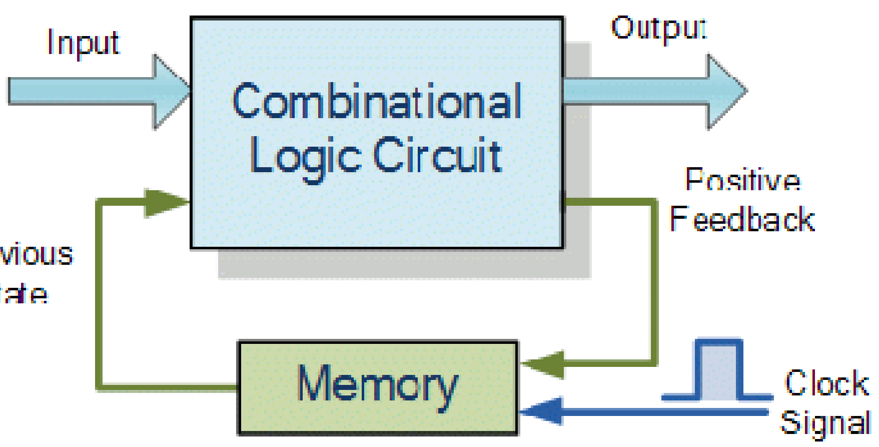
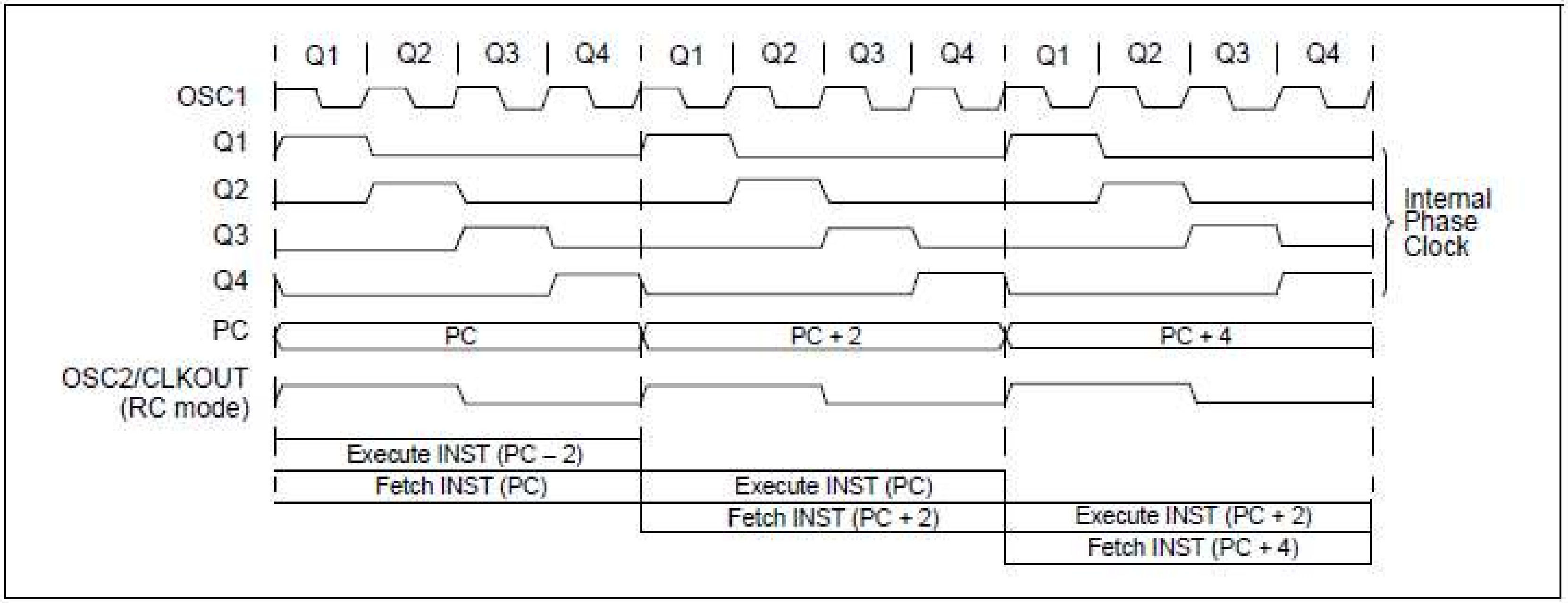
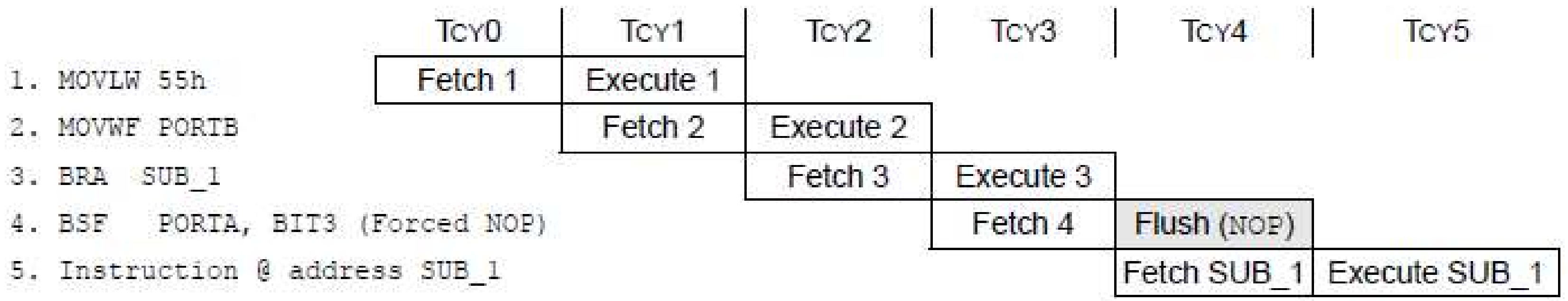


