EC-310 Microprocessor and Microcontroller Based Design

Chapter - 3

Time Delay Loop

Nazar Abbas Saqib
nazar.abbas@ceme.nust.edu.pk

Outline

- PIC18 Time Delay and Instruction Pipeline

PIC18 Time Delay and Instruction Pipeline

- **Delay Calculation for PIC18**
- The delays we have been using previously were not calculated delays. Now we shall be looking forward to the techniques of generating calculated delays.
- Two factors may influence the accuracy of time delay in PIC microcontrollers
  - The crystal frequency.
  - PIC Design
- The duration of the clock period for instruction cycle is a function of the crystal frequency.
- PIC design affects the time delay. Introduction of pipelining has reduced the number of instruction cycles to execute instruction.

PIC18 Time Delay and Instruction Pipeline

- **PIC Multistage Execution Pipeline**
- Speed of execution of instructions can be increased using superpipelining.
- In superpipelining, the execution of instructions is further split into many small steps that are all executed in parallel. In this way execution of many instructions is overlapped.
- Speed of execution in superpipelining is the speed of execution of the slowest stage of pipeline.
- In execution of instructions its important to keep intact the sequence of the instructions.

PIC18 Time Delay and Instruction Pipeline

- **Pipelining**
- In early microprocessors, the CPU could either fetch or execute at a given time.
- Pipelining allows the CPU to fetch and execute at the same time

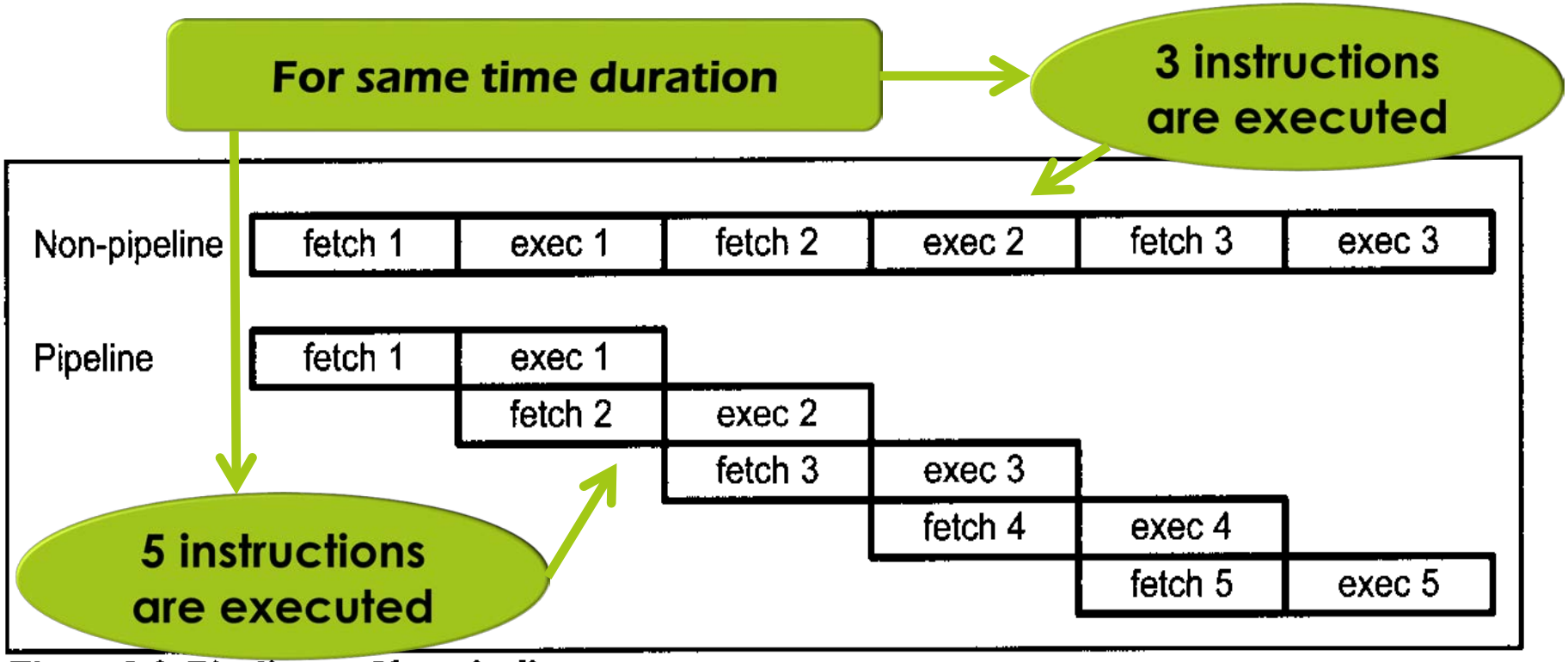

Figure 3-9. Pipeline vs. Non-pipeline

PIC18 Time Delay and Instruction Pipeline

- **PIC Multistage Execution Pipeline**
- The PIC superpipeline can be constructed as given in figure 3-11.

