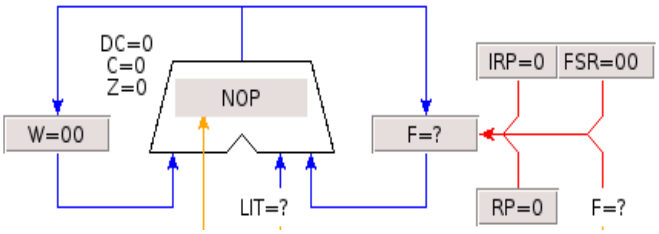
Introduction to the PIC16F877 architecture:

Recall from our exploration of the 74181 circuit that an arithmetic/logic unit has two data inputs 'a' and 'b' and a data output 'f'. Function select inputs 's' determine the ALU operation that is carried out, such as 'f = a AND b', 'f = a + b', f = -a, f = b + 1, f = 0. Note that the ALU may operate on both, one or none of the input variables a,b. The ALU also included some status bits, or flags:

- a carry bit that is set when there is an arithmetic overflow;
- a zero bit that is set when the result of the ALU operation is zero, that is f = 0.

Recall from our exploration of counters and multipliers that a sequential circuit requires a series of flip/flops known as an Accumulator to store intermediate results (the result of the current operation, or machine cycle, for use during the next operation).

Central to the operation of the PIC is the Arithmetic Logic Unit. The data paths to the ALU are shown in blue. The left input to the ALU is connected to the output of an accumulator register called W. The right input is connected to an array of 512 file registers. A memory address (path shown in red) selects which file register output will be connected to the ALU input. Alternately, a literal (constant) can be selected as the right input to the ALU. The output of the ALU is connected to both the W and file register inputs.



The yellow line represents the function select inputs to the ALU. These bits, known as the opcode, as well as a literal constant or file register address, are coded as a bit pattern that constitutes a program instruction that is stored in the instruction register (IR).

Shown below is a partial list of the PIC instruction set, including the 'incf' instruction.

Byte-oriented file register operations

13	8	7	6	0
OPCODE		d	f (FILE #)	

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

COMF	f, d	Complement f	1	00	1001 dfff ffff
DECF	f, d	Decrement f	1	00	0011 dfff ffff
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011 dfff ffff
INCF	f, d	Increment f	1	00	1010 dfff ffff
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111 dfff ffff
IORWF	f, d	Inclusive OR W with f	1	00	0100 dfff ffff
MOVF	f, d	Move f	1	00	1000 dfff ffff

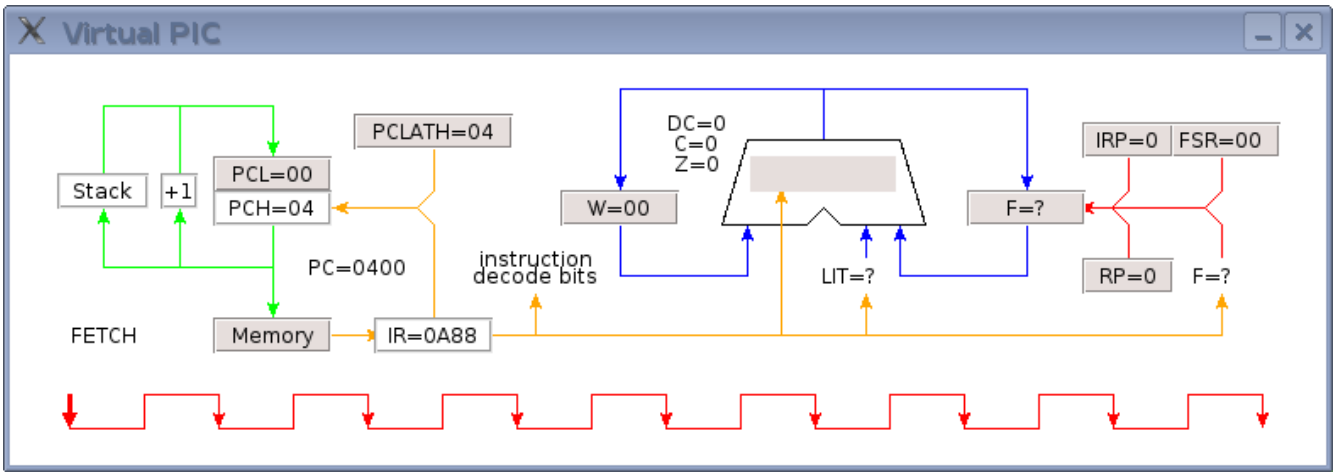
The 14-bit opcode for the 'incf' instruction is '00 1010 dfff ffff'.
The PORTD register address is f=8 or 0001000 in binary.
The 'incf PORTD,F' instruction is '00 1010 1000 1000 = 0x0A88'
The 'incf PORTD,W' instruction is '00 1010 0000 1000 = 0x0A08'.

