

1. Describe what an instruction cycle is.

Instruction Fetch:

- 1) The CPU places the content of the program counter on the address bus in order to fetch an instruction from memory,
- 2) The program counter is incremented so as to be ready to fetch the next instruction,
- 3) After the data in the data bus is ready, the CPU places it (the instruction) in the instruction register ready for decoding,

Instruction Decode:

The instruction decoder decodes the instruction to identify:

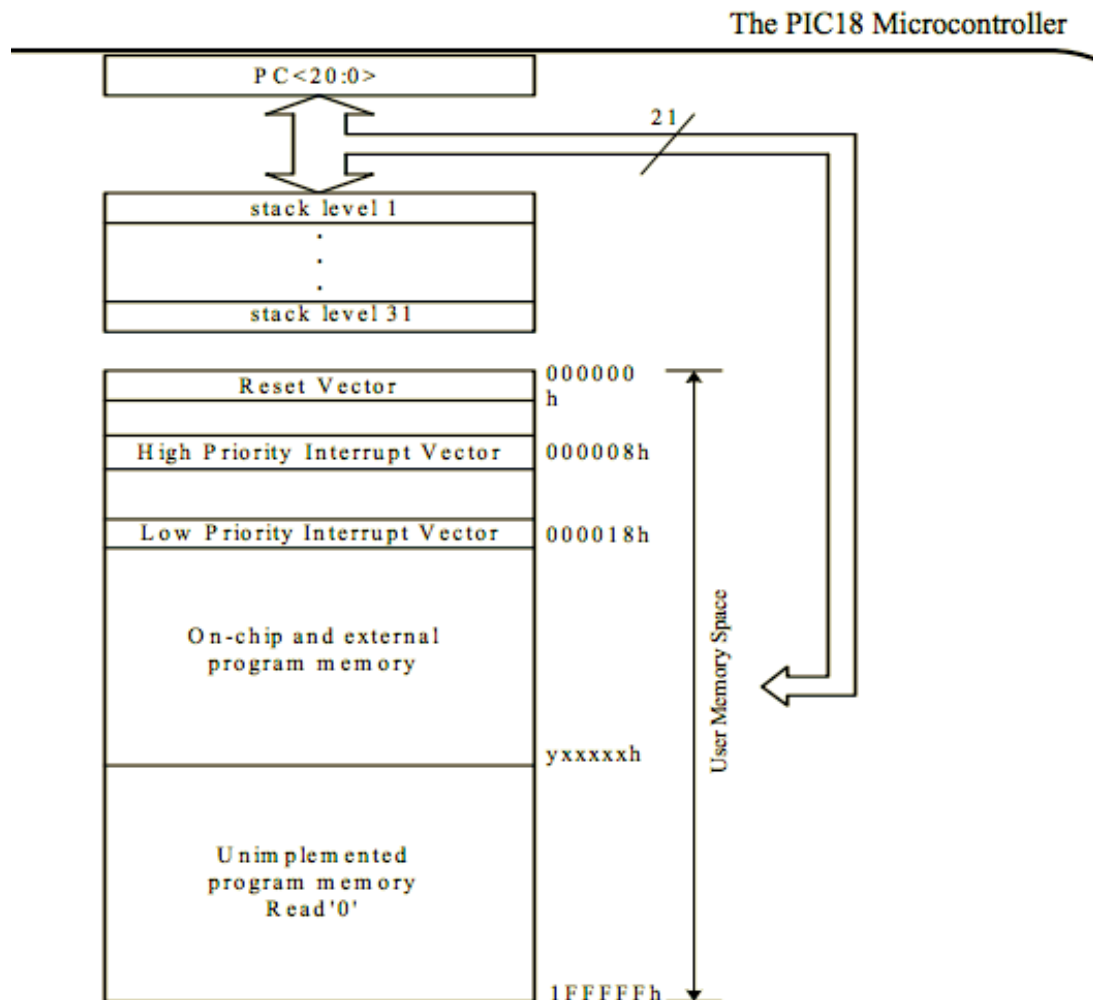
- i) the operation to be carried out,
- ii) the address of the source operands and the results,

Instruction Execution:

- 1) Additional memory accesses may be conducted to obtain all the operands needed (if any),
- 2) The specified operation will be performed,
- 3) The results are placed according to the destination address so determined. Again, additional memory access may be required (if any),

2. A programmer puts the first instruction at address 100H. What happens when the microcontroller is powered up?
PC=0, MPU fetch the first instruction from 100H and then executes it .