FIGURE 5-3: CLOCK/INSTRUCTION CYCLE



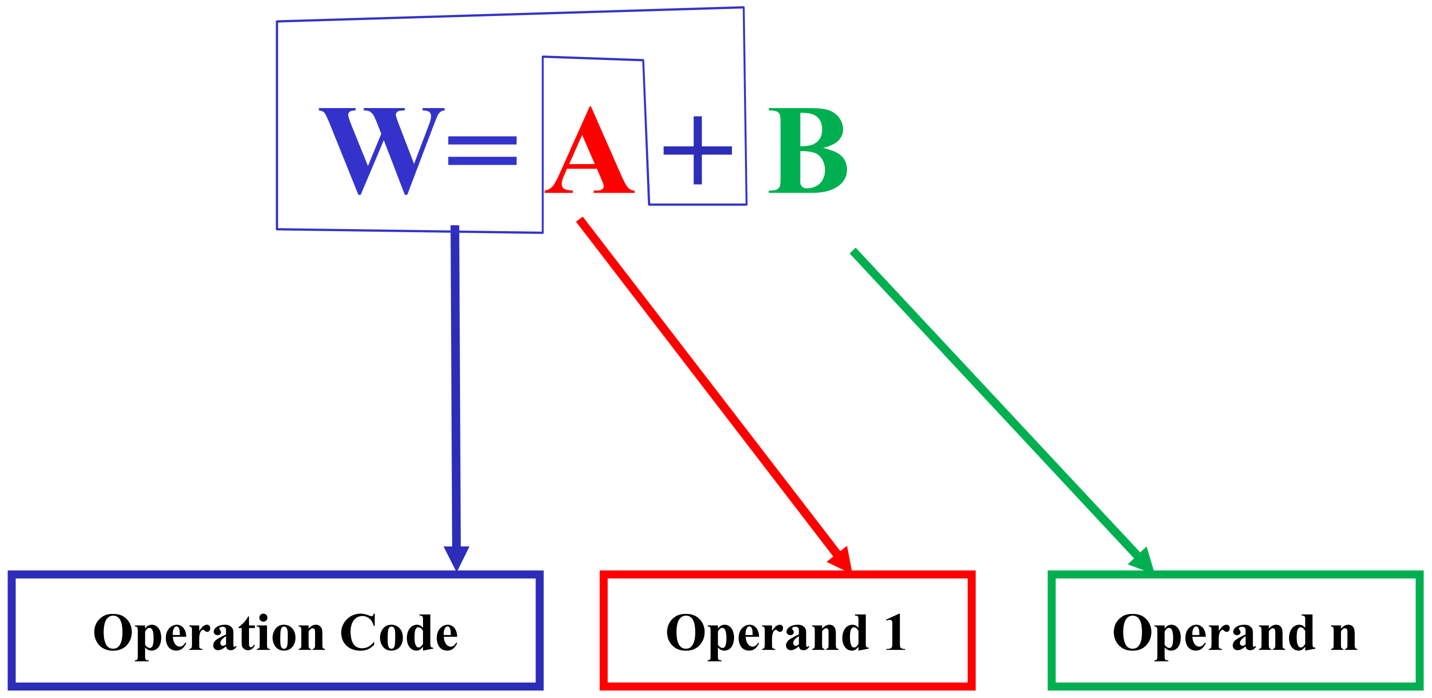
EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW



	$T_{CY}0$	$T_{CY}1$	$T_{CY}2$	$T_{CY}3$	$T_{CY}4$	$T_{CY}5$
`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. BRA SUB_1`			Fetch 3	Execute 3		
`4. BSF PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush (NOP)	
`5. Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

Instructions



Instruction Format

Byte-oriented file register operations

15 — 10	9	8	7 — 0
OPCODE	d	a	f (FILE #)

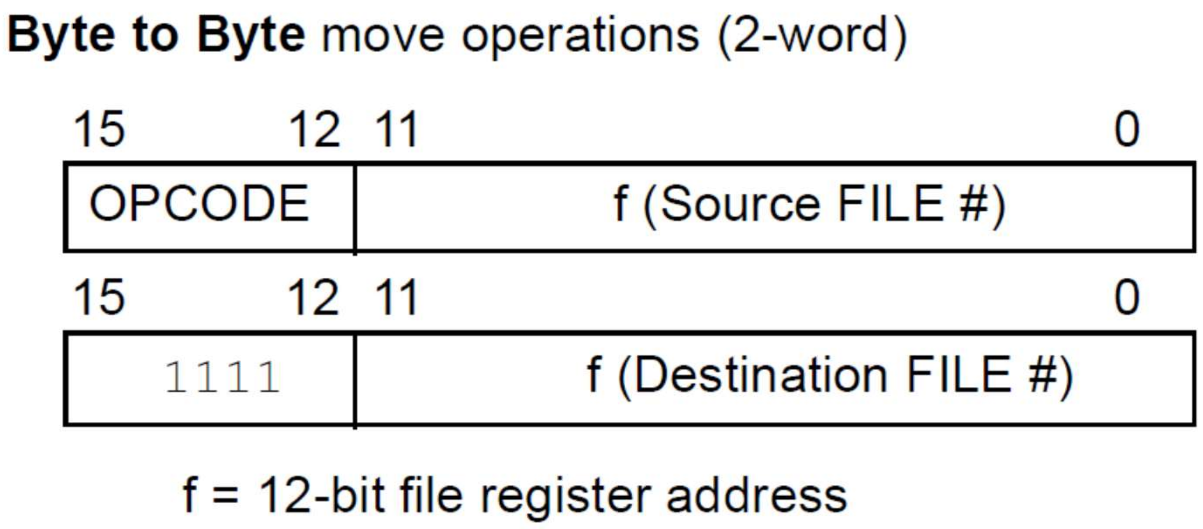
d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

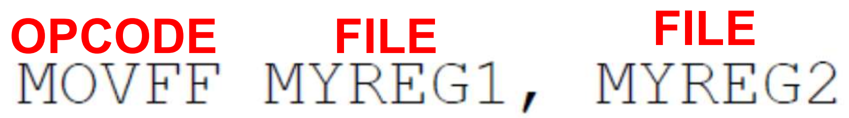
Example Instruction

```
OPCODE  f        d  a
ADDWF   MYREG,   W, B
```

Instruction format

Byte to Byte (or file to file) oriented instructions


Byte to Byte move operations (2-word)
15
12
11
0
OPCODE
f (Source FILE #)
15
12
11
0
1111
f (Destination FILE #)
f = 12-bit file register address



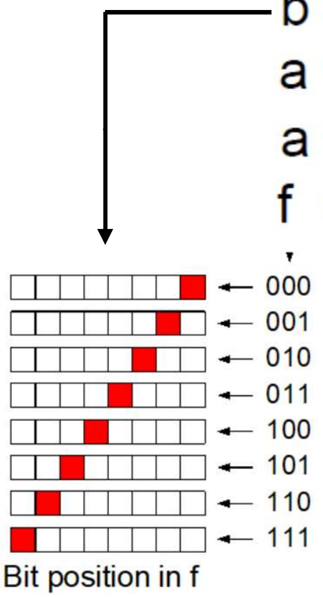
```
MOVFF MYREG1, MYREG2
```

Instruction format

Bit oriented instructions

15 12	11 9	8	7 0
OPCODE	b (BIT #)	a	f (FILE #)

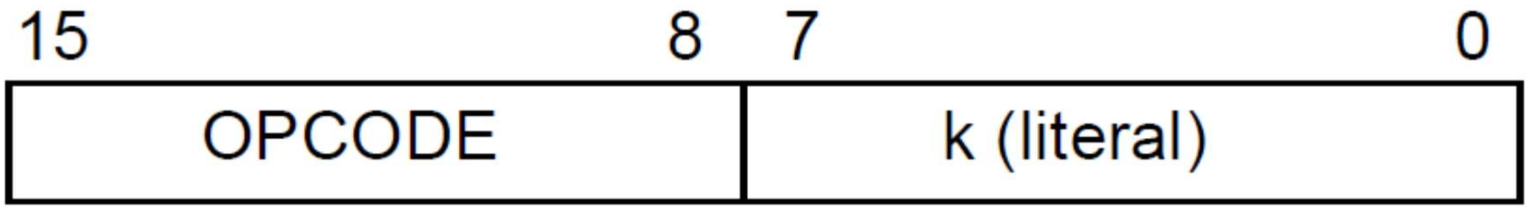
b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address



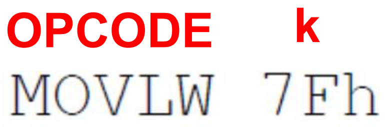
```
OPCODE   FILE      BIT NO   a
BSF      MYREG,    bit,     B
```