PIC instruction execution

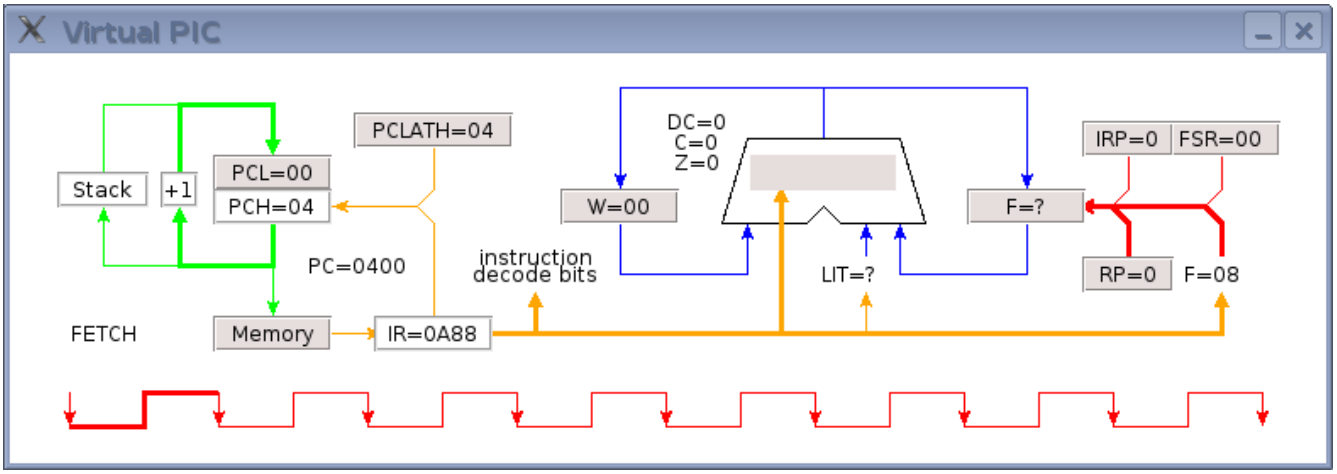
The program counter (PC) points to the next instruction in program memory that will be loaded in the instruction register. The PIC uses an instruction pipeline that pre-fetches the next instruction while the current instruction is being executed. Most instructions execute in four clock cycles:

	Data operations:	Program operations:
1)	Decode file register address	Increment program counter
2)	Read file register data to ALU	Instruction fetch
3)	operate on file register data	Instruction fetch
4)	write result to file register or W	Instruction fetch + store in IR