3. Show the lowest and highest values (in hex) that the PIC18 program counter can take.
000000H - 1F FFFFH



Note, y can be 0 or 1 whereas x can be 0-F

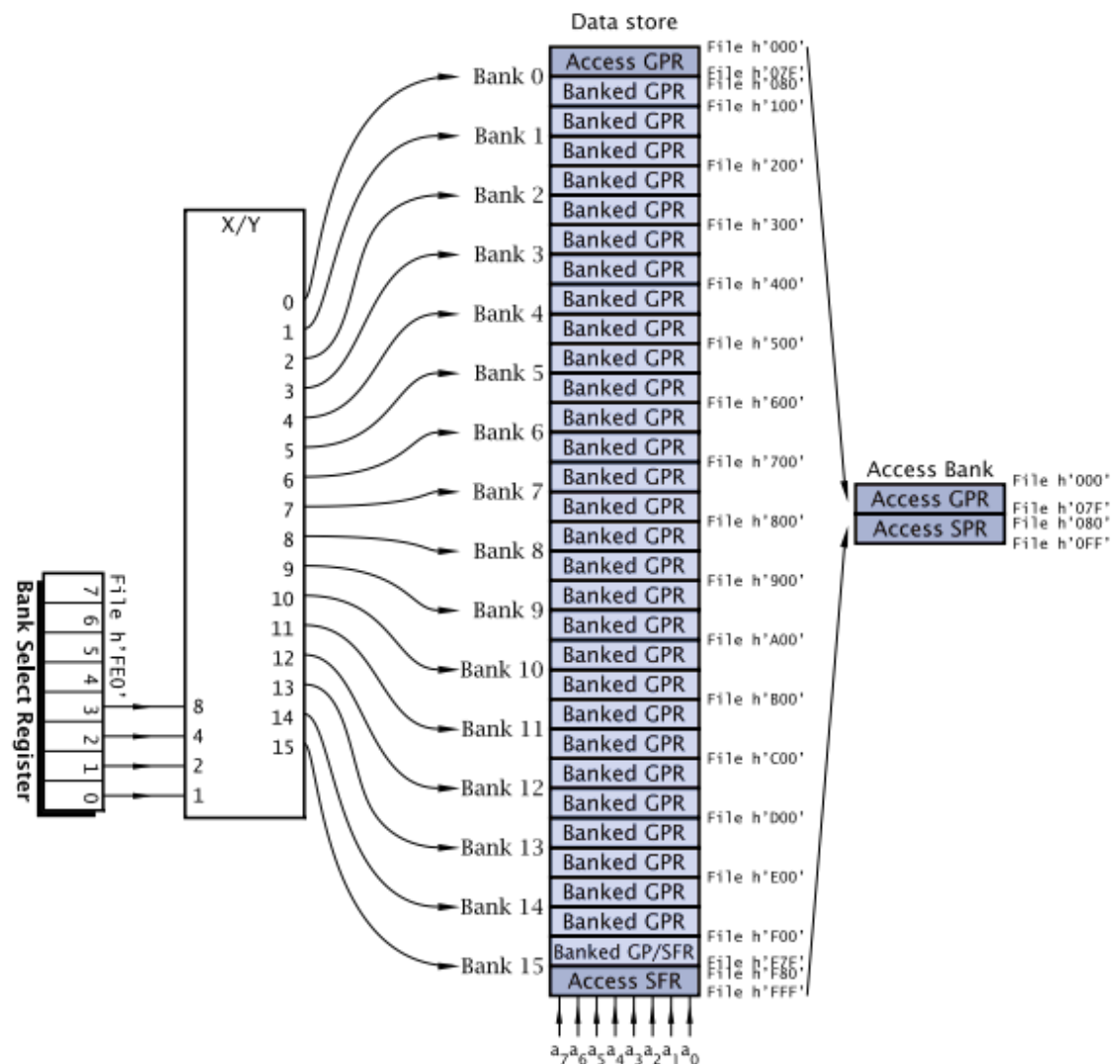
Figure 1.5 PIC18 Program memory Organization (redraw with permission of