Instruction format

Literal instructions



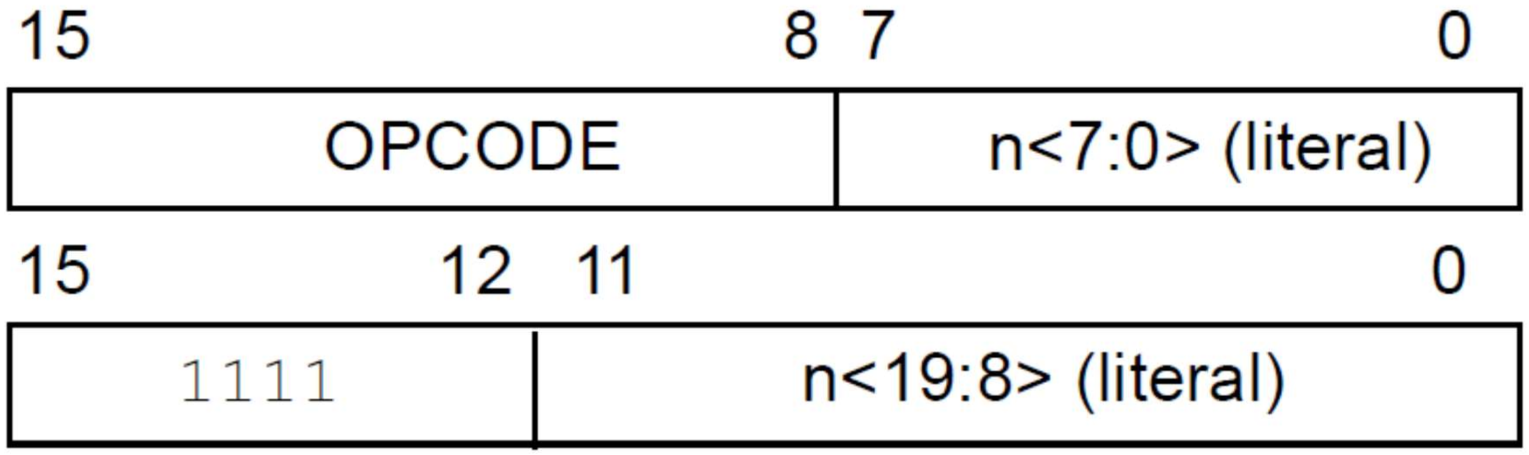
k = 8-bit immediate value



```
MOVLW 7Fh
```

Instruction format

Flow control instructions



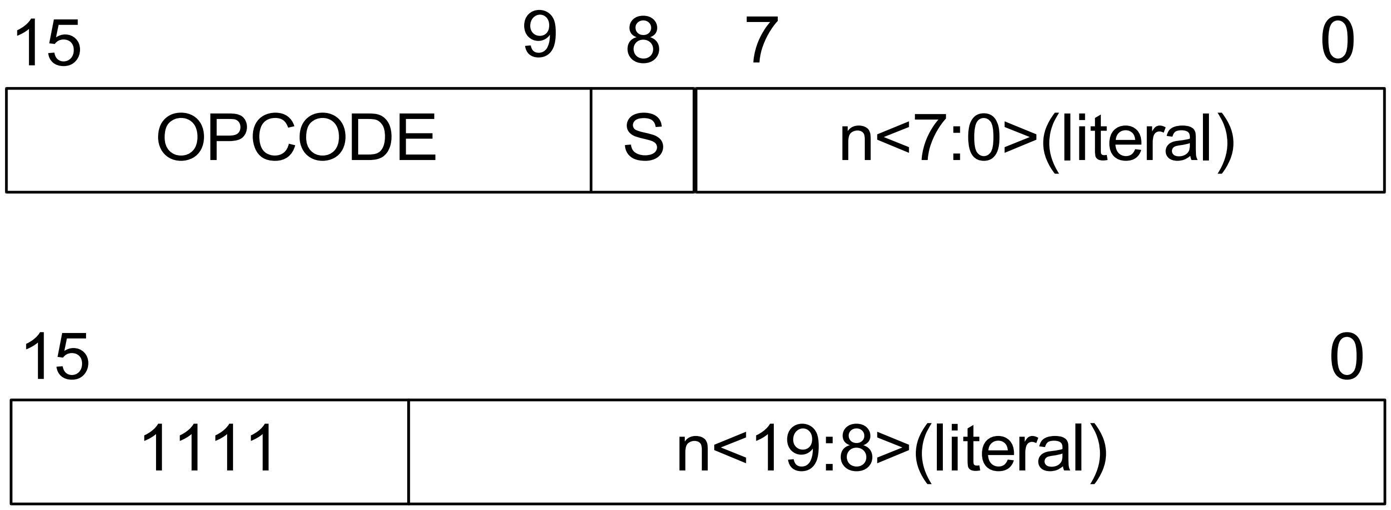
n = 20-bit immediate value

OPCODE **Literal address**

```
GOTO Label
```

Instruction format

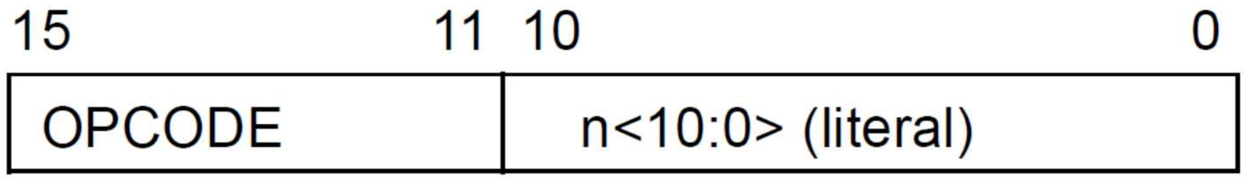
Flow control (CALL)



S = Fast bit

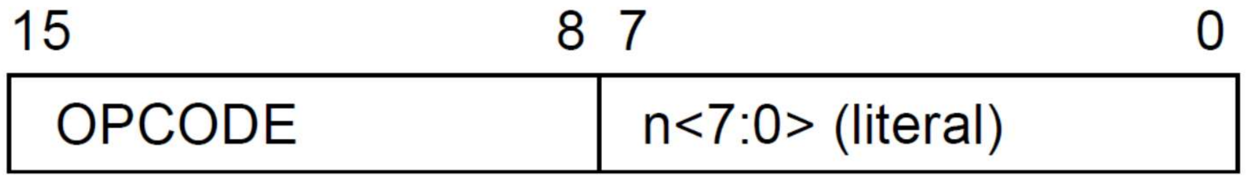
Instruction format

Flow control (BRA)



