Program memory Consideration & Addressing Modes

Memory Organization

- Program Memory
- Register File Memory

Program Memory

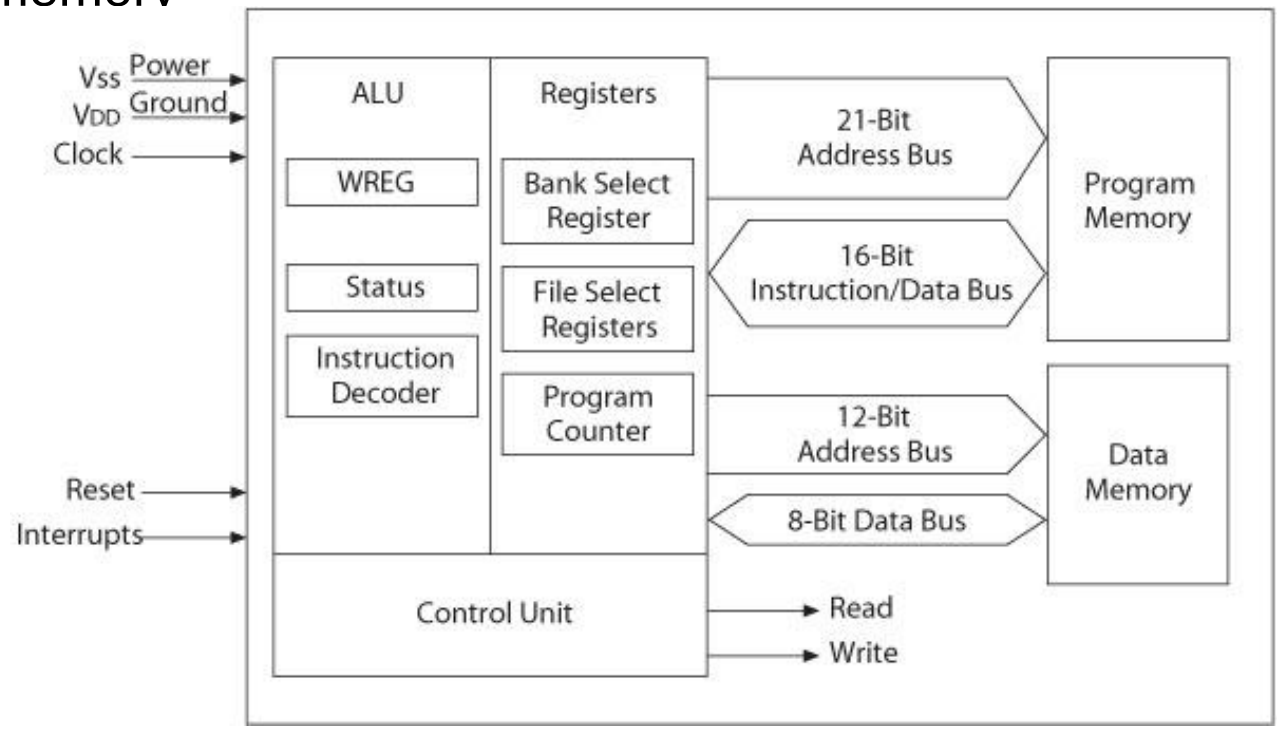
- Used for storing compiled code
- Each location is 14 bits long
- Every instruction is coded as a 14 bit word
- Addresses H’000’ and H’004’ are treated in a special way
- PC can address up to 8K addresses

Register File Memory

- Consist of 2 Components
  - General Purpose Register (GPR) Files (RAM)
  - Special Purpose Register (SPR) files
- This portion of memory is separated into banks of 128 bytes long