4. Most instructions allocate only 8 bits to address the data memory, whereas the width of the address bus of PIC18 data memory is more than 8 bits. How can these instructions uniquely specify where operands and destination of an arithmetic/logic operation are located in the memory?

/Instruction: 16 bit

Opcode 8 bit, Operand 8 bit

Use of data bank



5. What is result of the following code and where is it kept

MOVLW 25H

ADDLW 0x1F

MOVLW 25H: WREG=25H

ADDLW 0x1F: WREG=25+1F=44

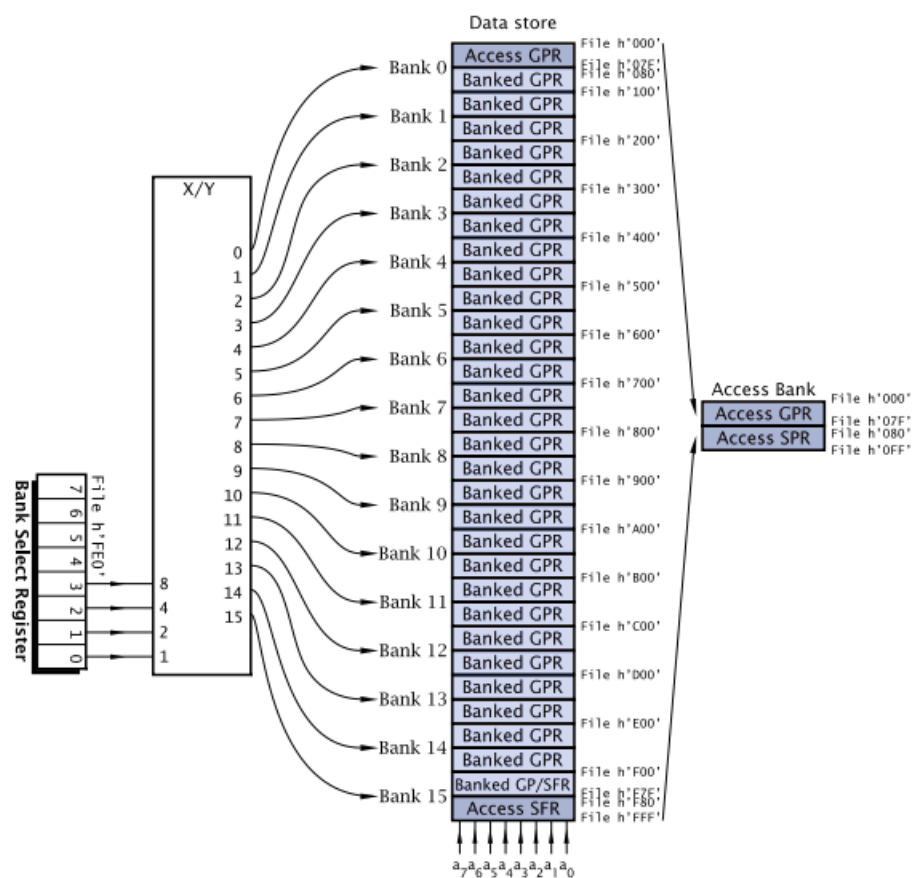
6. What is result of the following code and where is it kept

```
MOVLW 15H
ADDLW 0xEA
```

MOVLW 15H : WREG=15H
ADDLW 0xEA : WREG=15+EA=FFH

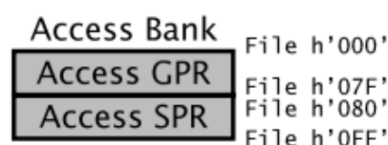
7. What are types of registers consist the file Register RAM? What are they called, and their corresponding address range?