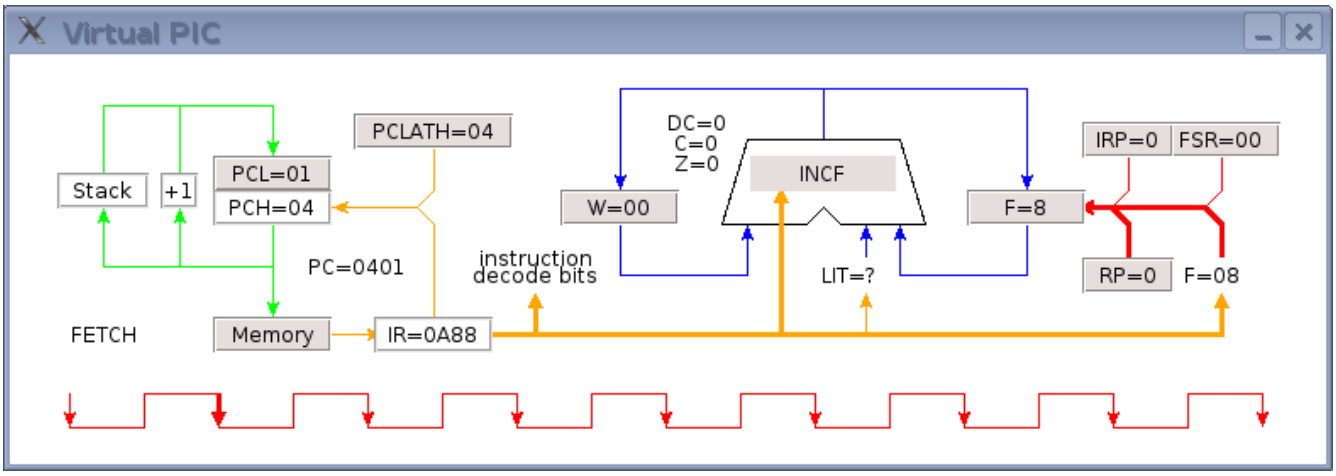
A user program, begins at program memory address PC=0x0400. Lets consider the step-by-step processing of the PIC instruction that increments the PORTD register, '`incf PORTD,F`'.



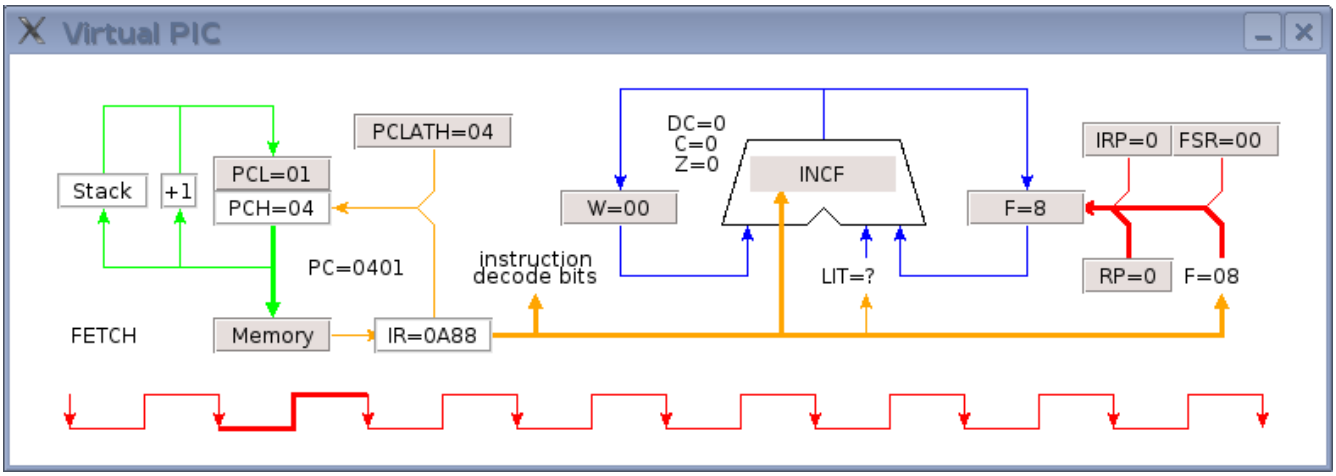
The binary representation of the 'incf PORTD' instruction (0x0A88) has been fetched from program memory location 0x0400 pointed to by the program counter (PC) and placed in the Instruction register (IR) on the falling edge of the PIC clock. In this case, the IR contains the 7-bit address of file register PORTD (8) as well as the code for the ALU operation that will be performed. The IR also specifies several decode bits that serve to control the internal operation of the PIC.