OPCODE Literal address

```
BRA  MYFUNC
```

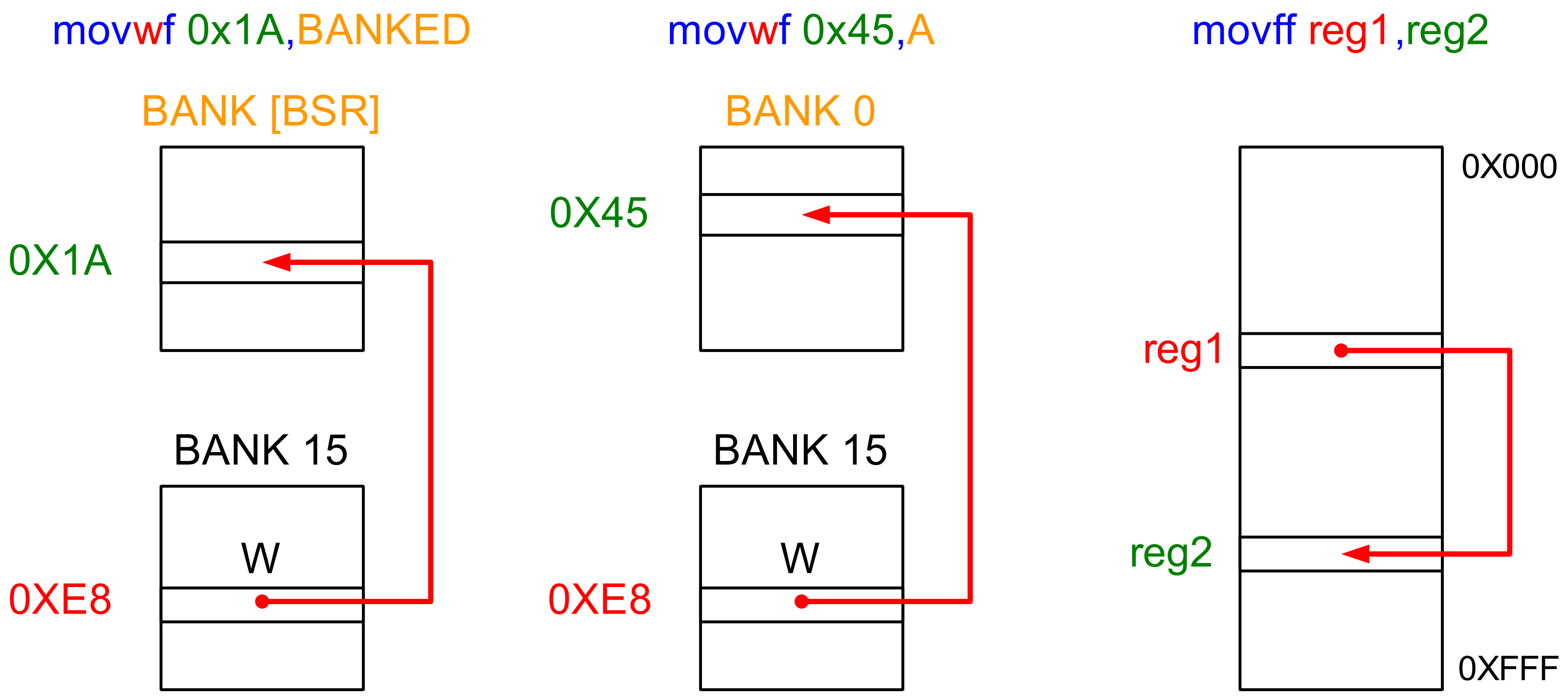


OPCODE Literal address

```
BC  MYFUNC
```

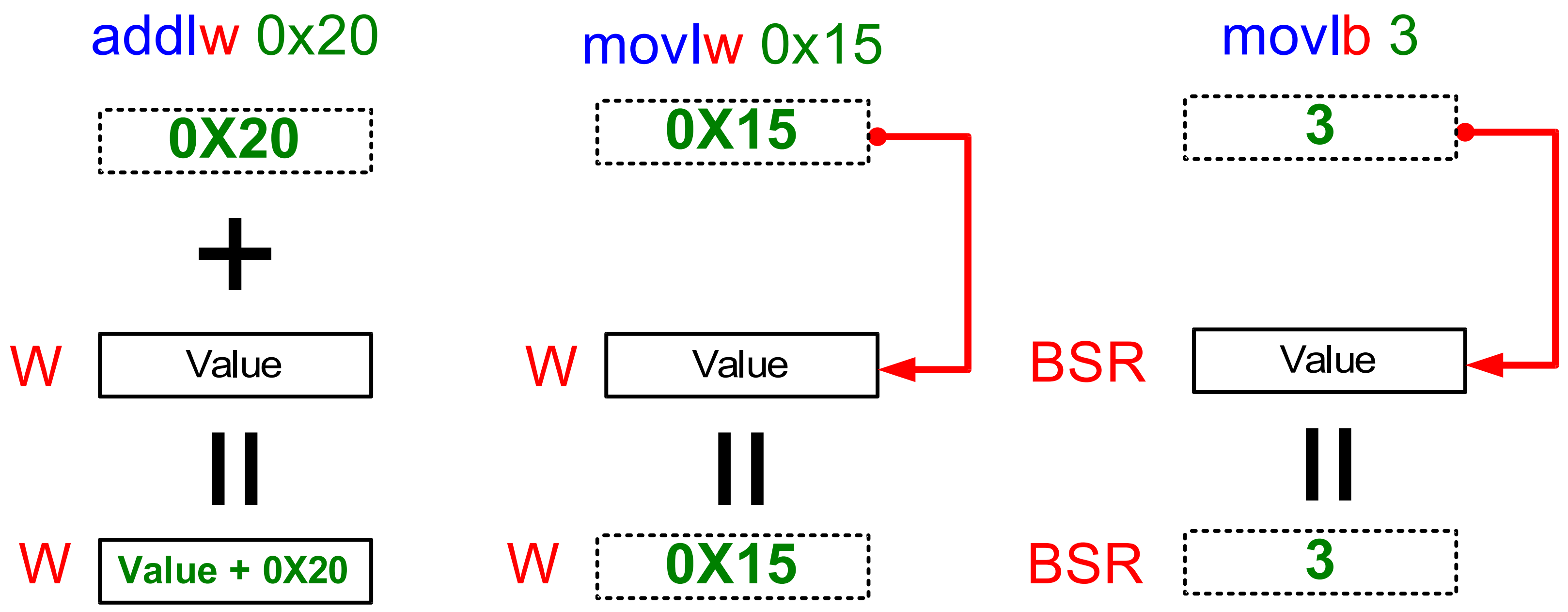
Addressing modes

Direct register



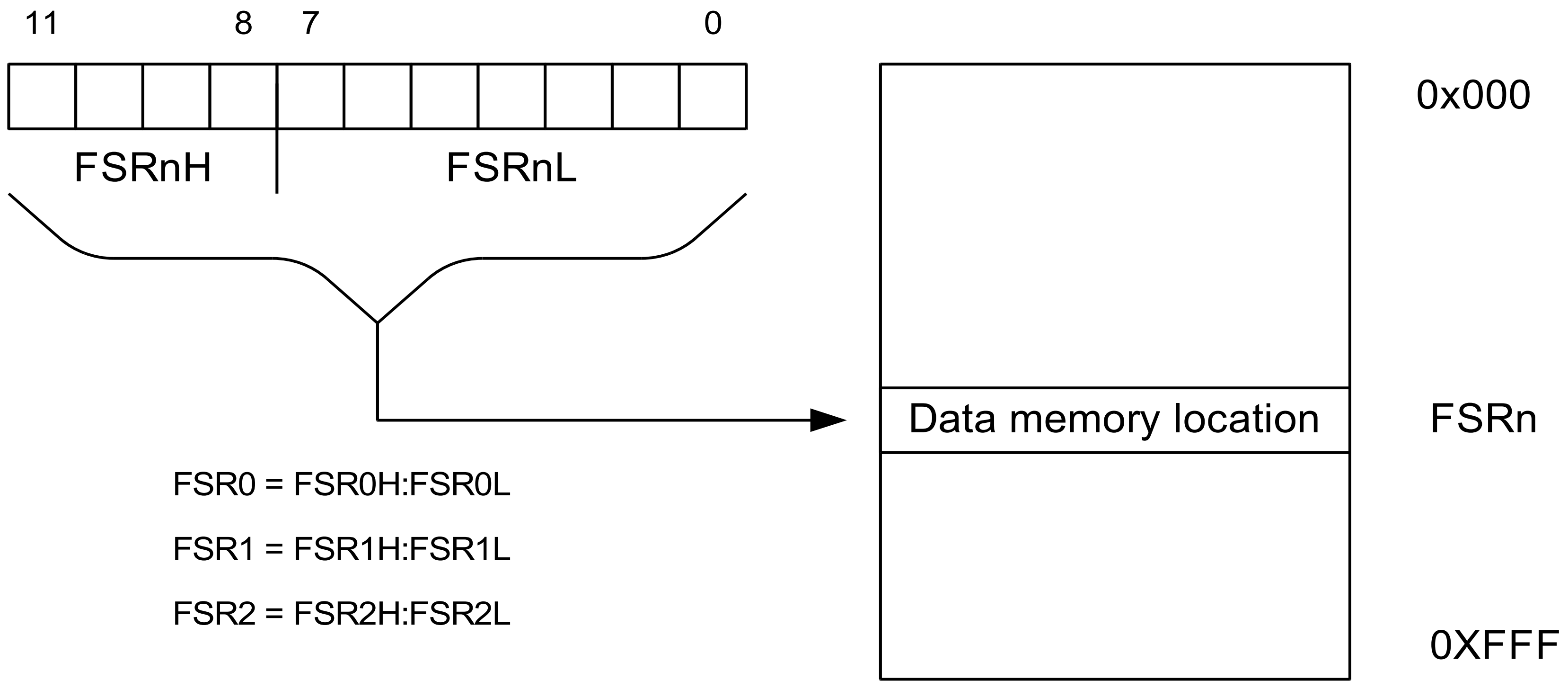
Addressing modes

Immediate (or literal)



Addressing modes

Indirect



FSR0 = FSR0H:FSR0L

FSR1 = FSR1H:FSR1L

FSR2 = FSR2H:FSR2L

Indirect addressing mode

Each file register can be indirectly accessed using 5 SFR registers that perform an action to the address when the instruction is executed.