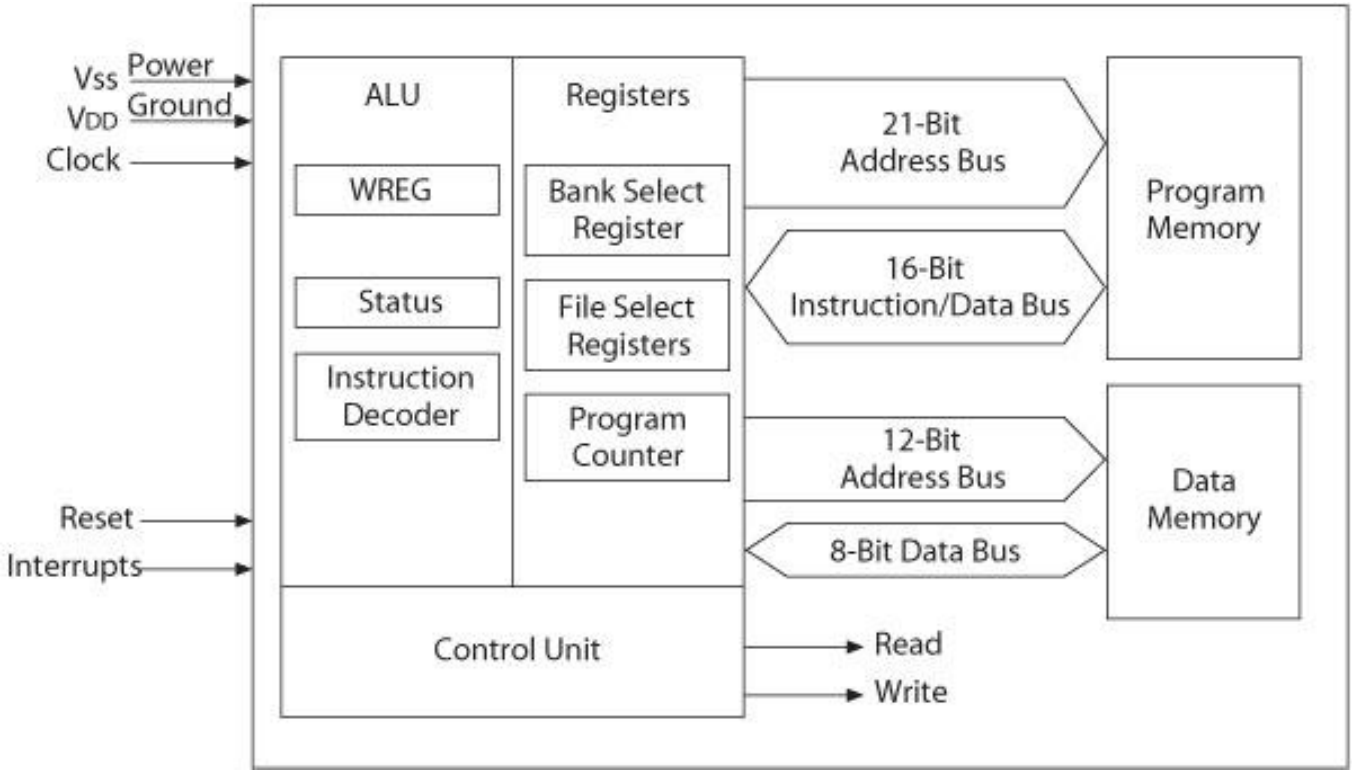
PIC18F - Address Buses

- Address bus
  - 21-bit address bus for program memory addressing capacity: 2 MB of memory
  - 12-bit address bus for data memory addressing capacity: 4 KB of memory