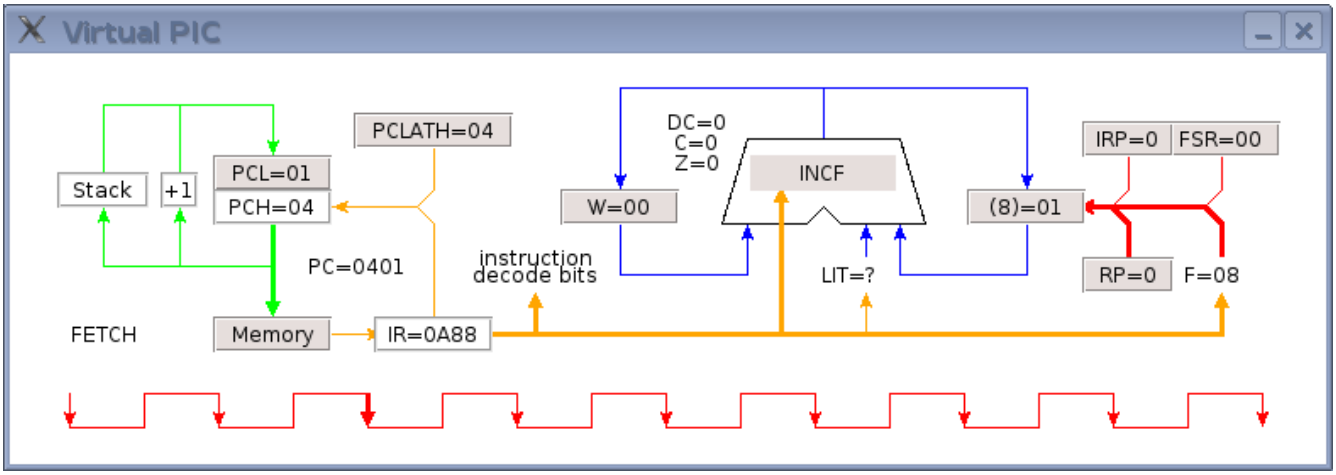
During the first clock cycle, the IR output is decoded as the file register address for PORTD (8), the ALU operation and the PIC hardware control bits. The program counter output is routed to an incrementing circuit. The file register read/write control is set to read.



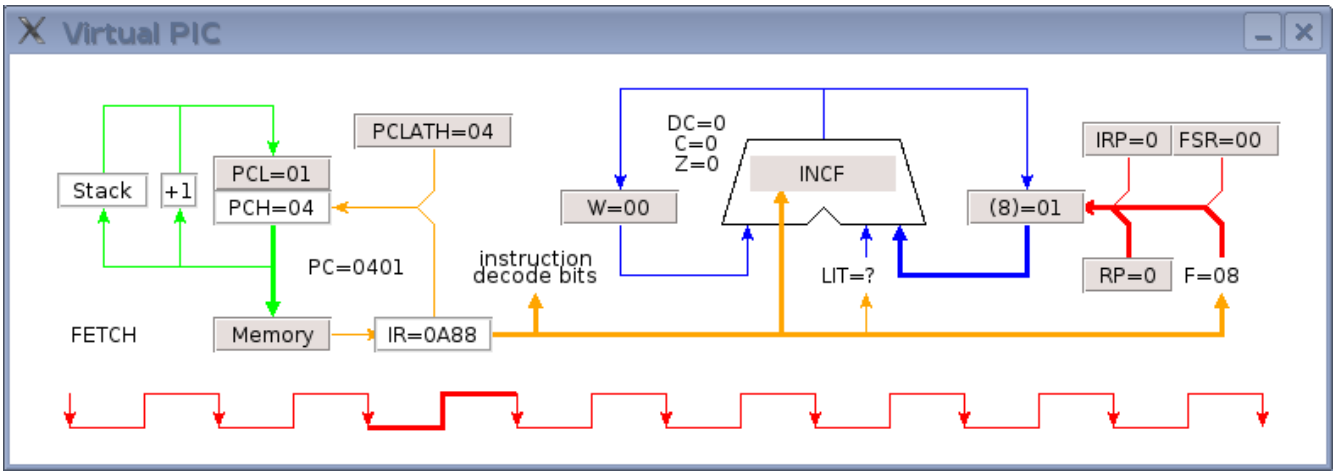
On the falling edge of clock cycle 1, the file register address for PORTD (8) has been decoded and is ready to use. The program counter is loaded with the output of the incrementing circuit, that is, the PC has been incremented and is now pointing to the next program memory location.