RAM: 2K
GRF and SFR



8. What is an access bank?

By default, if not specified the data bank, the data file registers are called access bank
Consisting of GRF: 00H – 7FH (128 byte) and SFR: 80H – FFH (128 byte)



9. Show a simple code to load values 30H and 97H into locations 0x05h and 0x06h, respectively.

```
MOVLW 30H
MOVWF 05H
```

```
MOVLW 97H
MOVWF 06H
```

10. Show a simple code to load value 15H into location 7, and then add to WREG five times and place the result in WREG as the values are added. WREG should be zero before the addition starts.

```
MOVLW    0x15    ; load 0x15 into WREG
MOVWF    0x07    ; move data from WREG -> loc 0x07
MOVLW    0x00    ; clear WREG
ADDWF    7H, W    ; WREG + 'loc 7H' -> WREG
ADDWF    7H, W
ADDWF    7H, W
ADDWF    7H, W
ADDWF    7H, W
```

11. Repeat Q10 with BNZ loop.

```
COUNTREG EQU    0x20
LOC        EQU    0x07

MOVLW    0x15    ; WREG = 0x15
MOVWF    LOC    ; WREG -> loc LOC
MOVLW    0x05    ; WREG = 0x05
MOVWF    COUNTREG ; WREG -> loc COUNTREG
MOVLW    0x00    ; clear WREG

ADDLOOP  ADDWF    LOC, W    ; (WREG) + (LOC) -> WREG
          DECF    COUNTREG ; (COUNTREG) = (COUNTREG) -1
          BNZ     ADDLOOP ; branch if not zero
```

12. What is the difference between MOVFF and MOVWF?

MOVFF is to copy the content from one RAM location to other RAM location
 $\text{fileREG}_A \rightarrow \text{fileREG}_B$
 MOVFF is four-byte instruction

MOVWF copies the contents of WREG to a location in the file register.
 MOVWF is a two-byte instruction

13. State the value used for each of the following data:

MYDAT_1	EQU 55	;55H
MYDAT_2	EQU D'98'	;62H
MYDAT_3	EQU A'G'	;47H
MYDAT_4	EQU 0x50	;50H
MYDAT_5	EQU B'01111110'	;7E

14. Show a simple code to:

(a) load value 11H into locations 0x00h-0x05h, and

(b) add them together and place the result in WREG as the values are added.

Use EQU to assign the names R0-R5 to locations 0-5

(a)

R0 EQU 0x00

R1 EQU 0x01

R2 EQU 0x02

R3 EQU 0x03

R4 EQU 0x04

R5 EQU 0x05

MOVLW 0x11

MOVWF R0

MOVWF R1

MOVWF R2

MOVWF R3

MOVWF R4

MOVWF R5

(b)

MOVLW 0x00

ADDWF R0, W

ADDWF R1, W

ADDWF R2, W

ADDWF R3, W

ADDWF R4, W

ADDWF R5, W

15. What is the difference between:

```
MOVWF MYREG, F, 1
MOVWF MYREG, F, 0
```

```
MOVWF XXX, F, 0 or
MOVWF XXX, F
```

It means that the access bank is used for the location XXX

```
MOVWF XXX, F, 1
```

