The University of Texas at Arlington

Lecture 4
Branching

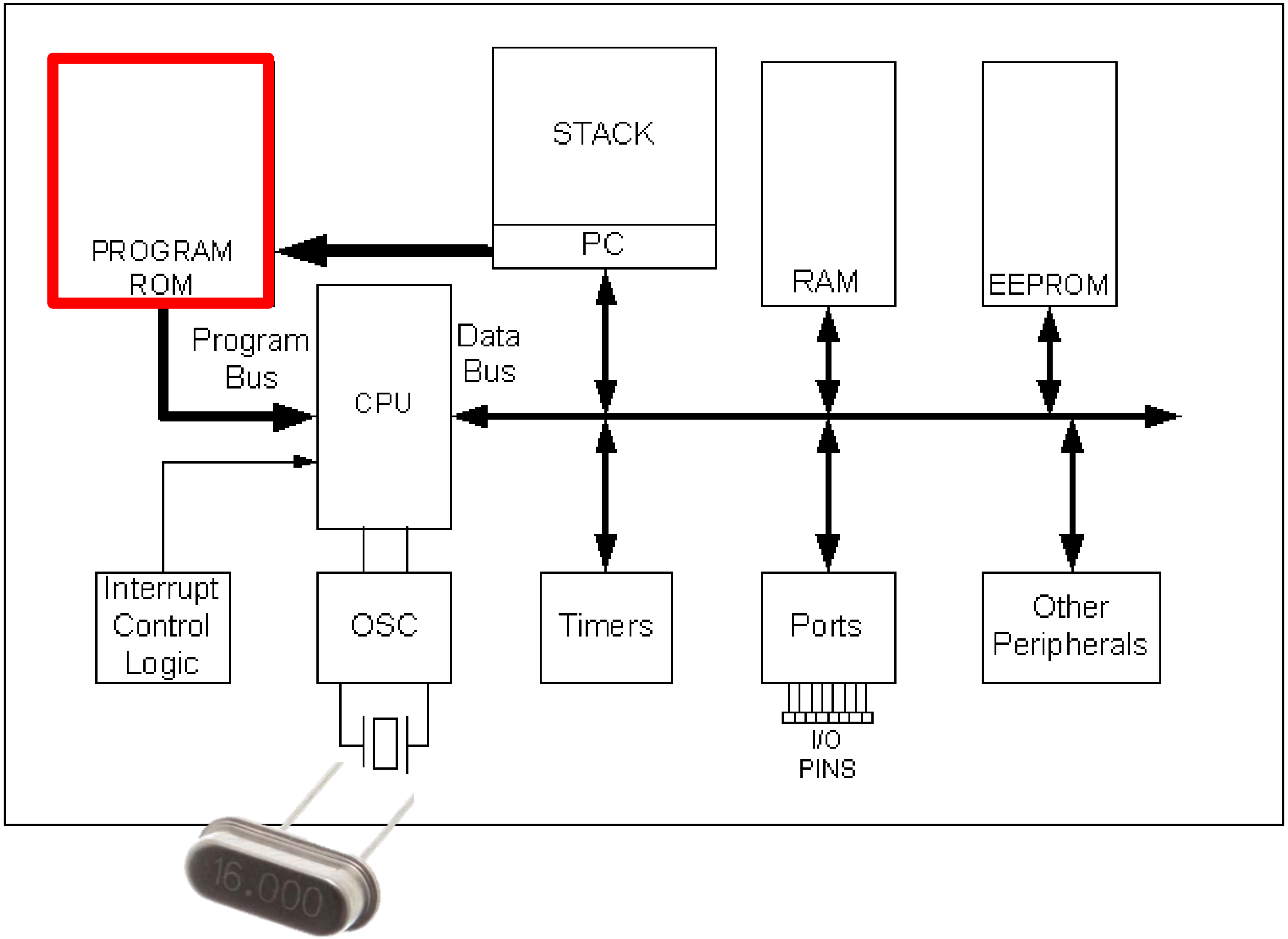
CSE@UTA

CSE 3442/5442

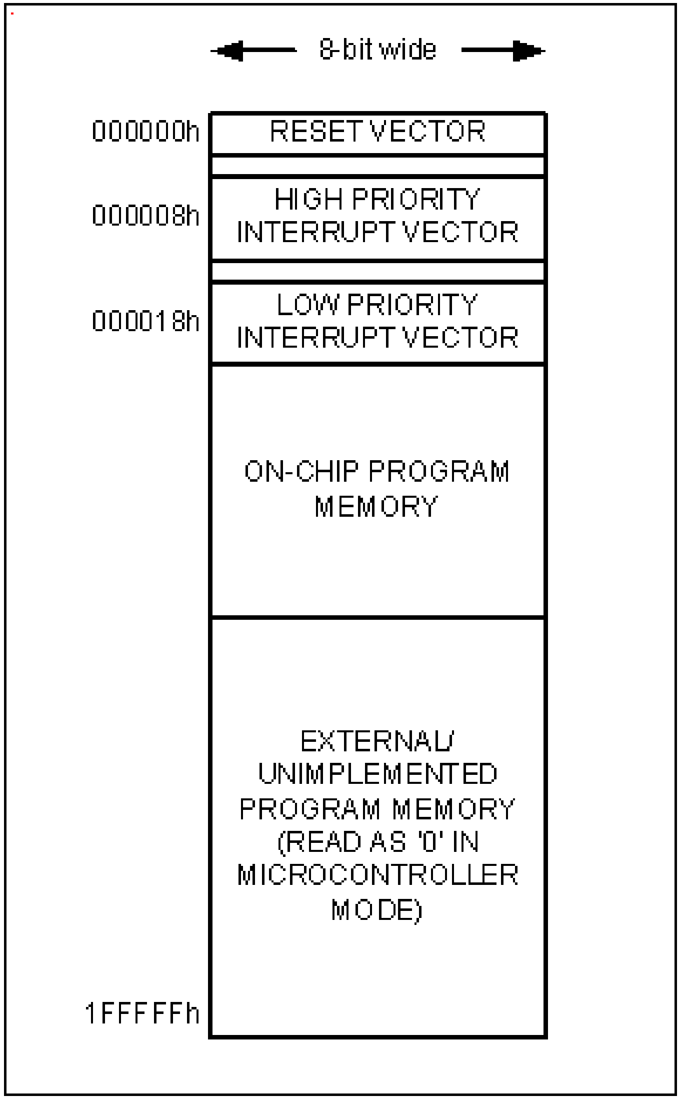
Embedded Systems I

Based heavily on slides by Dr. Gergely Záruba and Dr. Roger Walker

Program ROM


PROGRAM
ROM
STACK
PC
RAM
EEPROM
Program
Bus
CPU
Data
Bus
Interrupt
Control
Logic
OSC
Timers
Ports
Other
Peripherals
I/O
PINS
16.000