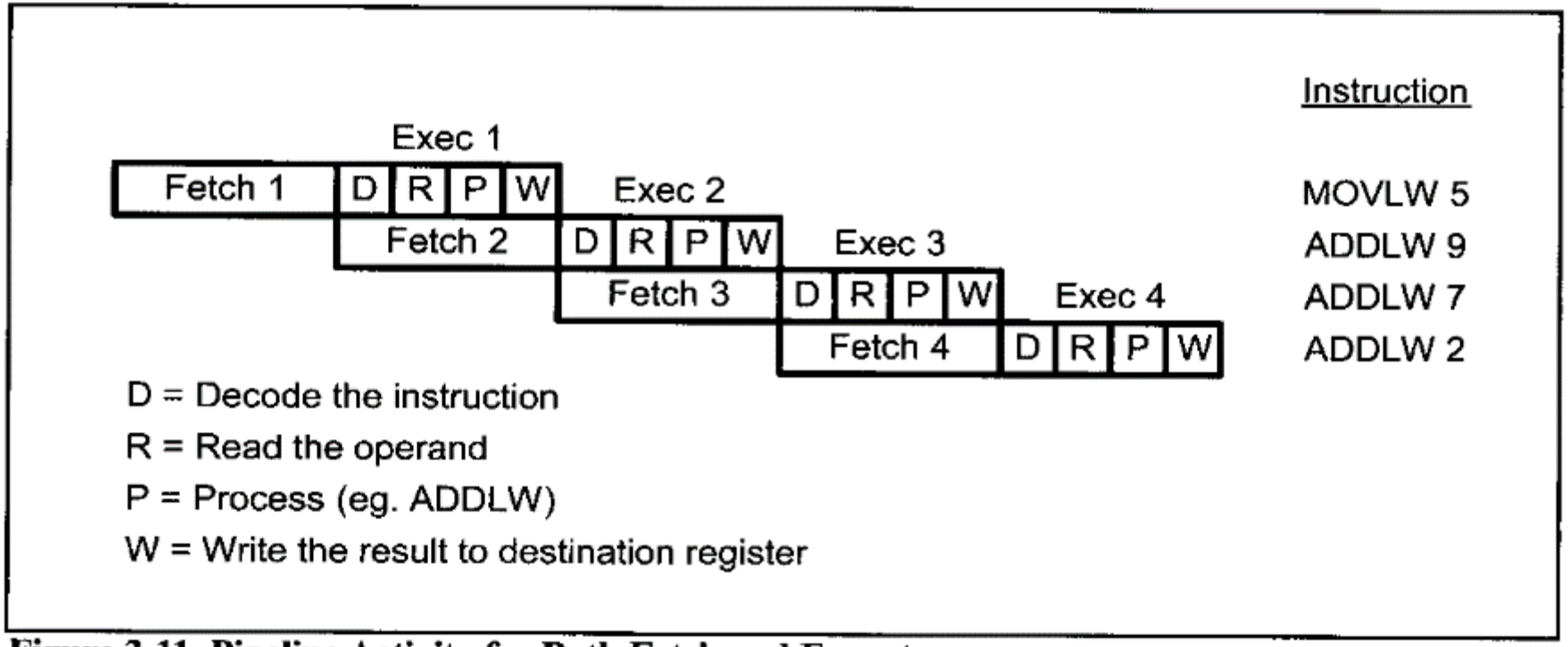

Figure 3-11. Pipeline Activity for Both Fetch and Execute

PIC18 Time Delay and Instruction Pipeline

- **PIC Multistage Execution Pipeline**

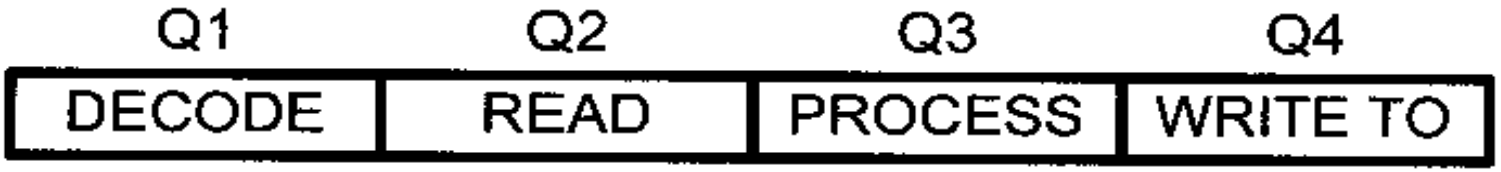

Figure 3-10: Pipeline Activity after the instruction has been fetched

- Figure 3-10 explains why we divide oscillator frequency by 4.
- In Q1 fetched instruction is decoded.
- In Q2 operand is fetched from file register.
- In Q3, the operation is performed (processing i-e execution)
- In Q4, the result is stored in destination register.