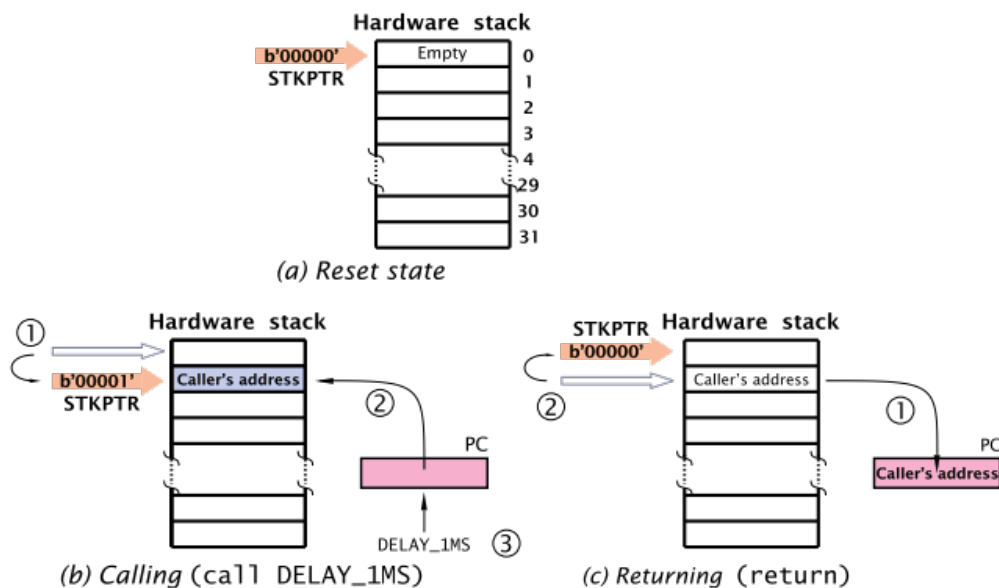
It means that the data Bank specified by the BSR is used.

Example

```
MYREG EQU 0x40
MOVLB 0x02 ; load 2 into BSR ( use data bank 2)
MOVLW 0x01 ; WREG = 01H
MOVWF MYREG, F, 1 ; WREG = 01H, loc 0x240 = '01H'
```

16. When CALL is executed, how many locations of the stack is used?

1 location of the stack is used.



17. Describe the operations involved in a CALL Instruction.

Operations:

1. The Stack Pointer is incremented.
2. Copy the 21-bit contents of the PC into the stack at the location pointed to by the Stack Pointer. Effectively this stored datum will be the address of the instruction following the call instruction.
3. The destination address of the subroutine, that is the location of the entry point instruction of the subroutine, overwrites the original state of the PC. This causes the program execution to transfer to the subroutine.

18. In PIC18, which address is pushed into the stack and what is the change of the stack pointer when a call instruction is executed.

The next instruction pointed by the PC, operation as described in Q17.

The address of the instruction next to the CALL is pushed to the stack and the SP is incremented by one.

19. It is true that target of BNZ can be anywhere in the 2M address space.
NO. only within plus and minus 255 since PCL is used for updating the short jump
 $PCL = PCL + \text{relative address}$

20. Find the number of times the following loop is performed:

	MOVLW	D'200'	
	MOVWF	REGA	; REGA = ' 200 '
BACK	MOVLW	D'100'	; outer loop
	MOVWF	REGB	; REGB = ' 100 '
Here	DECf	REGB, F	; inner loop
	BNZ	Here	; end of inner loop
	DECf	REGA, F	; update the outer loop counter by -1
	BNZ	BACK	; branch to outer loop

'100' * '200' = '20,000'

21. Write a program to get 8-bit data from PORTD and send it to PORTB and PORT C.

```
CLRF  TRISB ; set PORTB output port
CLRF  TRISC ; set PORTC output port
SETF  TRISD ; set PORTD input port
```

```
MOVFW PORTD
MOVWF PORTB
MOVWF PORTC
```

Or

```
MOVFF PORTD, PORTB
MOVFF PORTD, PORTC
```

22. Write a program to add the unpacked BCD digits stored in locations 0x05h. Store the result in location 0x06h.

```
0x05      0x06
OBCD1     OBCD2
```

```
MOVF      0x05, W ; (0x05) -> WREG
ADDWF     0x06, F
DAW
```

MOVF	Move f				
Syntax:	[<i>label</i>] MOVF f [,d [,a]]				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$f \rightarrow \text{dest}$				
Status Affected:	N, Z				
Encoding:	<table><tr><td>0101</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>	0101	00da	ffff	ffff
0101	00da	ffff	ffff		
Description:	The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. 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).				