PIC18 Program ROM Space Review from Last Lecture



8-bit wide

000000h RESET VECTOR

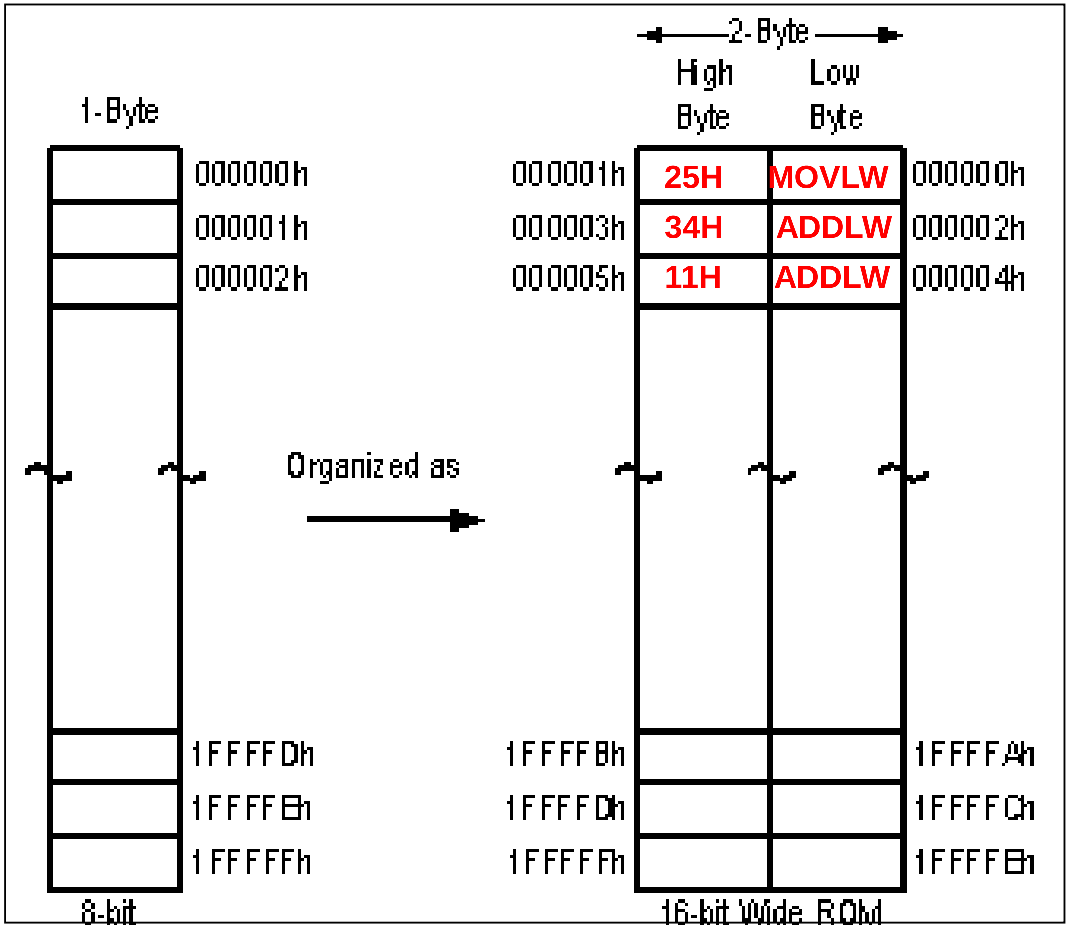
000008h HIGH PRIORITY INTERRUPT VECTOR

000018h LOW PRIORITY INTERRUPT VECTOR

ON-CHIP PROGRAM MEMORY

EXTERNAL/ UNIMPLEMENTED PROGRAM MEMORY (READ AS '0' IN MICROCONTROLLER MODE)

1FFFFFh



1-Byte

000000h
000001h
000002h
1FFFFDh
1FFFFEh
1FFFFFh

8-bit

Organized as

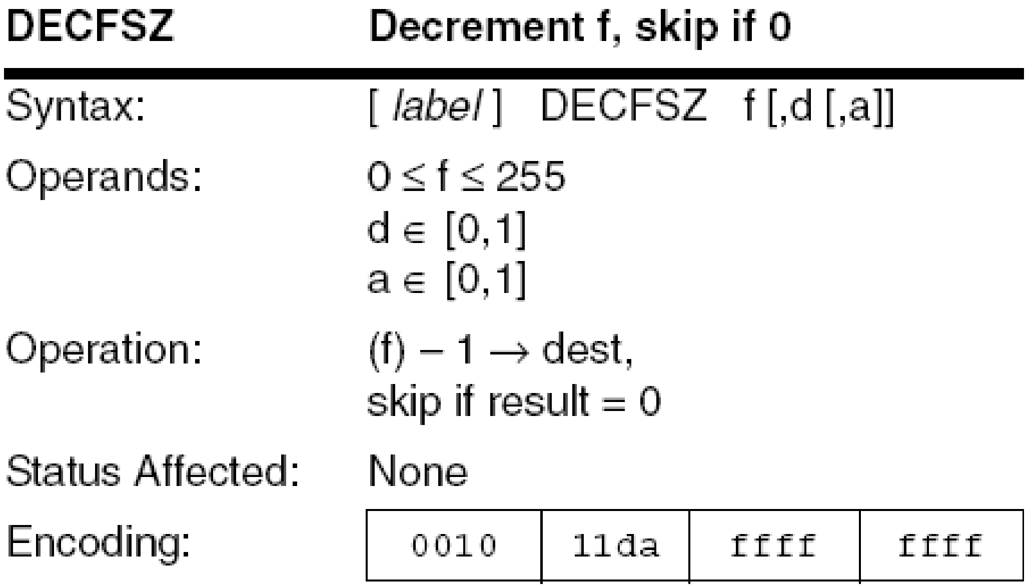
2-Byte

	High Byte	Low Byte	
000001h	25H	MOVLW	000000h
000003h	34H	ADDLW	000002h
000005h	11H	ADDLW	000004h
1FFFFBh			1FFFFAh
1FFFFDh			1FFFFCh
1FFFFFh			1FFFFEh

16-bit Wide ROM

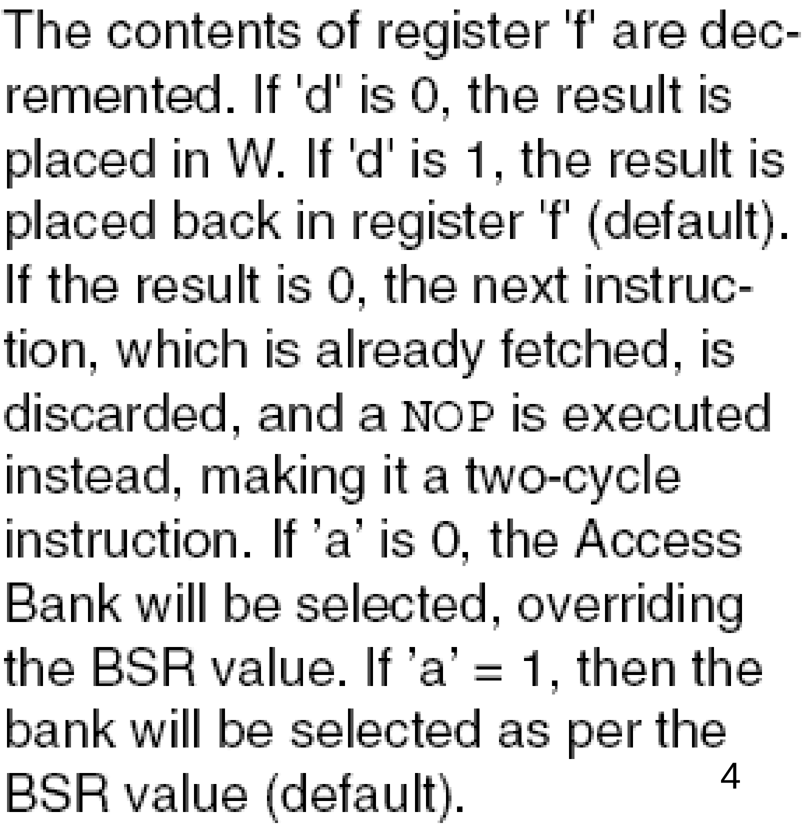
DECFSZ Instruction

DECFSZ fileReg, d ; Decrement fileReg and Skip next instruction if new value is 0
; if d==0 or d==w put new decremented value in WREG
; if d==1 or d==f put .... in fileReg