PIC18 Time Delay and Instruction Pipeline

- **Instruction Cycle time for the PIC**
- CPU takes certain amount of time to execute instructions. This time is known as **instruction cycle**.
- Most instructions in PIC take 2 or 3 instruction/machine cycles.
- Length of the instruction cycles depends upon the frequency of the crystal oscillator.
- In PIC18, one instruction cycle consist of four oscillator periods.
- Therefore to calculate the instruction cycles for PIC, we take ¼ of the oscillator frequency and then take its invers.

PIC18 Time Delay and Instruction Pipeline

- **Instruction Cycle time for the PIC**

Example 3-14

The following shows the crystal frequency for three different PIC-based systems. Find the period of the instruction cycle in each case.
(a) 4 MHz (b) 16 MHz (c) 20 MHz

Solution:
(a) 4/4 = 1 MHz; instruction cycle is 1/1 MHz = 1 µs (microsecond)
(b) 16 MHz/4 = 4 MHz; instruction cycle = 1/4 MHz = 0.25 µs = 250 ns (nanosecond)
(c) 20 MHz/4 = 5 MHz; instruction cycle = 1/5 MHz = 0.2 µs = 200 ns

Instruction Cycles

PIC18 has 77 instructions (4 -32bit & rest all of16bit)

Mnemonic, Operand		Description	Cycles	Status Affected
Byte-Oriented File Register Operands				
ADDWF	f, d, a	Add WREG and f	1	C, DC, Z, OV, N
ADDWFC	f, d, a	Add WREG and carry bit to f	1	C, DC, Z, OV, N
ANDWF	f, d, a	AND WREG with f	1	Z, N
CLRF	f, a	Clear f	1	Z
COMF	f, d, a	Complement f	1	Z, N
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	None
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	None
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	None
DECF	f, d, a	Decrement f	1	C, DC, Z, OV, N
DECFSZ	f, d, a	Decrement f, skip if 0	1 (2 or 3)	None
DCFSNZ	f, d, a	Decrement f, skip if not 0	1 (2 or 3)	None
INCF	f, d, a	Increment f	1	C, DC, Z, OV, N
INCFSZ	f, d, a	Increment f, skip if 0	1 (2 or 3)	None
INFSNZ	f, d, a	Increment f, skip if not 0	1 (2 or 3)	None
IORWF	f, d, a	Inclusive OR WREG with f	1	Z, N
MOVF	f, d, a	Move f	1	Z, N
MOVFF	fs, fd	Move fs to fd	2	None
MOVWF	f, a	Move WREG to f	1	None
MULWF	f, a	Multiply WREG with f	1	None
NEGF	f, a	Negate f	1	C, DC, Z, OV, N
RLCF	f, d, a	Rotate left f through carry	1	C, Z, N
RLNCF	f, d, a	Rotate left f (no carry)	1	Z, N
RRCF	f, d, a	Rotate right f through carry	1	C, Z, N
RRNCF	f, d, a	Rotate right f (no carry)	1	Z, N
SETF	f, a	Set f	1	None
SUBFWB	f, d, a	Subtract from WREG with borrow	1	C, DC, Z, OV, N
SUBWF	f, d, a	Subtract WREG from f	1	C, DC, Z, OV, N
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	C, DC, Z, OV, N
SWAPF	f, d, a	Swap nibbles in f	1	None
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	None
XORWF	f, d, a	Exclusive OR WREG with f	1	Z, N

Instruction Cycles

Bit-Oriented File Register Operands				
BCF	f, b, a	Bit clear f	1	None
BSF	f, b, a	Bit set f	1	None
BTFSC	f, b, a	Bit test f, skip if clear	1 (2 or 3)	None
BTFSS	f, b, a	Bit test f, skip if set	1 (2 or 3)	None
BTG	f, d, a	Bit toggle	1	None