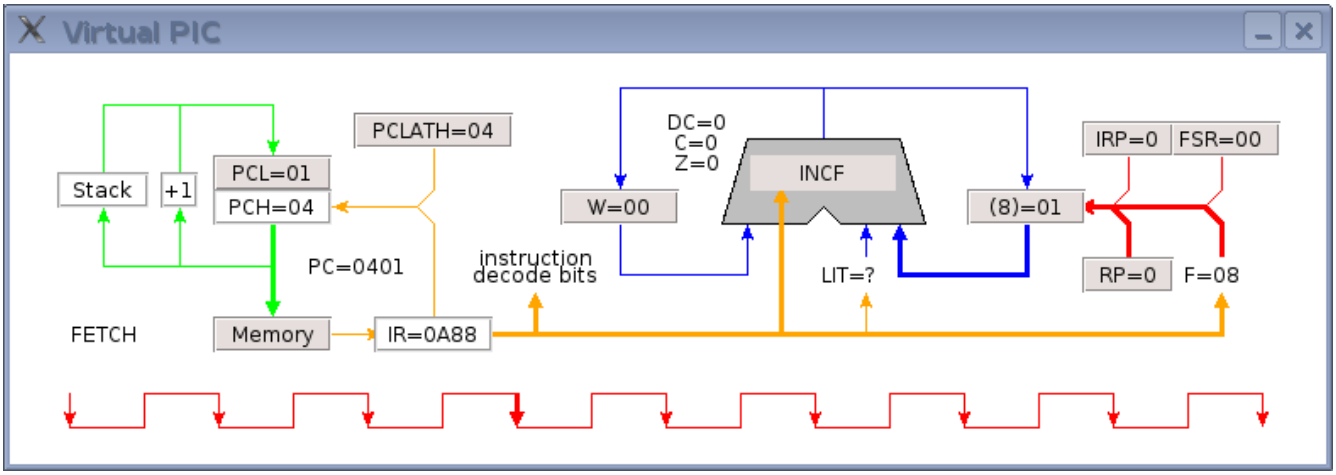
During clock cycle 2, the file register data at address 8 is being accessed. The PC begins to access the next instruction in program memory.



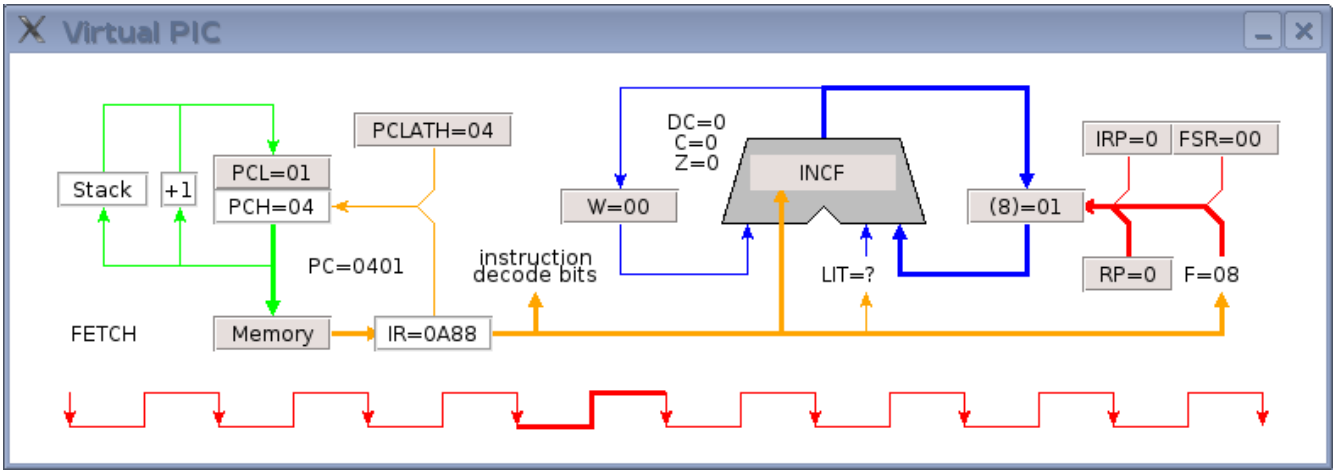
On the falling edge of clock cycle 2, the file register data (0x01) read at address 0x08 is ready to use. The program memory read cycle continues.