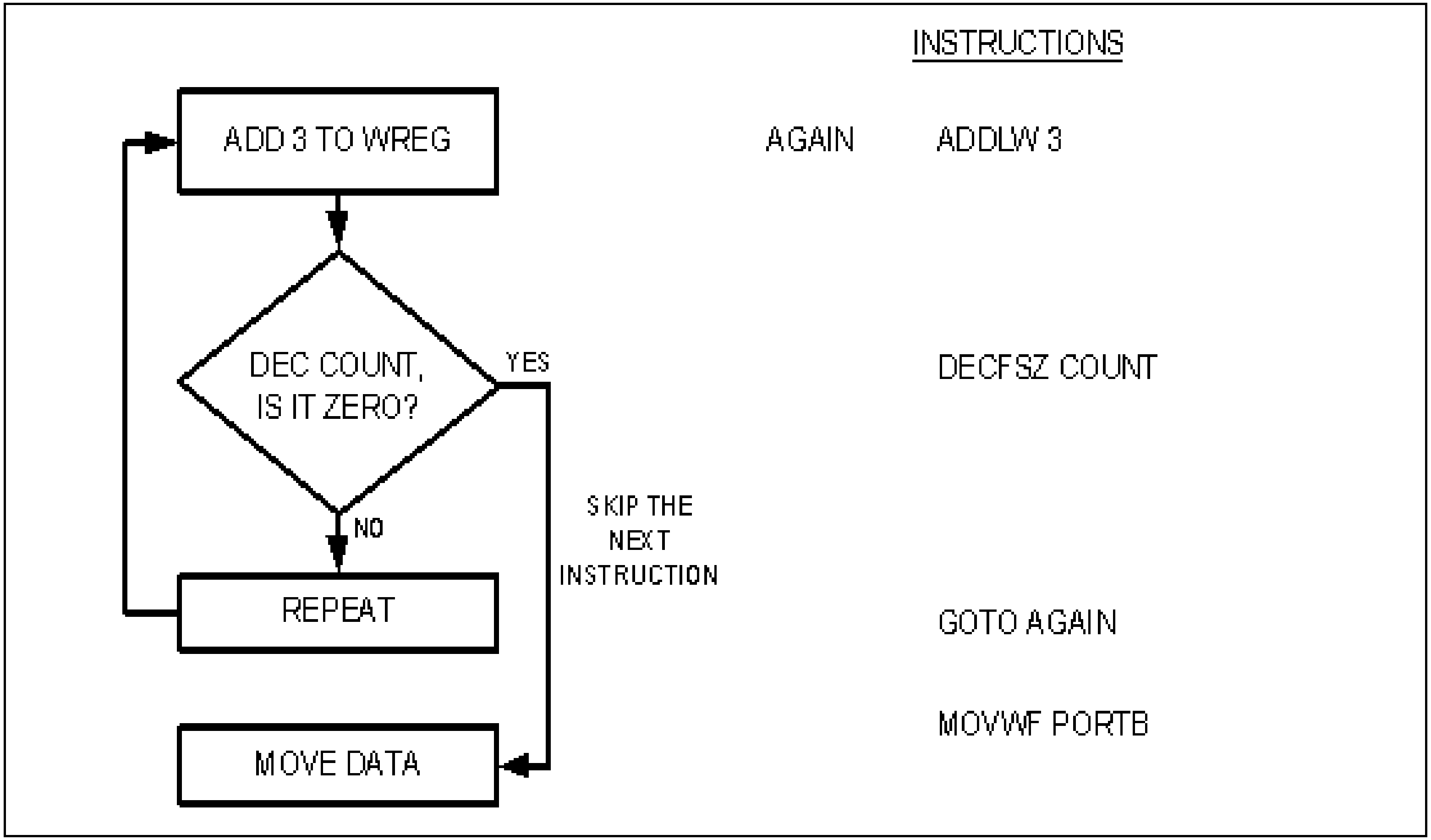
DECFSZ **Decrement f, skip if 0**

Syntax:	[*label*] DECFSZ f [,d [,a]]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – 1 → dest, skip if result = 0
Status Affected:	None
Encoding:	0010 11da ffff ffff



The contents of register 'f' are decremented. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). If the result is 0, the next instruction, which is already fetched, is discarded, and a NOP is executed instead, making it a two-cycle instruction. If ’a’ is 0, the Access Bank will be selected, overriding the BSR value. If ’a’ = 1, then the bank will be selected as per the BSR value (default).

DECFSZ Instruction



In executing this instruction, the specified fileReg is decremented, and if contents zero skips next instruction

DECFSZ Instruction

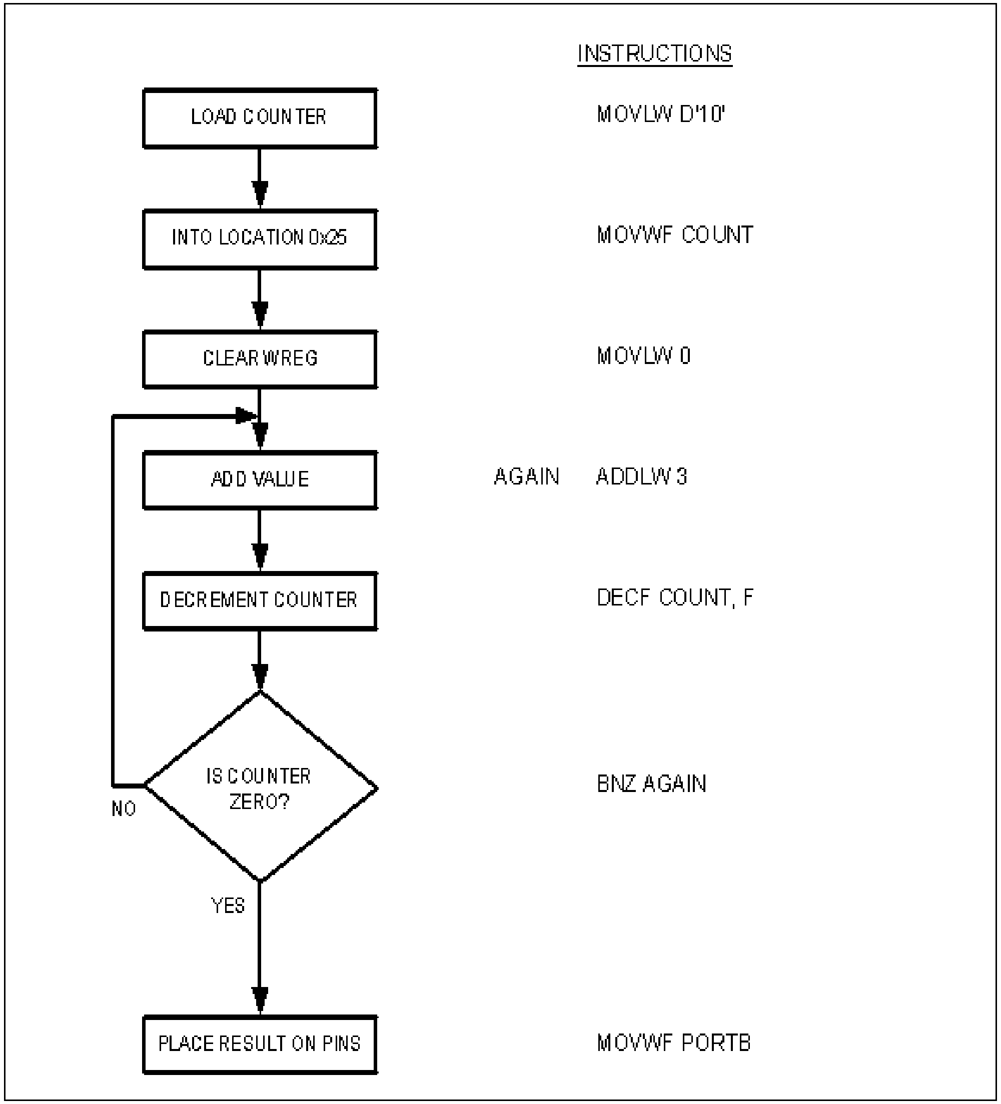
- Adds 3 to WREG 10 times

```
COUNT   EQU  0X25
             MOVLW      d'10'  ;10 → WREG
             MOVWF      COUNT
             MOVLW      0      ;0 → WREG
       AGAIN ADDLW      3
             DECFSZ     COUNT,F
             GOTO       AGAIN
             …..
```