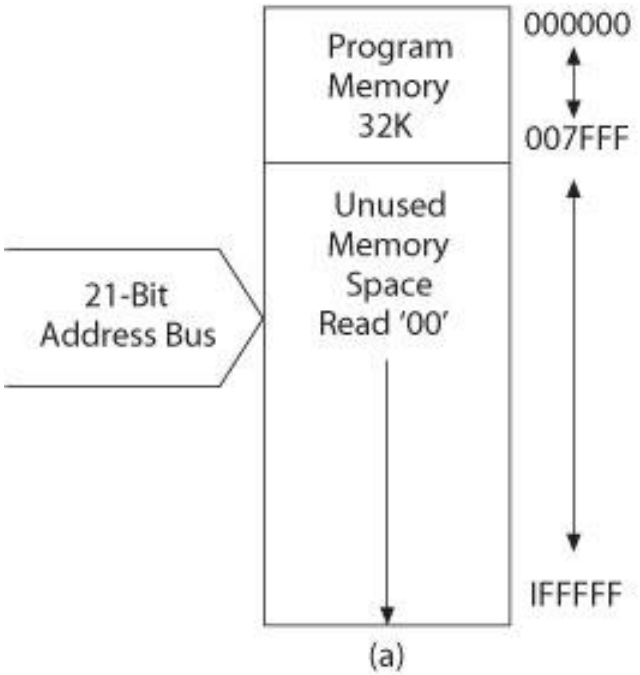
Data Bus and Control Signals

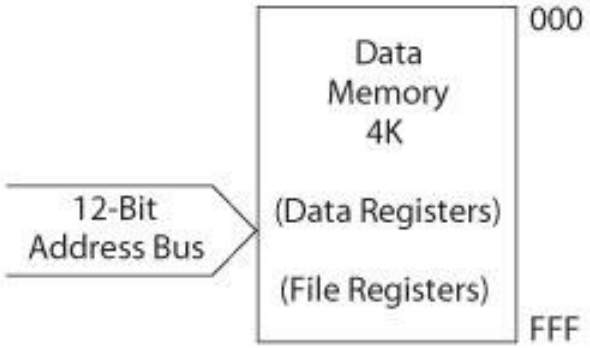
- Data bus
  - 16-bit instruction/data bus for program memory
  - 8-bit data bus for data memory
- Control signals
  - Read and



PIC18F452/4520 Memory

- Program memory with addresses (Flash)
- Data memory with addresses





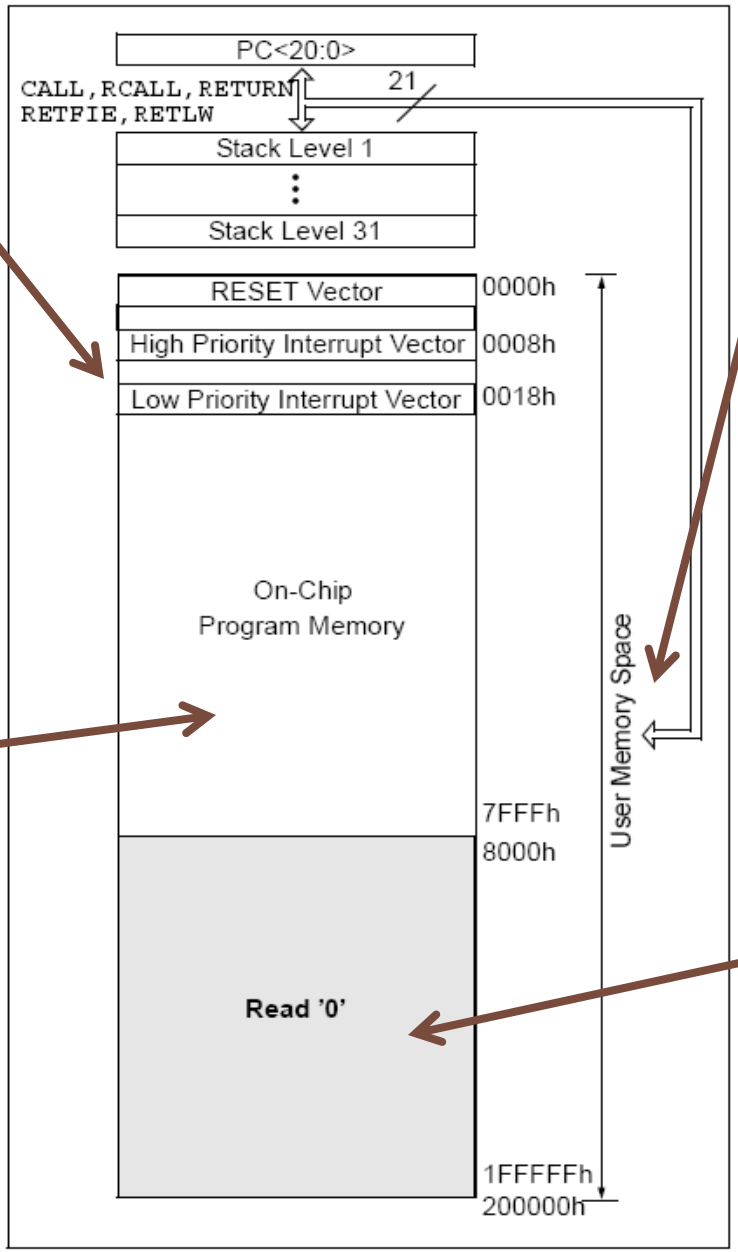
$FFF=2^{12}=16x256=4096$

=4K

Program Memory

The RESET vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

PIC18F452 each have 32 Kbytes of FLASH memory. This means that it can store up to 16K of single word instructions