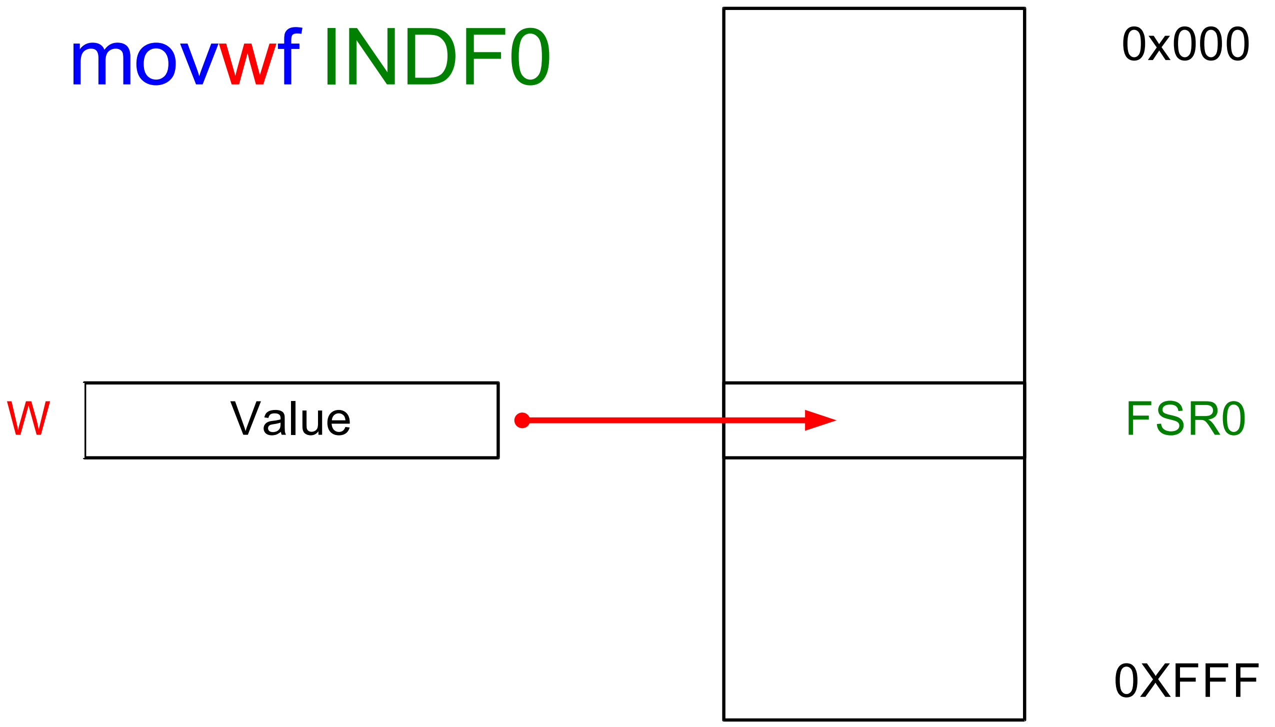
- Don’t do nothing (just access) FSRn = INDFn (n=0..3)
- Decrement the address before FSRn = POSTDECn (n=0..3)
- Increment the address after FSRn = POSTINCn (n=0..3)
- Increment the address before FSRn = PREINCn (n=0..3)
- Add the W register (signed) al FSRn = PLUSWn*

**The FSRn register is not modified*

Indirect addressing mode

Move (copy) the content of W to the file register that is addressed by the FSR0 register and do nothing to the FSR0 register.