23. Write a program to add three numbers (in locations 0x00h-0x02h and save the result in file register 0x02.

```
MOVF      0x00, W ; (0x00) -> WREG  move the data from loc 0x00 to WREG
ADDWF     0x01, W ; (WREG) + (0x01) -> WREG
ADDWF     0x02, F ; (WREG) + (0x02) -> 0x02
```

24. Write a program to add three packed BCD numbers (in locations 0-2) and save the result in file registers declared as 0x02:

```
Each loc. Has two BCD    00    BCD2, BCD1
                        01    BCD4, BCD3
                        02    BCD6, BCD5
```

```

L_BYTE EQU 0x06
H_BYTE EQU 0x07

        MOVLW      0          ;clear WREG (WREG =0)
        MOVWF      H_BYTE     ;H_BYTE = '0'
        ADDWF      0x00, W;    ;WREG = 0 + BCD2 BCD1
        DAW        ;WREG = BCD2 BCD1
        BNC        N_1        ;branch if C=0
        INCF        H_BYTE, F  ;Add '1' to H_BYTE if C=1

N_1      ADDWF      0x01, W     ;WREG = BCD4 BCD3 + BCD2 BCD1
        DAW        ;WREG = BCD4 BCD3 + BCD2 BCD1 + C
        BNC        N_2        ; branch if C=0
        INCF        H_BYTE, F  ; Add '1' to H_BYTE if C=1

N_2      ADDWF      0x02, W     ; WREG = BCD6 BCD5 + BCD4BCD3
        ;          + BCD2 BCD1 + C
        DAW        ;
        BNC        N_3        ; branch if C=0
        INCF        H_BYTE, F  ; Add '1' to H_BYTE if C=1

N_3      MOVWF      L_BYTE;
        MOVFF      L_BYTE, 0x02
        MOVFF      H_BYTE, 0x03
```

25. Write a program to add 7F9Ah to BC48h, and save the result starting at 0x40h.

```

; loc for addition
BYTE_L EQU 0x40 ; BYTE_L = loc 0x40  9A + 48 (1)
BYTE_H EQU 0x41 ; BYTE_H = loc 0x41  7F + BC + C from (1) (2)
BYTE_U EQU 0x42 ; BYTE_U = loc 0x42  Carry from (2)

; Initialize BYTE as 007F9A

                                WRED   42    41    40
MOVLW 0x9A                     9A
MOVWF BYTE_L                   9A          9A
MOVLW 0x7F                     7F
MOVWF BYTE_H                   7F          7F  9A
MOVLW 0                         0          7F  9A ;           U  H  L
MOVWF BYTE_U                   0          0  7F  9A ; BYTE: 00 7F 9A

; ADD 48 with 9A
MOVLW 0x48                     48          0  7F  9A
ADDWF BYTE_L, F                48          0  7F  9A + 48

; ADD 7F with BC
MOVLW 0xBC                     BC          0  7F  9A + 48
ADDWFC BYTE_H, F               BC          0  BC + 7F + C BYTE_L addition
MOVLW 0x00                     00          0  BC + 7F + C
ADDWFC BYTE_U, F               00          00 + BC + 7F + C

```

ADDWFC	ADD W and Carry bit to f				
Syntax:	[<i>label</i>] ADDWFC f [,d [,a]]				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(W) + (f) + (C) \rightarrow \text{dest}$				
Status Affected:	N,OV, C, DC, Z				
Encoding:	<table><tr><td>0010</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>	0010	00da	ffff	ffff
0010	00da	ffff	ffff		
Description:	Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden.				