Note: if use 'W' instead of 'F' infinite loop

Another way: Branch Non-Zero

BNZ n



INSTRUCTIONS

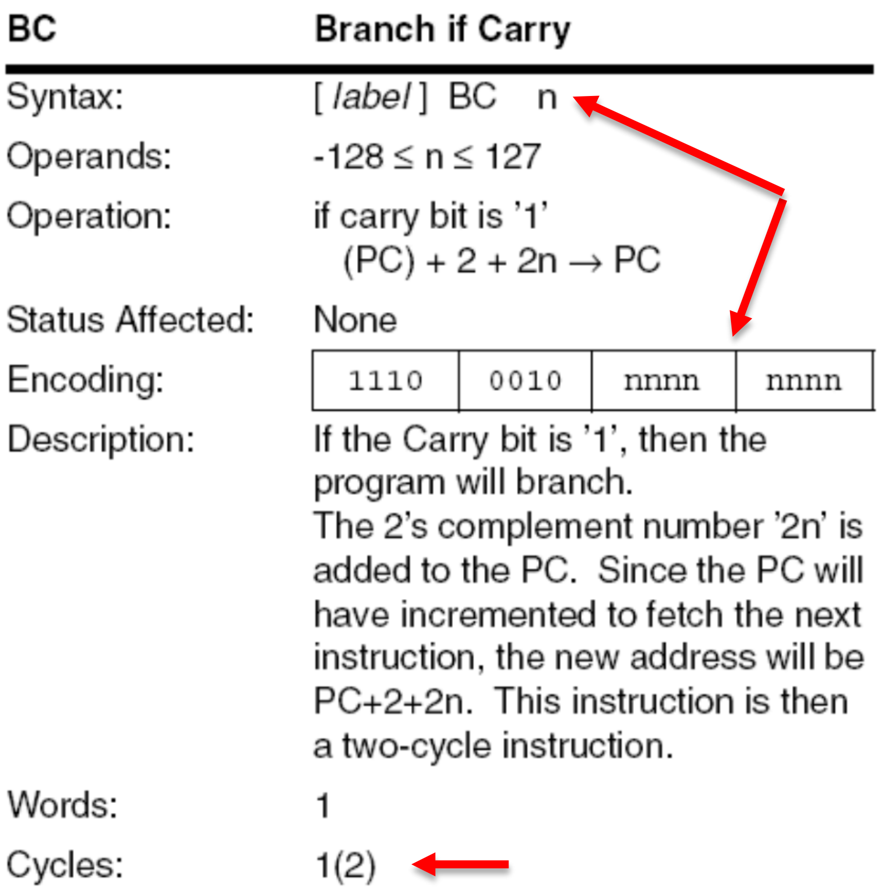
```
        MOVLW D'10'
        MOVWF COUNT
        MOVLW 0
AGAIN   ADDLW 3
        DECF COUNT, F
        BNZ AGAIN
        MOVWF PORTB
```

Nested Loops

- Loops inside loops
- Repeat an action more than 255 times (as restricted by a single byte counter)
- Branching is done based on the status register which reflects the last instruction (have to make sure instruction effects status register)
- Textbook Ex: pdf pg 118

Other Conditional Branch Instructions

BC	(C=1)
BNC	
BZ	(Z=1)
BNZ	
BN	(N=1)
BNN	
BOV	(OV=1)
BNOV	



BC **Branch if Carry**

Syntax: [*label*] BC n

Operands: $-128 \le n \le 127$

Operation: if carry bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0010	nnnn	nnnn

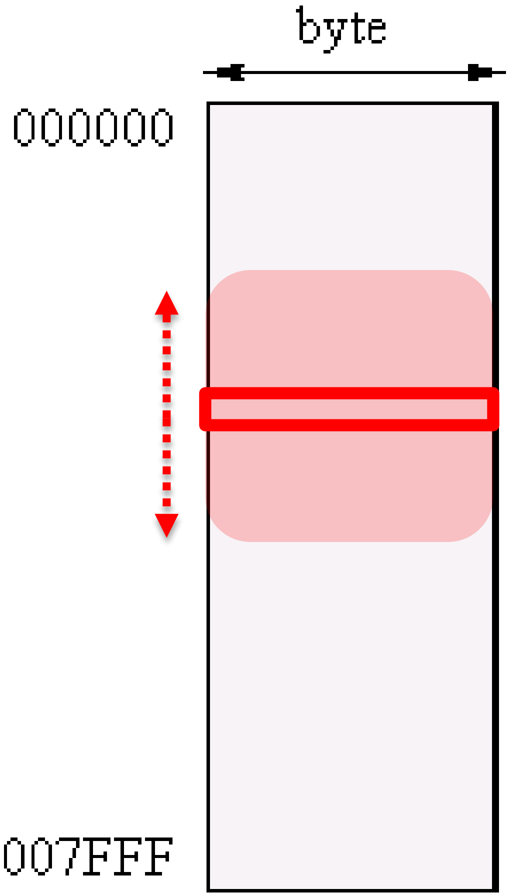
Description: If the Carry bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

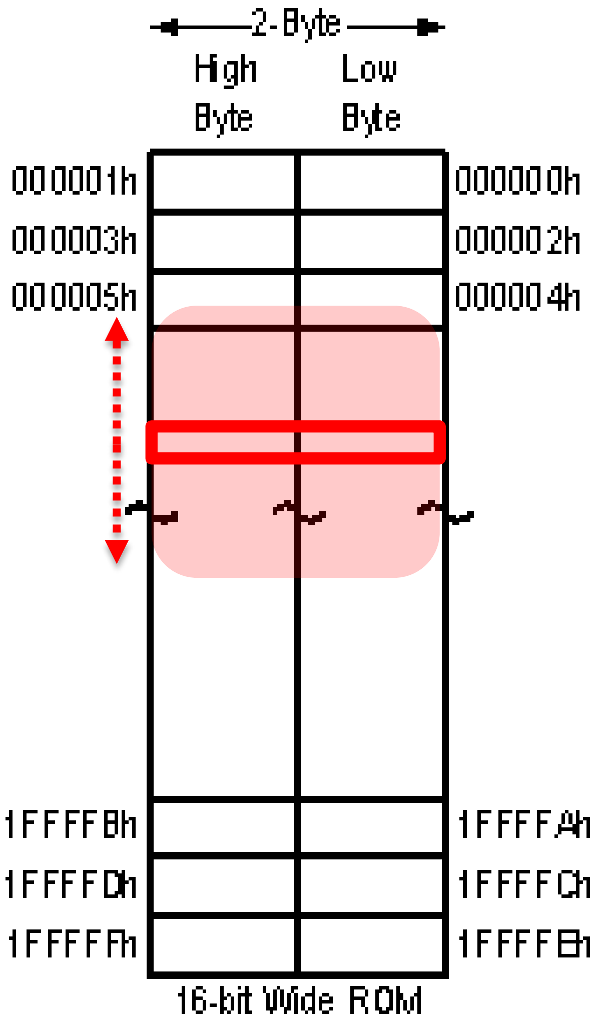
Words: 1

Cycles: 1(2)

Note: (All conditional branches are 2 bytes thus represent short jumps, within ~ +/-128 bits to PC)

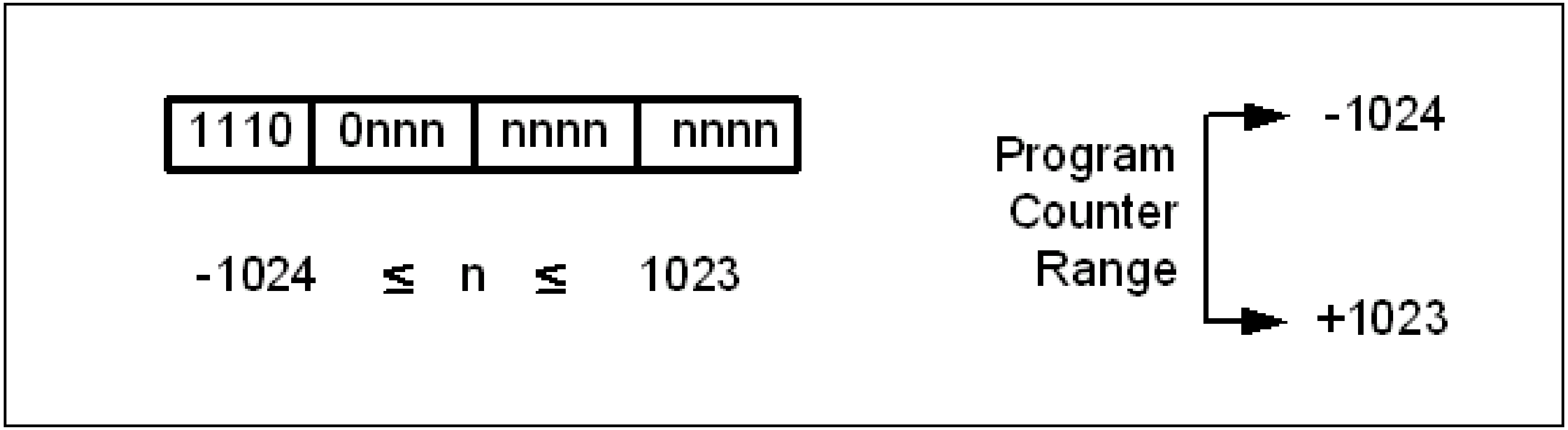
Jumping Range in ROM for Conditional Branches