Control Operations				
BC	n	Branch if carry	1 (2)	None
BN	n	Branch if negative	1 (2)	None
BNC	n	Branch if not carry	1 (2)	None
BNN	n	Branch if not negative	1 (2)	None
BNOV	n	Branch if not overflow	1 (2)	None
BNZ	n	Branch if not zero	2	None
BOV	n	Branch if overflow	1 (2)	None
BRA	n	Branch unconditionally	1 (2)	None
BZ	n	Branch if zero	1 (2)	None
CALL	n, s	Call subroutine	2	NOne
CLRWDT		Clear watchdog timer	1	$\overline{TO}$, $\overline{PD}$
DAW		Decimal adjust WREG	1	C
GOTO	n	Goto	2	None
NOP		No operation	1	None
NOP		No operation	1	None
POP		Pop top of return address	1	None
PUSH		Push top of return address	1	None
RCALL	n	Relative call	2	None
RESET		Software device RESET	1	All
RETFIE	s	Return from interrupt enable	2	GIE/GIEH,PEIE
RETLW	k	Return with literal in WREG	2	None
RETURN	s	Return from subroutine	2	None
SLEEP		Go into standby mode	1	$\overline{TO}$, $\overline{PD}$

Instruction Cycles

Literal Operations				
ADDLW	k	Add literal and WREG	1	C, DC, Z, OV, N
ANDLW	k	AND literal with WREG	1	Z, N
IORLW	k	Inclusive OR literal with WREG	1	Z, N
LFSR	f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	None
MOVLB	k	Move literal to BSR<3:0>	1	None
MOVLW	k	Move literal to WREG	1	None
MULLW	k	Multiply literal with WREG	1	None
RETLW	k	Return with literal in WREG	2	None
SUBLW	k	Subtract WREG from literal	1	C, DC, Z, OV, N
XORLW	k	Exclusive OR literal with WREG	1	Z, N
Data memory to and from Program memory Operations				
TBLRD*		Table read	2	None
TBLRD*+		Table read with post-increment		None
TBLRD*–		Table read with post-decrement		None
TBLRD+*		Table read with pre-increment		None
TBLWT*		Table write	2 (5)	None
TBLWT*+		Table write with post-increment		None
TBLWT*–		Table write with post-decrement		None
TBLWT+*		Table write with pre-increment		None

Example 3-15

For a PIC18 system of 4 MHz, find how long it takes to execute each of the following instructions:

```
(a) MOVLW      (b) DECF       (c) MOVWF
(d) ADDLW      (e) NOP        (f) GOTO
(g) CALL       (h) BNZ
```

Solution:

The machine cycle for a system of 4 MHz is 1 μs, as shown in Example 3-14. Appendix A shows instruction cycles for each of the above instructions. Therefore, we have:

Instruction	*Instruction cycles*	*Time to execute*
(a) MOVLW 0x55	1	1 × 1 μs = 1 μs
(b) DECF MYREG	1	1 × 1 μs = 1 μs
(c) MOVWF	1	1 × 1 μs = 1 μs
(d) ADDLW	1	1 × 1 μs = 1 μs
(e) NOP	1	1 × 1 μs = 1 μs
(f) GOTO	2	2 × 1 μs = 2 μs
(g) CALL	2	2 × 1 μs = 2 μs
(h) BNZ	2/1	(2 μs taken, 1 μs if it falls through)

Example 3-16

Find the size of the delay of the code snippet below if the crystal frequency is 4 MHz:

Solution:

From Appendix A, we have the following machine cycles for each instruction of the DELAY subroutine:

```
                                        Instruction Cycle
MYREG EQU   0x08          ;use location 08 as counter

DELAY     MOVLW     0xFF                  1
          MOVWF     MYREG                 1

AGAIN     NOP                             1
          NOP                             1
          DECF      MYREG,F               1
          BNZ       AGAIN                 2

          RETURN                          1
```

F = 4Mhz / 4 = 1MHz
T = 1/f = 1/1MHz = 1uS

Therefore, we have a time delay of $[(255 \times 5) + 1 + 1 + 1] \times 1\ \mu s = 1278\ \mu s$.

Notice that BNZ takes two instruction cycles if it jumps back, and takes only one when falling through the loop. That means the above number should be 1277 µs.

Example 3-18

Find the size of the delay in the following program if the crystal frequency is 4 MHz:

```
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     0xFA        ;WREG = 250, the counter
          MOVWF     MYREG
AGAIN     NOP                   ;no operation wastes clock cycles
          NOP
          NOP
          DECF      MYREG, F
          BNZ       AGAIN       ;repeat until MYREG becomes 0
          RETURN                ;return to caller
          END                   ;end of asm file
```

Solution on next slide .....