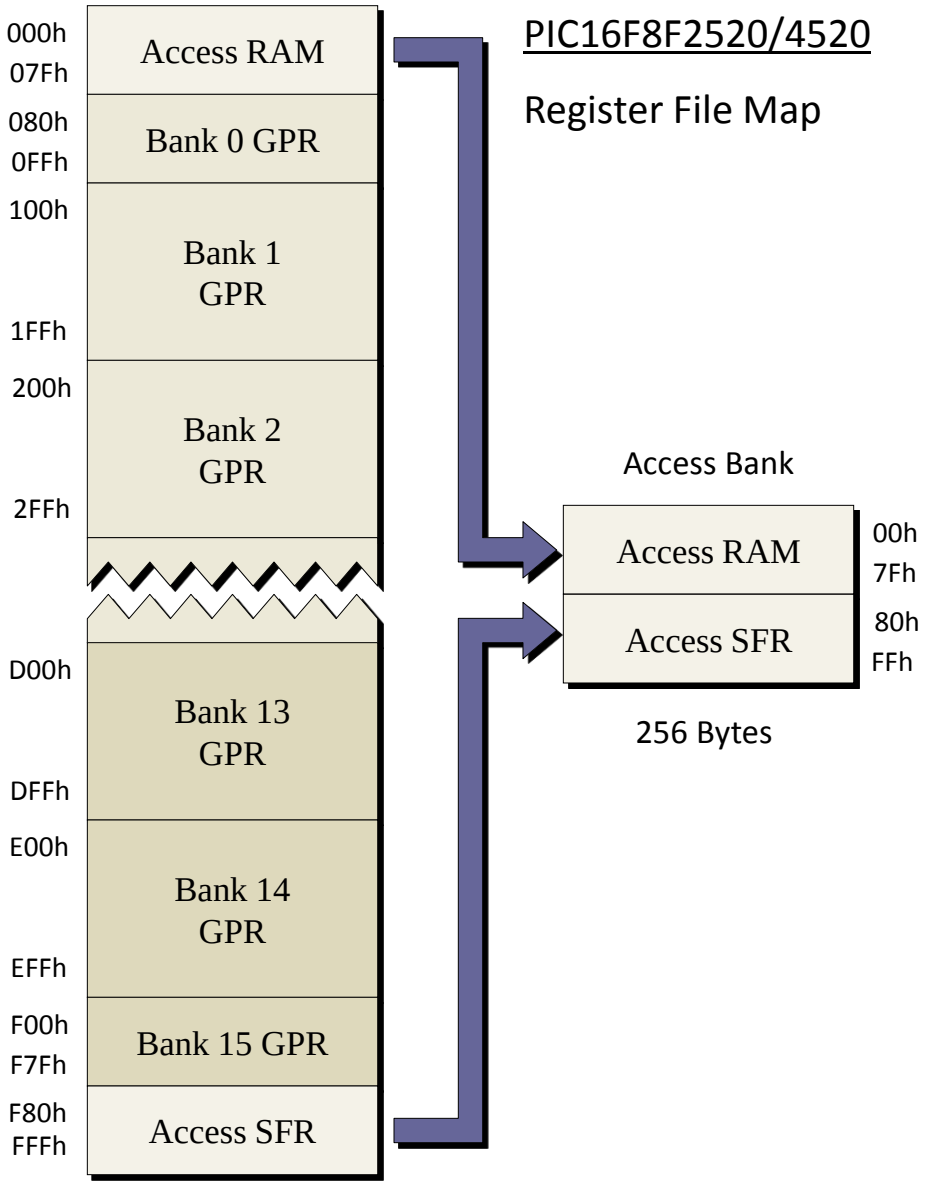


A 21-bit program counter is capable of addressing the 2-Mbyte program memory space.