BRA (Branch Unconditionally) Instruction Address Range

BRA n



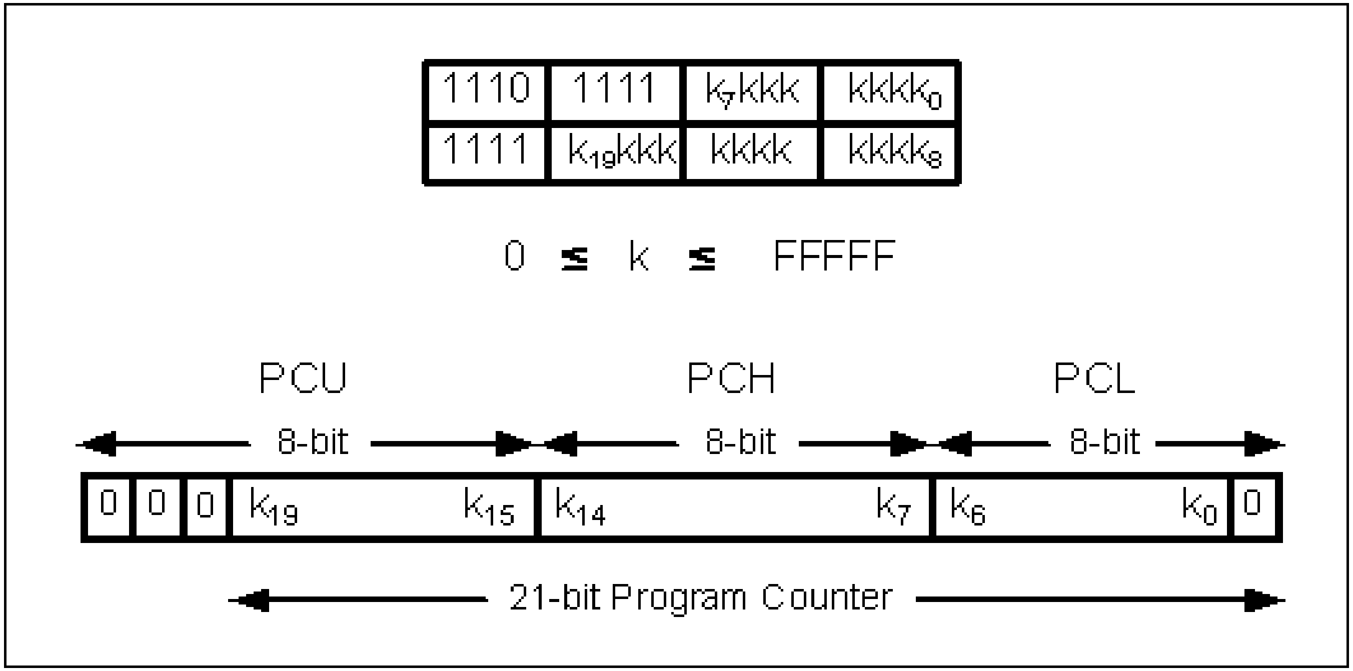
1110	0nnn	nnnn	nnnn

$-1024 \leq n \leq 1023$

GOTO Instruction

GOTO k

4 Byte Instruction

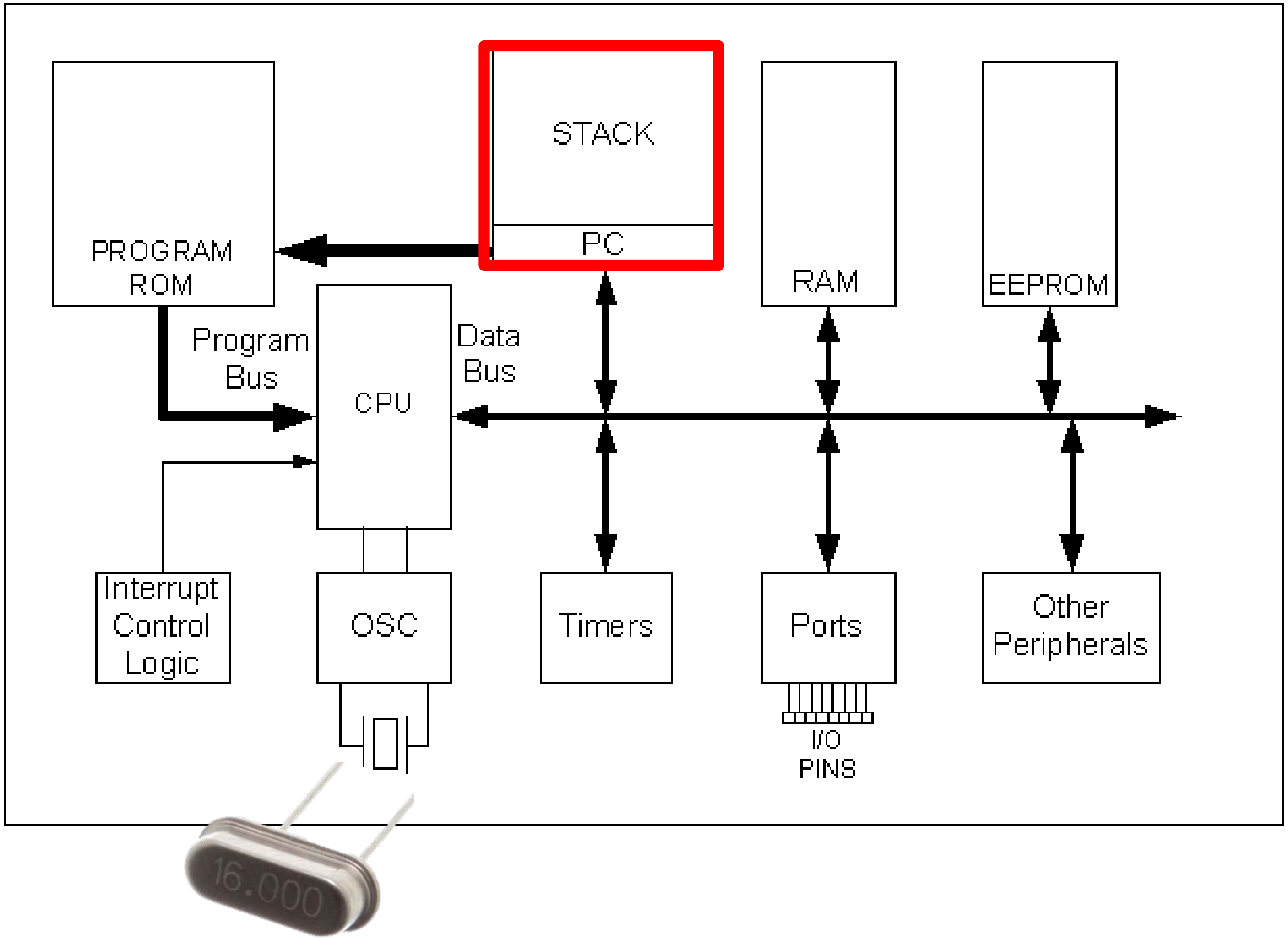


1110	1111	$k_7$kkk	kkkk_0
1111	k_{19}kkk	kkkk	kkkk_8

$0 \le k \le FFFFF$

0	0	0	k_{19} k_{15}	k_{14} k_7	k_6 k_0	0

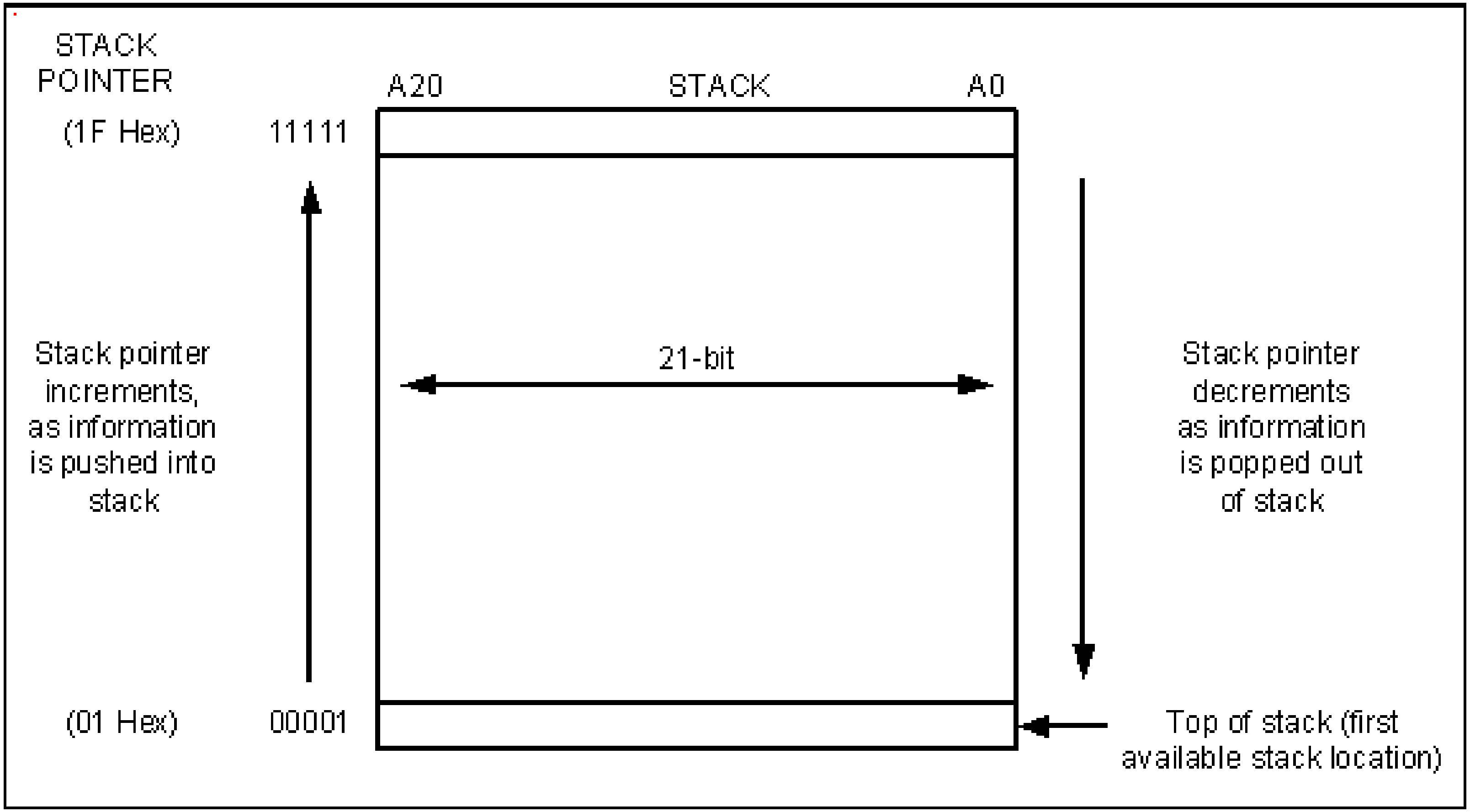
Stack


STACK
PC
PROGRAM ROM
RAM
EEPROM
Program Bus
CPU
Data Bus
Interrupt Control Logic
OSC
Timers
Ports
Other Peripherals
I/O PINS
16.000

Stack

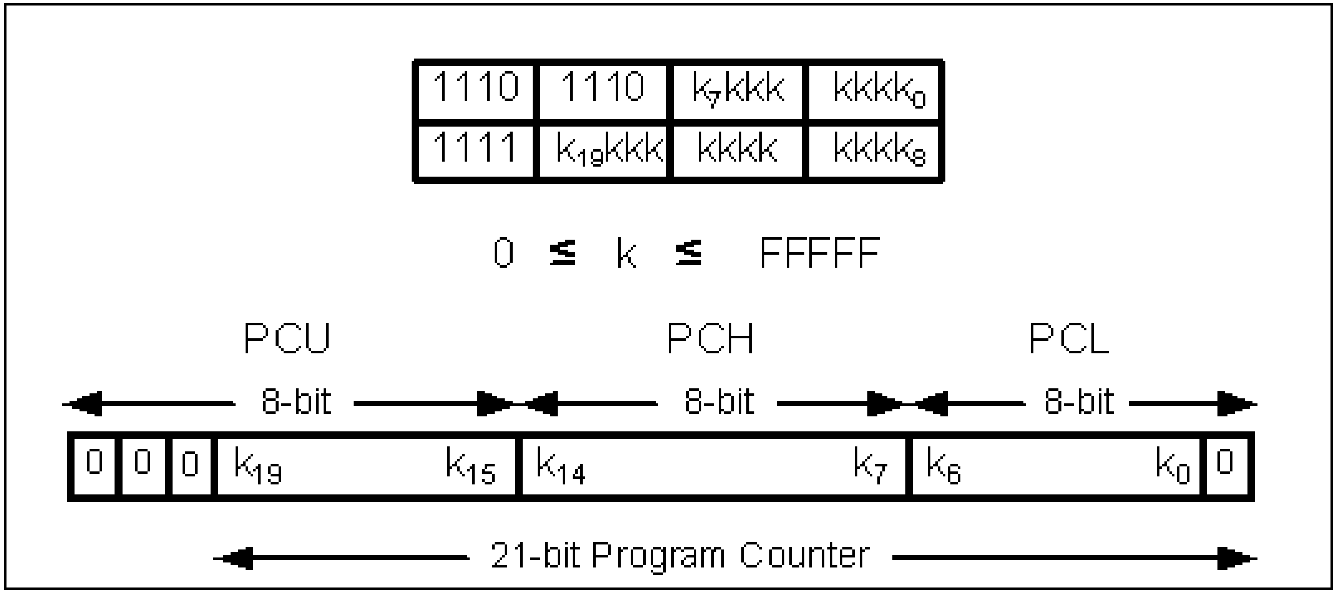
- Subroutines require stacks
- **CALL** and **RCALL** instructions can create subroutines (RETURN)
  - They are jumps but put the current PC onto the stack
- **Program Counter** needs to be stored so microcontroller knows where to return
- Stack thus has 21-bit words
  - Needs to be longer than one unit as there may be nested subroutines
  - Stack is separate RAM close to the CPU
- Separate 5-bit register (SP) for keeping track of stack (relative address)
  - SP is incremented from 0!
- User has to “stack” (store) other registers.

PIC Stack 31 × 21


STACK POINTER
(1F Hex)
11111
A20
STACK
A0
Stack pointer increments, as information is pushed into stack
21-bit
Stack pointer decrements as information is popped out of stack
(01 Hex)
00001
Top of stack (first available stack location)