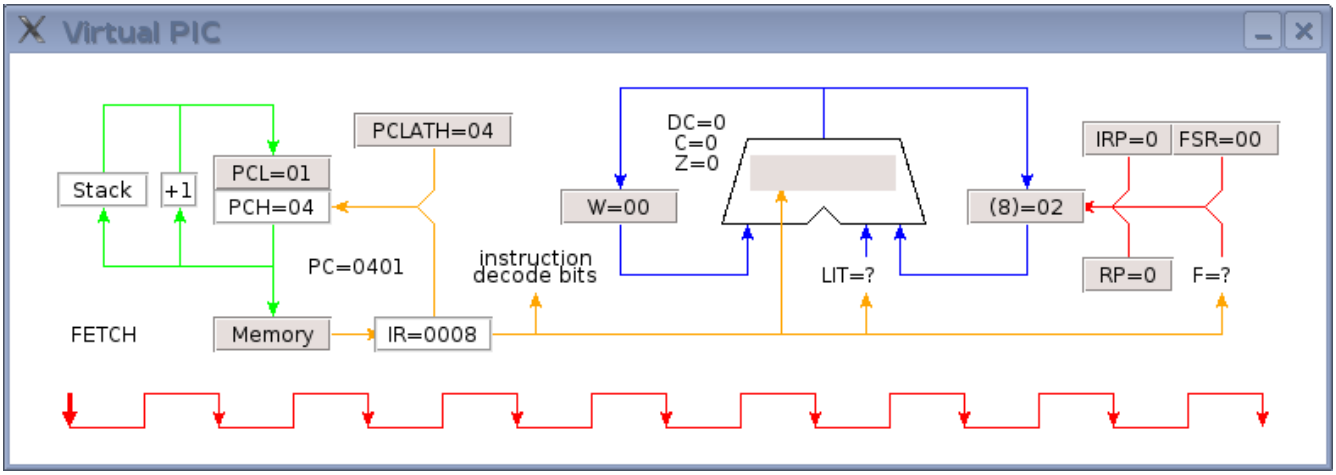
During clock cycle 3, the file register data is operated on by the ALU. The program memory read cycle continues.



On the falling edge of clock cycle 3, the file register data has been incremented and is ready at the output of the ALU. The program memory read cycle continues.