Accessing a location between the physically implemented memory and the 2-Mbyte address will cause a read of all '0's (a NOP instruction).

Data Memory Organization

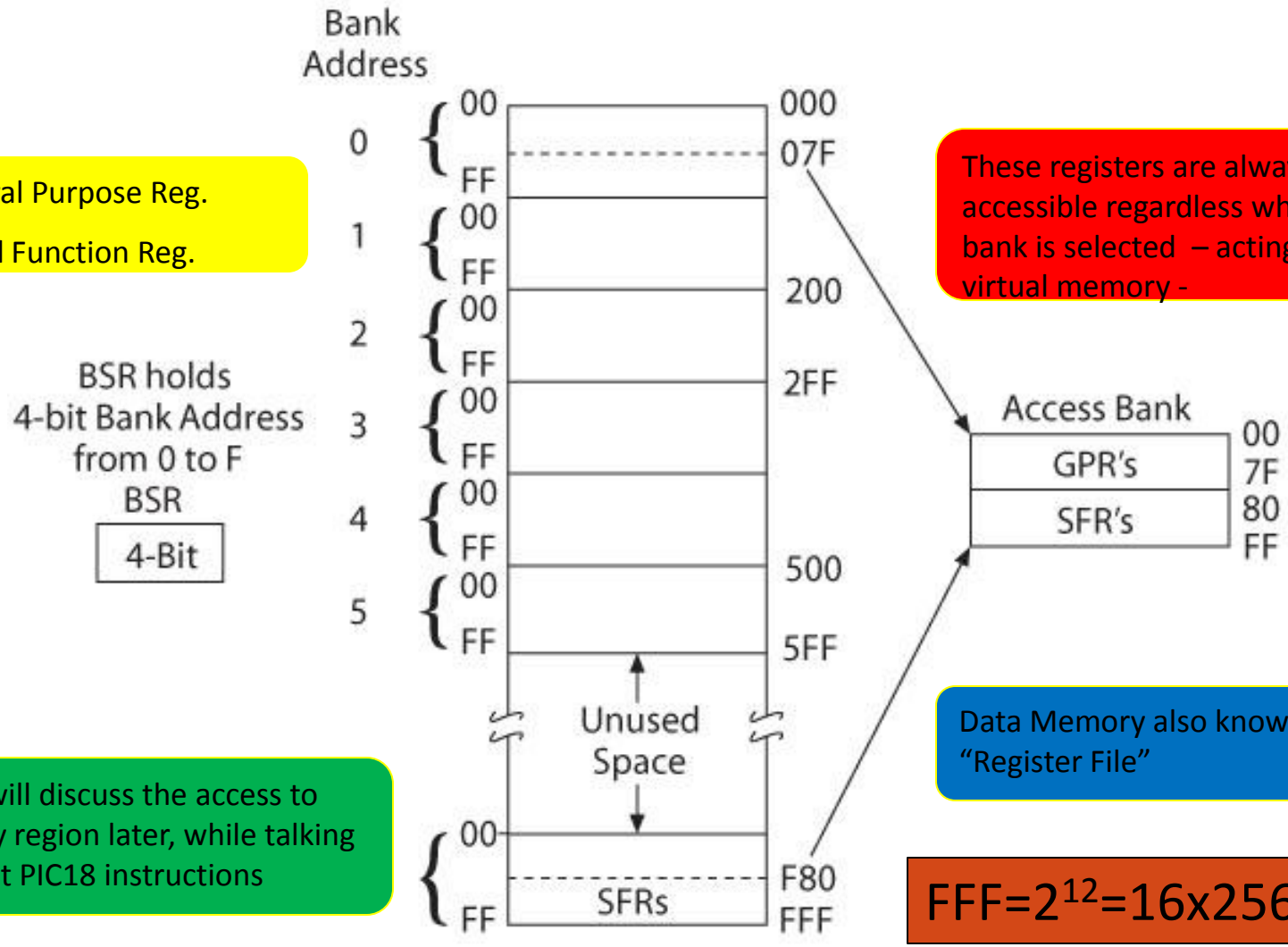
- Data Memory up to 4k bytes
- Divided into 256 byte banks
- Half of bank 0 and half of bank 15 form a virtual bank that is accessible no matter which bank is selected



Data Memory with Access Banks

GPR=General Purpose Reg.