CALL Instruction

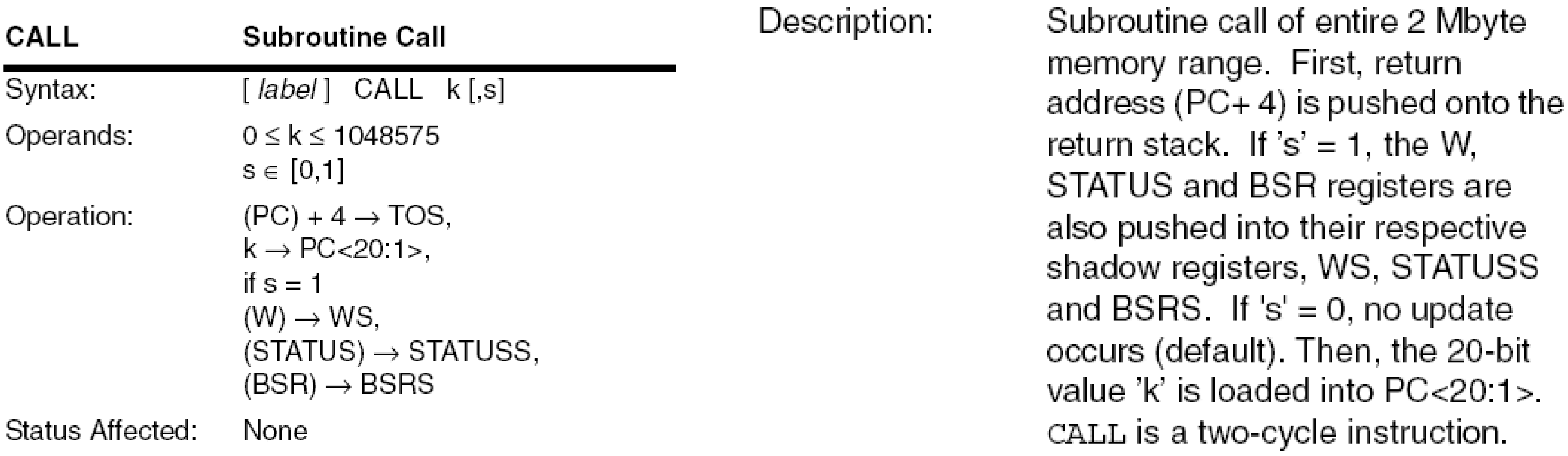
CALL k



1110	1110	$k_7$kkk	kkkk$_0$
1111	k_{19}kkk	kkkk	kkkk$_8$

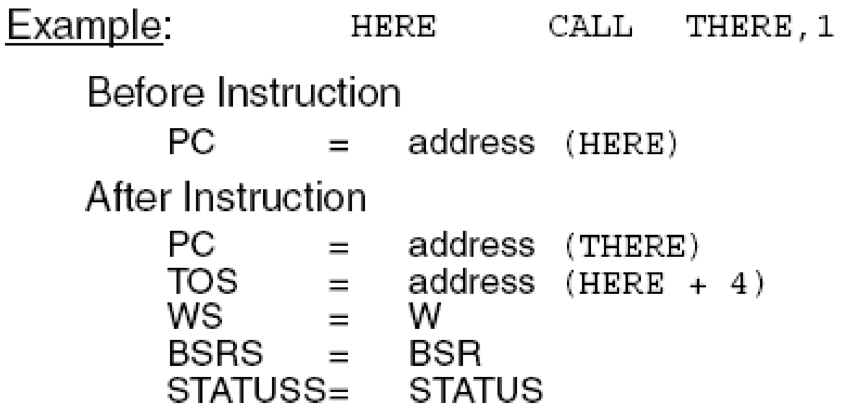
$0 \leq k \leq \text{FFFFF}$

CALL Instruction



CALL	**Subroutine Call**
Syntax:	[*label*] CALL k [,s]
Operands:	$0 \leq k \leq 1048575$ $s \in [0,1]$
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WS, (STATUS) → STATUSS, (BSR) → BSRS
Status Affected:	None

Description: Subroutine call of entire 2 Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If ’s’ = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value ’k’ is loaded into PC<20:1>. CALL is a two-cycle instruction.



Example: HERE CALL THERE,1

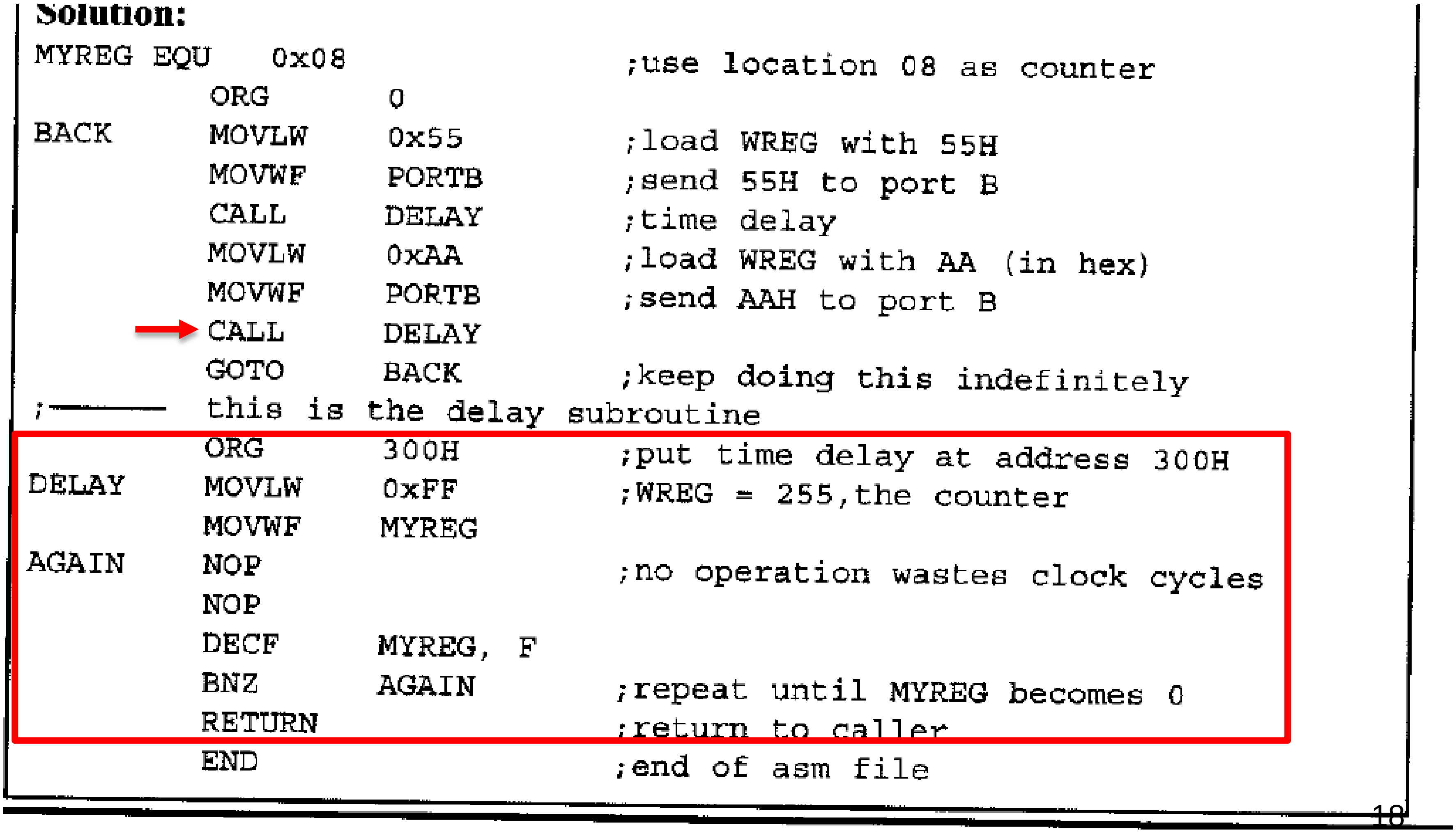
Before Instruction

PC	=	address (HERE)