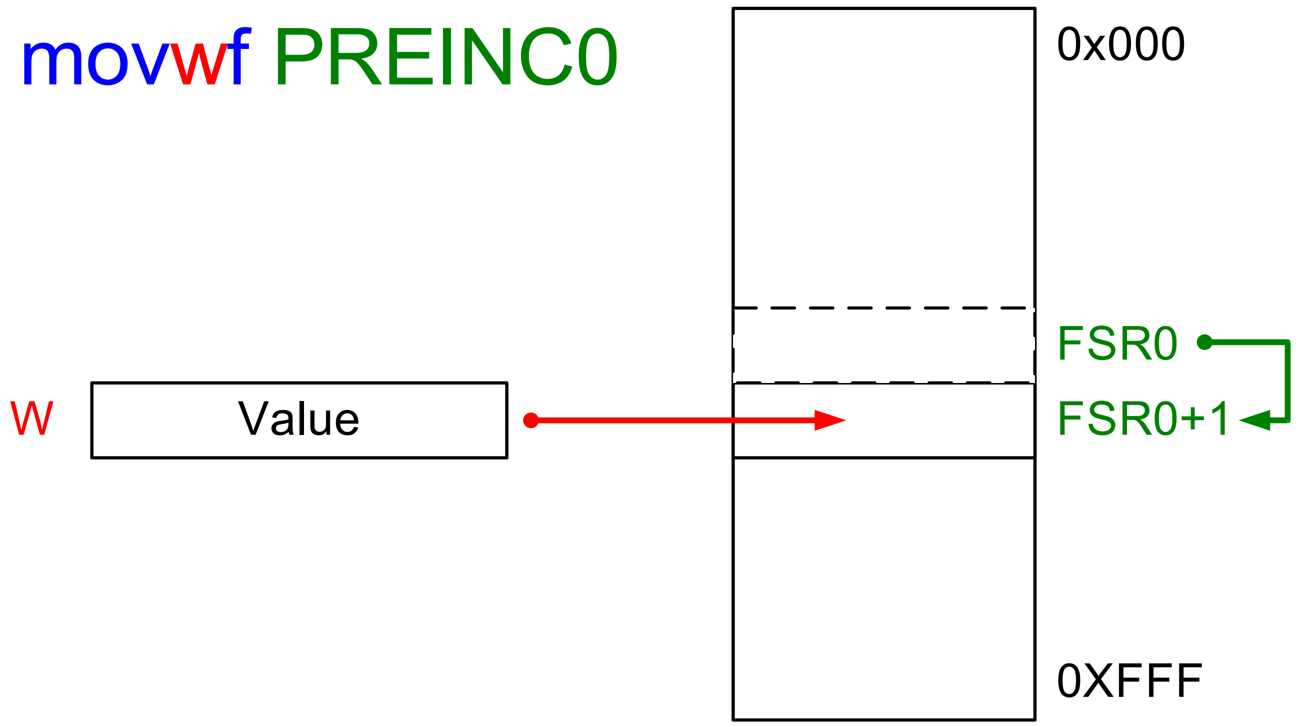
movwf INDF0



Indirect addressing mode

Increment the addressing register FSR0 and then move (copy) the content of the W register to the address stored in the FSR0 (pointed by)

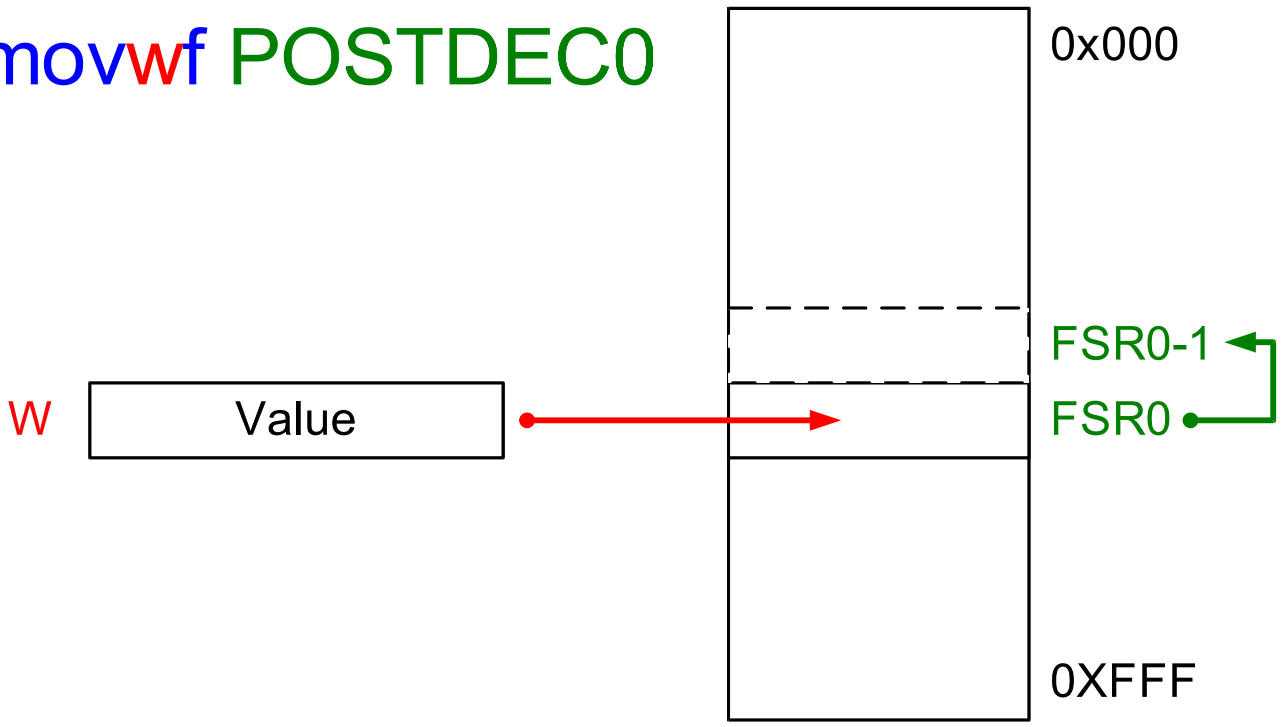
movwf PREINC0



Indirect addressing mode

Move (copy) to the content of the memory location (file register) pointed (addressed) by FWR0 and after that (post), increment the FSR0 (to point to the next location)

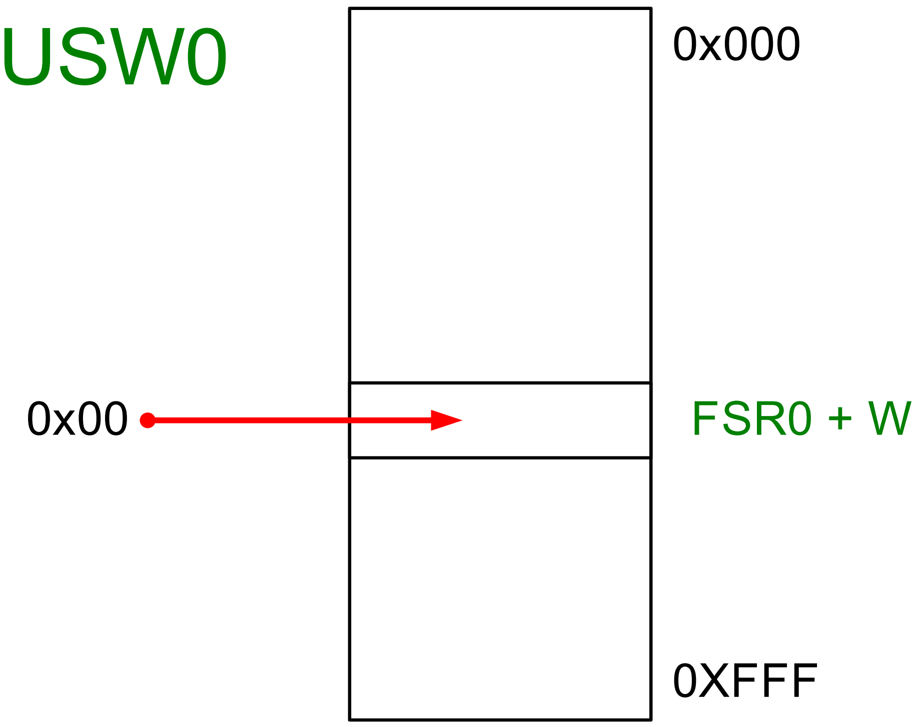
movwf POSTDEC0



Indirect addressing mode

Clears (copy a 0) to the content of the memory location (file register) pointed (addressed) by FWR0 without modifying neither W or the FSR0 register or W