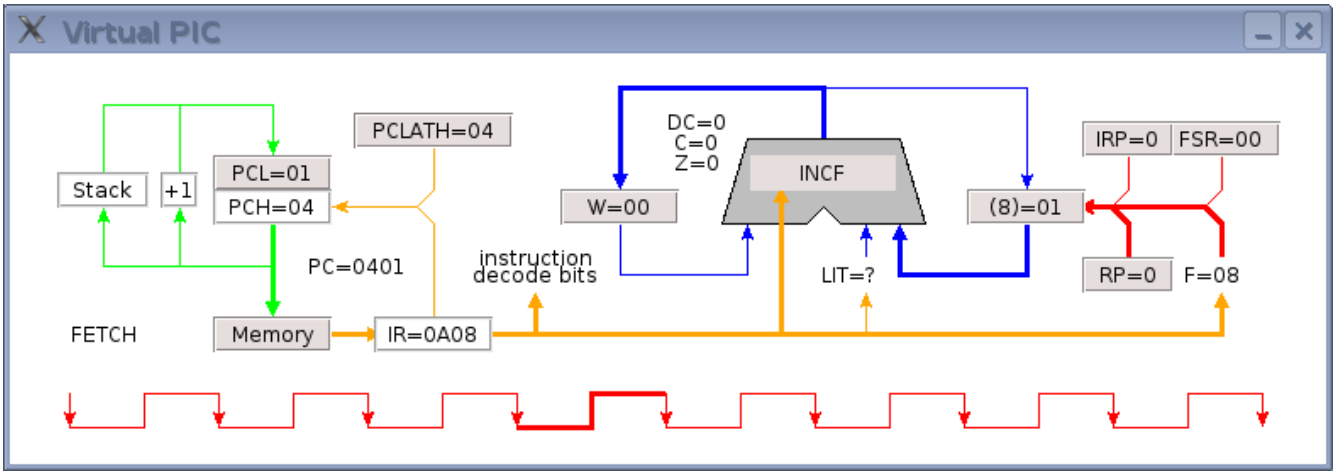


During clock cycle 4, the incremented data is presented to the file register. The file register read/write control is set to write. The program memory read cycle continues.

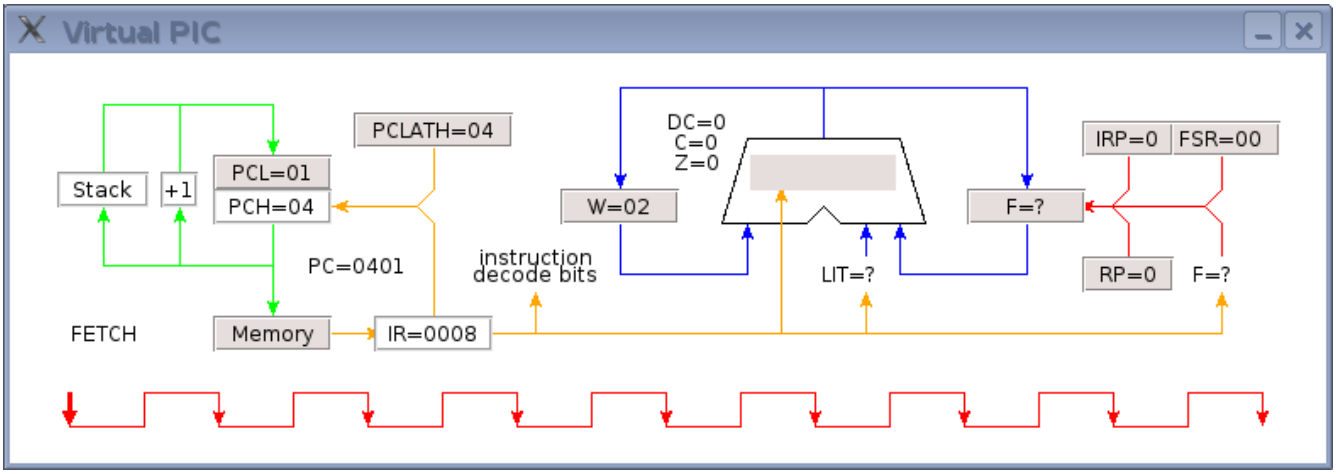


On the falling edge of clock cycle 4, the incremented data (02) is written to file register address 8. The instruction at memory location 0x0401 is written to the instruction register, terminating the current and beginning the next instruction cycle.

The 'incf PORTD,W' instruction selects the accumulator register W as destination. Here, the first three clock cycles are identical to the previous case. The write cycle changes as follows:



During clock cycle 4, the W register is selected as the destination for the ALU data.



The incremented PORTD data is stored in W. PORTD is not changed.

The following table summarizes the complete instruction set for the PIC16F877 microcontroller.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	