Solution:

From Appendix A, we have the following machine cycles for each instruction of the DELAY subroutine:

```
                                        Instruction Cycle

DELAY   MOVLW   0xFA                        1
        MOVWF   MYREG                       1

AGAIN   NOP                                 1
        NOP                                 1
        NOP                                 1
        DECF    MYREG, F                    1
        BNZ     AGAIN                       2

        RETURN                              1
```

Therefore, we have a time delay of $[(250 \times 6) + 1 + 1 + 1] \times 1\ \mu s = 1503\ \mu s$.

Example 3-18

For a instruction cycle of 1 μs, find the time delay in the following subroutine:

```
R2      EQU     0x7
R3      EQU     0x8
DELAY                                   Instruction Cycle
                MOVLW       D'200'      1
                MOVWF       R2          1
AGAIN           MOVLW       D'250'      1
                MOVWF       R3          1
HERE            NOP                     1
                NOP                     1
                DECF        R3, F       1
                BNZ         HERE        2
                DECF        R2, F       1
                BNZ         AGAIN       2
                RETURN                  1
```

Beginning and end of the AGAIN loop add 5x200x1us=1000us to the time delay.

Solution:

For the HERE loop, we have (5 × 250) 1 μs = 1250 μs. The AGAIN loop repeats the HERE loop 200 times; therefore, we have 200 × 1250 μs = 250000 μs, if we do not include the overhead. However, the following instructions of the outer loop add to the delay:

```
AGAIN           MOVLW       D'250'      1
                MOVWF       R3          1
                .....
                DECF        R2, F       1
                BNZ         AGAIN       2
```

We should also subtract 200us for the times BNZ HERE falls through. As a result the delay time is 250000+1000-200=250800 us=250.8 mS

Example 3-19

Find the time delay for the following subroutine, assuming a crystal frequency of 4 MHz. Discuss the disadvantage of this over Example 3-18.

```
MYREG EQU   0x8
                                        Machine Cycle

DELAY   MOVLW   D'200'          1
        MOVWF   MYREG           1

AGAIN   NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        NOP                     1
        DECF    MYREG, F        1
        BNZ     AGAIN           2

        RETURN                  1
```

The time delay inside the AGAIN loop is [200(13+2)]x1us=3000us NOP is a 2-byte instructions in the DELAY program and all the instructions are 2-byte instruction. Thus the loop delay takes 34 bytes of ROM code space and gives us only a 3000us delay.

Example 3-20

Write a program to toggle all the bits of SFR PORTB every 1 s. Assume that the crystal frequency is 10 MHz and the system is using a PIC18F458.

Solution:

```
;tested using MPLAB with PIC18F458 operating at 10 MHz
R2      EQU     0x2
R3      EQU     0x3
R4      EQU     0x4


            MOVLW     0x55          ;load WREG with 55H
            MOVWF     PORTB         ;send 55H to PORTB B
BACK        CALL      DELAY_500MS   ;time delay
            COMF      PORTB         ;complement PORTB
            GOTO      BACK          ;keep doing this indefinitely

;-------  this is the delay subroutine
DELAY_500MSEC
                MOVLW       D'20'
                MOVWF       R4
BACK            MOVLW       D'100'
                MOVWF       R3
AGAIN           MOVLW       D'250'
                MOVWF       R2
HERE            NOP
                NOP
                DECF        R2, F
                BNZ         HERE
                DECF        R3, F
                BNZ         AGAIN
                DECF        R4, F
                BNZ         BACK
                RETURN
```

FOR PRACTICE

Delay $20 \times 100 \times 250 \times 5 \times 400$ ns $= 1{,}000{,}000{,}000$ ns $= 1{,}000{,}000$ µs $= 1$ s.

In this calculation, we have not included the overhead associated with the two outer loops. Use the MPLAB simulator to verify the delay.

PIC18 Time Delay and Instruction Pipeline

- Branch Penalty
- For pipelining to work, we need a buffer or a queue into which instruction is prefetched and ready for execution.
- In some circumstances, the CPU needs to flush out the queue.
- Example: When branch instruction needs to be executed. In this, the CPU starts to fetch code from the new target location and code already in the queue has to be discarded.
- In this, the execution unit has to wait until the fetch unit fetches new instruction. This is known as **branch penalty**.
- Penalty is an extra instruction cycle to fetch new instruction.

PIC18 Time Delay and Instruction Pipeline

- **Branch Penalty**
- In this way, few instructions take more than 2 instruction cycles. These instructions include GOTO, BRA, CALL and all conditional branch instructions.
- If conditional branch instruction doesn't jumps, then in that case it only takes one instruction cycle.