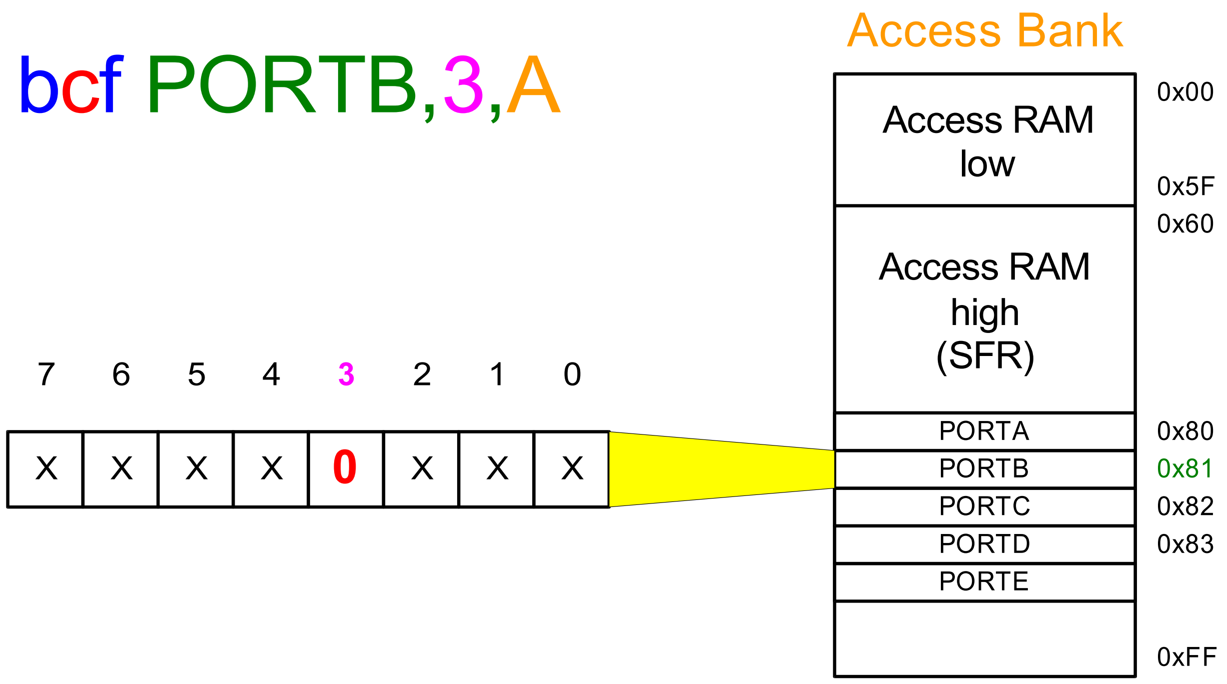
clrf PLUSW0



Bit addressing mode

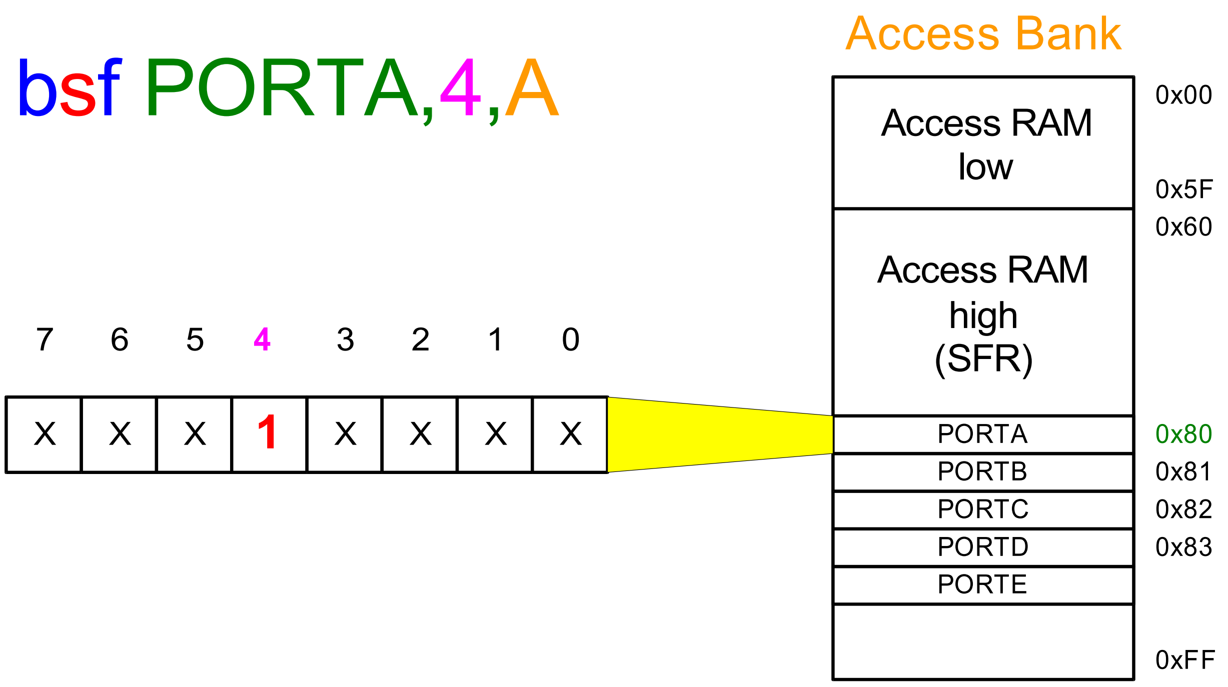
Bit direct

bcf PORTB,3,A

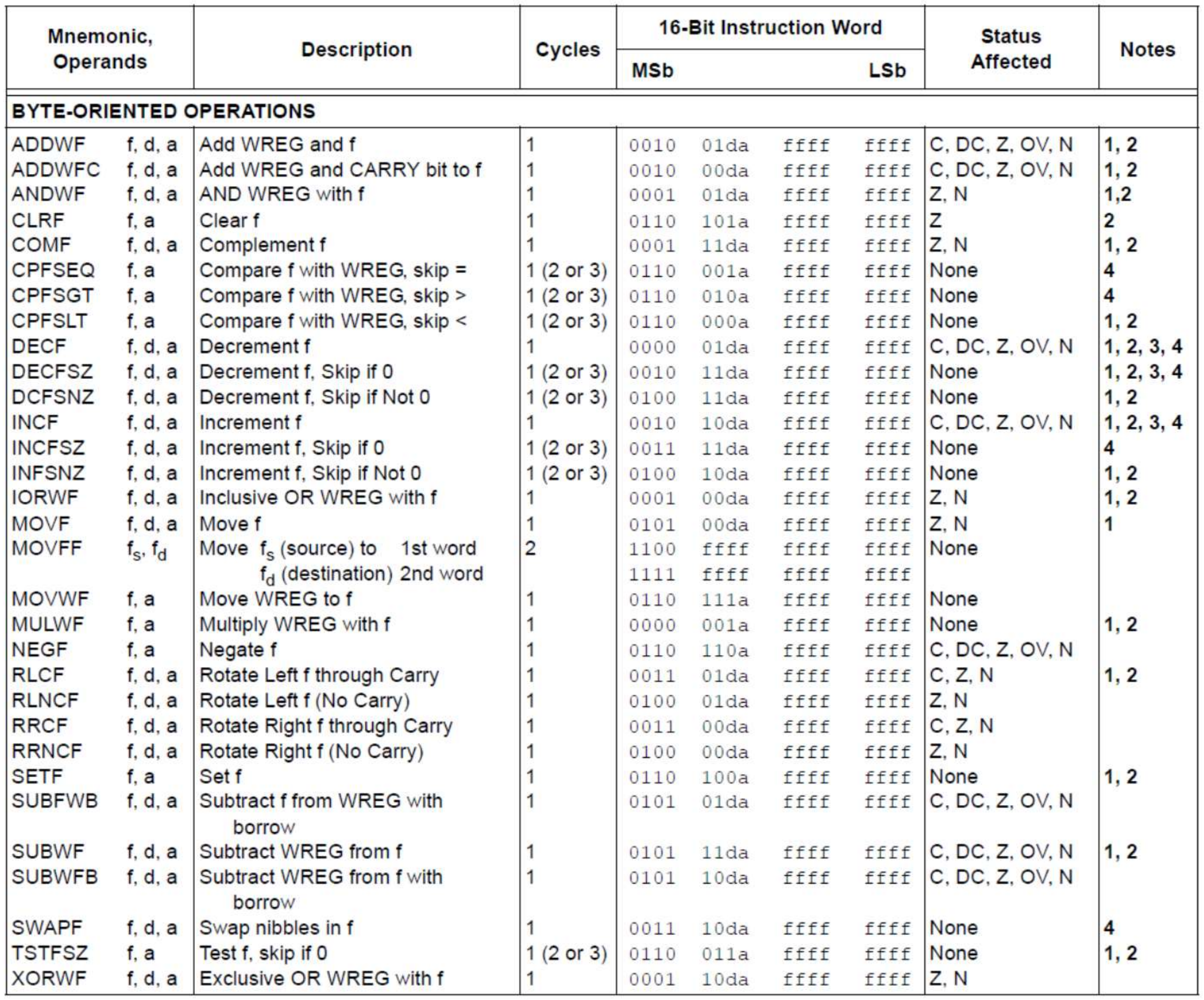


Bit addressing mode

Bit direct



The instruction set (75)

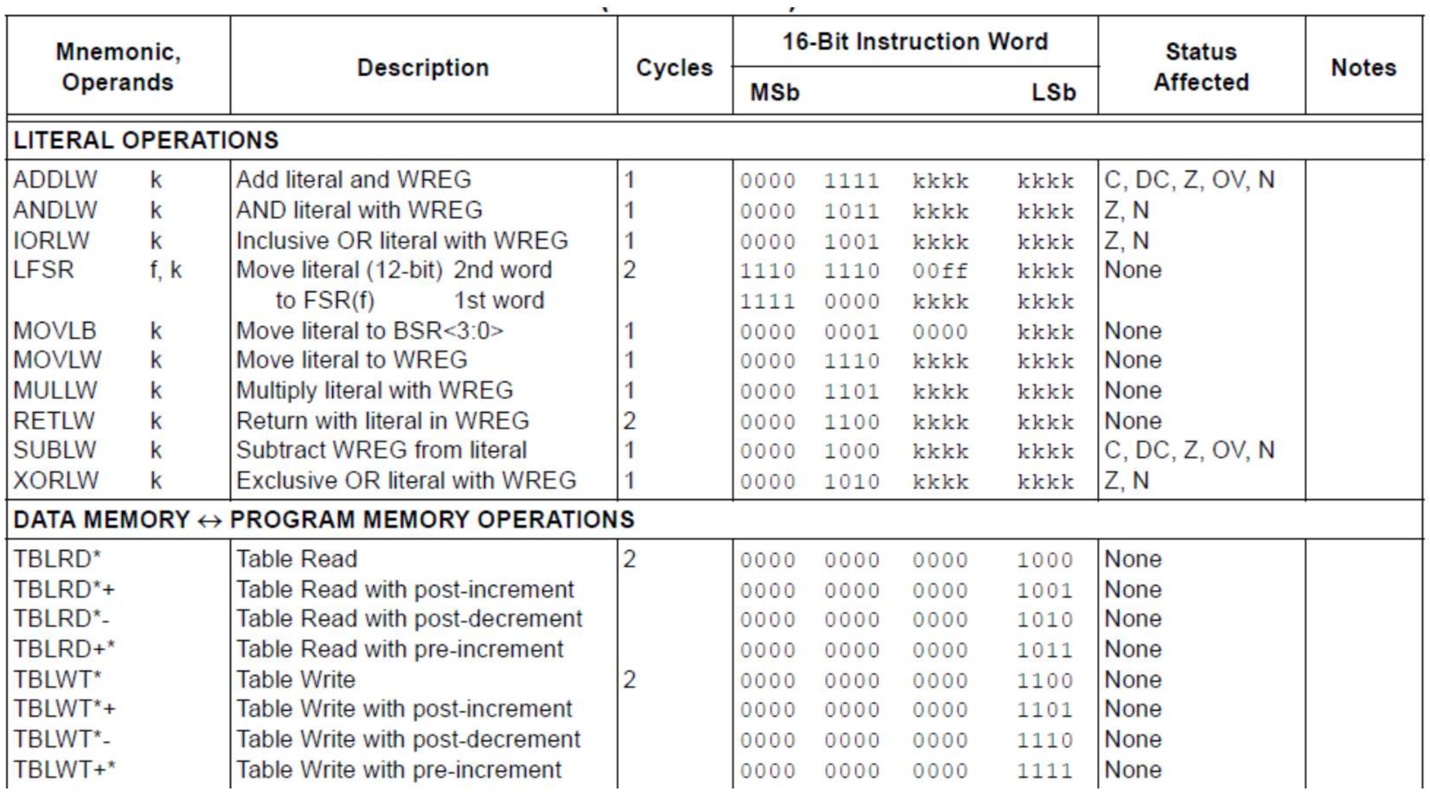


Mnemonic,	Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word	2	1100	ffff	ffff	ffff	None	
		f_d (destination) 2nd word		1111	ffff	ffff	ffff		
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

The instruction set

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

The instruction set



Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Basic instruction examples

Examples of data movement

Move the content of W regiser to the file regiser 0x30 in the access bank

- movwf 0x30,A

Move the content of file register 0x30 to file register 0x40

- movff 0x030,0x040

Move the file register 0x40 to the W register in access bank

- movf 0x040,W,A

Load the value 0x200 al registro FSR0

- lfsr FSR0,0x200

Basic instruction (arithmetic)

The addition instruction

INSTRUCT	DESC
addwf f,d,a	Add WREG and f
addwfc f,d,a	Add WREG, carry bit and f
addw k	Add literal to W

Basic instruction (arithmetic)

Examples of the add instruction