After Instruction

PC	=	address (THERE)
TOS	=	address (HERE + 4)
WS	=	W
BSRS	=	BSR
STATUSS	=	STATUS

PIC Assembly Main Program That Calls Subroutine

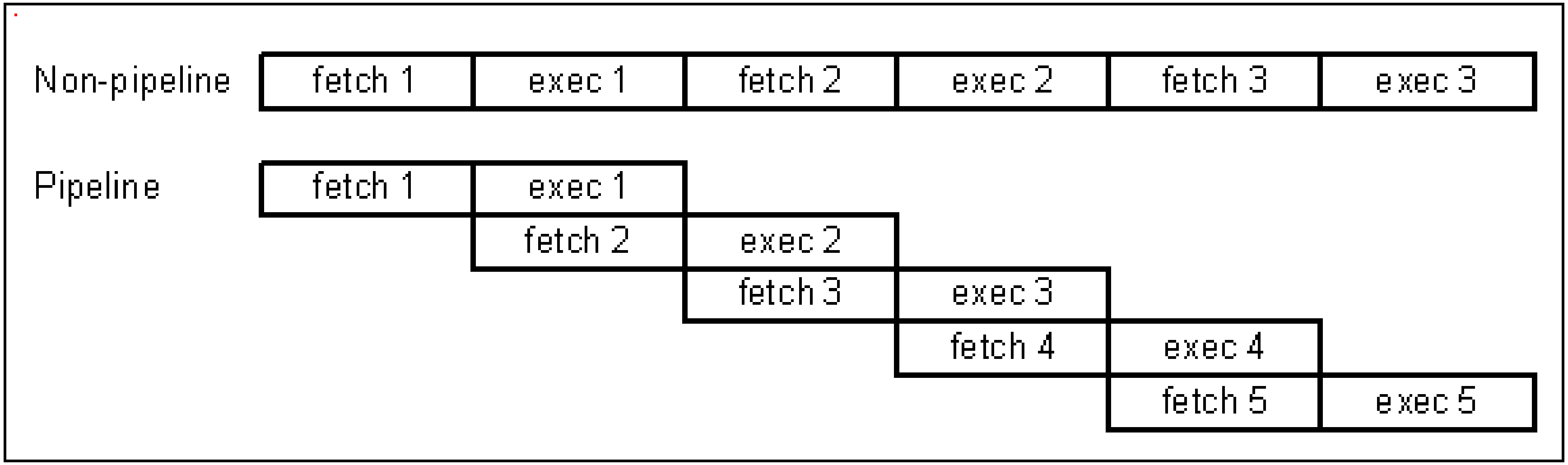


Solution:

```
MYREG EQU   0x08                    ;use location 08 as counter
        ORG       0
BACK    MOVLW     0x55              ;load WREG with 55H
        MOVWF     PORTB             ;send 55H to port B
        CALL      DELAY             ;time delay
        MOVLW     0xAA              ;load WREG with AA (in hex)
        MOVWF     PORTB             ;send AAH to port B
   →    CALL      DELAY
        GOTO      BACK              ;keep doing this indefinitely
;————   this is the delay subroutine
        ORG       300H              ;put time delay at address 300H
DELAY   MOVLW     0xFF              ;WREG = 255,the counter
        MOVWF     MYREG
AGAIN   NOP                         ;no operation wastes clock cycles
        NOP
        DECF      MYREG, F
        BNZ       AGAIN             ;repeat until MYREG becomes 0
        RETURN                      ;return to caller
        END                         ;end of asm file
```

Simple Pipeline vs. Non-pipeline

Most instructions take one or two cycles to execute



Non-pipeline	fetch 1	exec 1	fetch 2	exec 2	fetch 3	exec 3
Pipeline	fetch 1	exec 1				
		fetch 2	exec 2			
			fetch 3	exec 3		
				fetch 4	exec 4	
					fetch 5	exec 5

Branch Instructions can take more cycles to execute

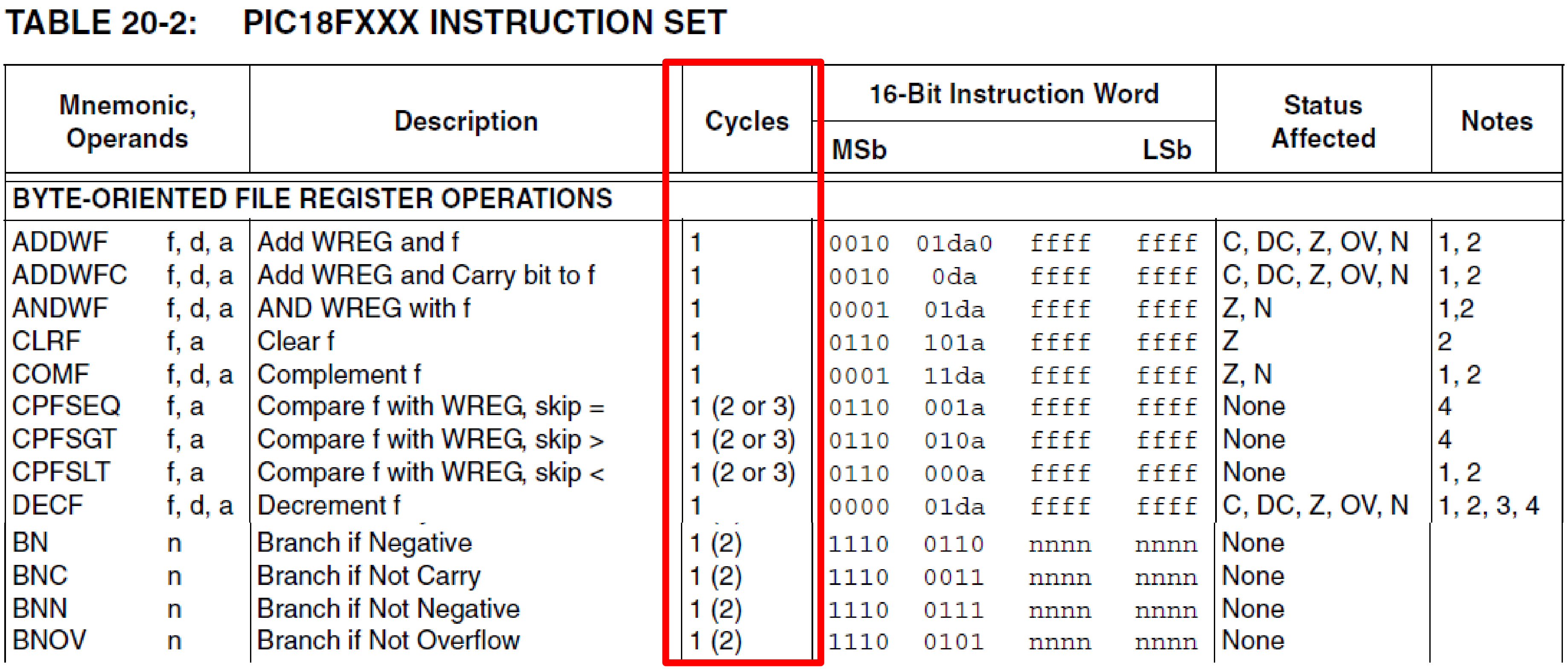
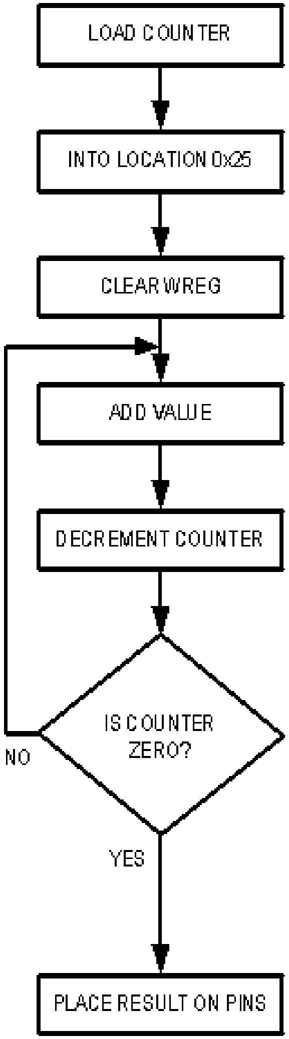


TABLE 20-2: PIC18FXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da0	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	0da	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
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	

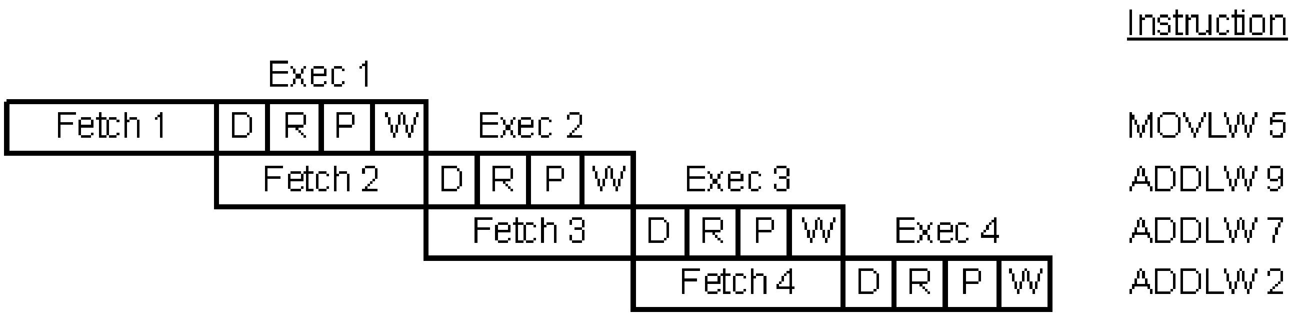
Branch Instructions can take more cycles to execute



```
            MOVLW D'10'
            MOVWF COUNT
            MOVLW 0
AGAIN       ADDLW 3
            DECF COUNT, F
            BNZ AGAIN
            MOVWF PORTB
```

What is in an Instruction Cycle

Q1	Q2	Q3	Q4
DECODE	READ	PROCESS	WRITE TO



D = Decode the instruction

R = Read the operand

P = Process (eg. ADDLW)

W = Write the result to destination register

Each micro operation takes one clock cycle, thus instruction rate is a quarter of the clock rate.

Questions?

- Textbook Ch. 3-1, 3-2 for Branching examples and more details
- Start reading Chapter 4
  - PIC I/O