SFR=Special Function Reg.



These registers are always accessible regardless which bank is selected – acting as a virtual memory -

Data Memory also known as "Register File"

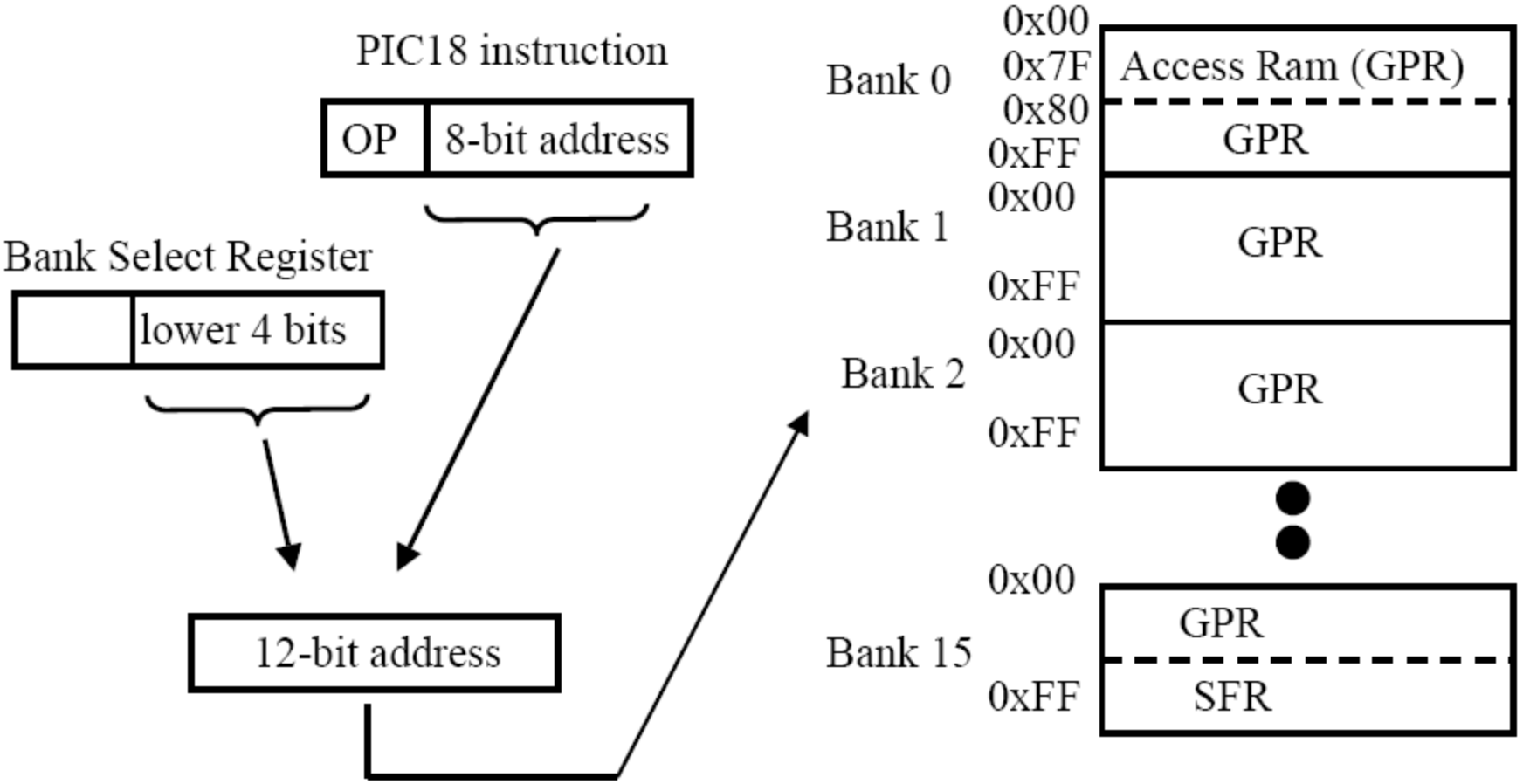
We will discuss the access to every region later, while talking about PIC18 instructions