Add the content of the W register to the file regiser 0x40 in the access bank and leave the result in W

- addwf 0x40,W,A

Add the content of the W regiter to the filer regiser 0x040 in the register bank 0x02 and leave the result in W

- movlb 0x02
- addwf 0x40,W,BANKED

Basic instruction (arithmetic)

Examples of the ADD instruction

Write a sequence of instructions to increment by 3 units the content of register files 0x030 to the 0x032 (Access bank)

```
•    movlw 0x03         ;Set the W register with constant 3
•    addwf  0x30,F,A    ;Add W to register 30h, result in 30h
•    addwf  0x31,F,A    ;Add W to register 31h, result in 31h
•    addwf  0x32,F,A    ; Add W to register 32h, result in 32h
```

Basic instruction (arithmetic)

Examples of the ADD instructions

Write a instruction sequence to add 10 units to the files register 0x300 to 0x303

```
movlw 0x0A              ;Move the 10 constant to W register
lfsr    FSR0,0x300      ;Load the FRS0 with address 0x300
addwf  POSTINC0,F       ;Add W to  [FSR0] after FSR0 = 0x300 +1
addwf  POSTINC0,F       ;Add W to  [FSR0] after FSR0 = 0x301 +1
addwf  POSTINC0,F       ;Add W to  [FSR0] after FSR0 = 0x302 +1
addwf  POSTINC0,F       ;Add W to  [FSR0] after FSR0 = 0x303 +1
```

Examples using the MPLAB