$FFF=2^{12}=16x256=4096$

$=4K$

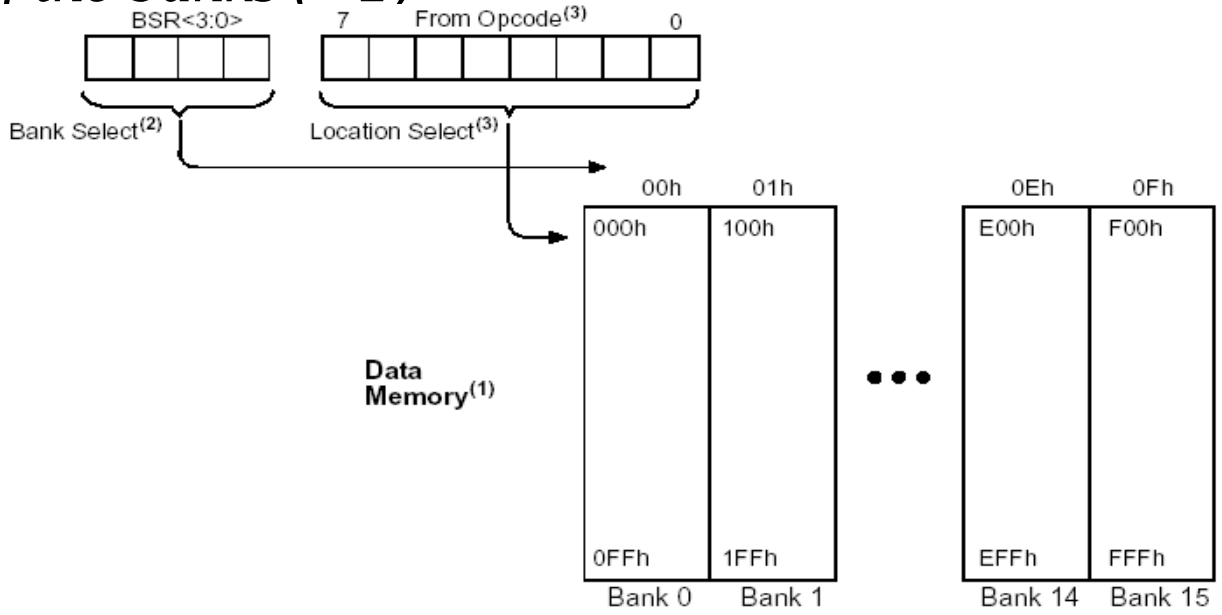
Accessing Data Memory

- The machine code for a PIC18 instruction has only **8** bits for a data memory address which needs 12 bits. The Bank Select Register (BSR) supplies the other 4



Data Memory Addressing

- *Direct Addressing - Operand address(es) embedded in the opcode*
  - 8 bits of the 16-bit instruction specify any one of 256 locations
  - The 9th bit specifies either the *Access Bank (=0)* or *one of the banks (=1)*



Data Memory Addressing

- *Direct Addressing Examples*
  - Direct addressing (banked)

```
movlb  02                    ;set BSR to Bank 2
addwf  0x55, W, BANKED       ; add WREG with the content of
                             ; addr. 55 (f=55) in bank 2 (a=1),
                             ; save the result to WREG (d=0)
```

Operand is the content of data memory at add. 0x255

Mnemonic in MPASM:

A (a=0) - the access bank; BANKED (a=1) - banked

W (d=0) - the WREG register; F (d=1) - the data register

Data Memory Addressing

- *Direct Addressing Examples*
  - Direct addressing (using access bank)

```
addwf  0x55, F, A        ;movlb not required
                         ; add WREG to content of
                         ; addr. 55 (f=55) in access
                         ; bank (a=0), save the result
                         ; in the data memory at the
                         ;address 0x55 (d=0)
```

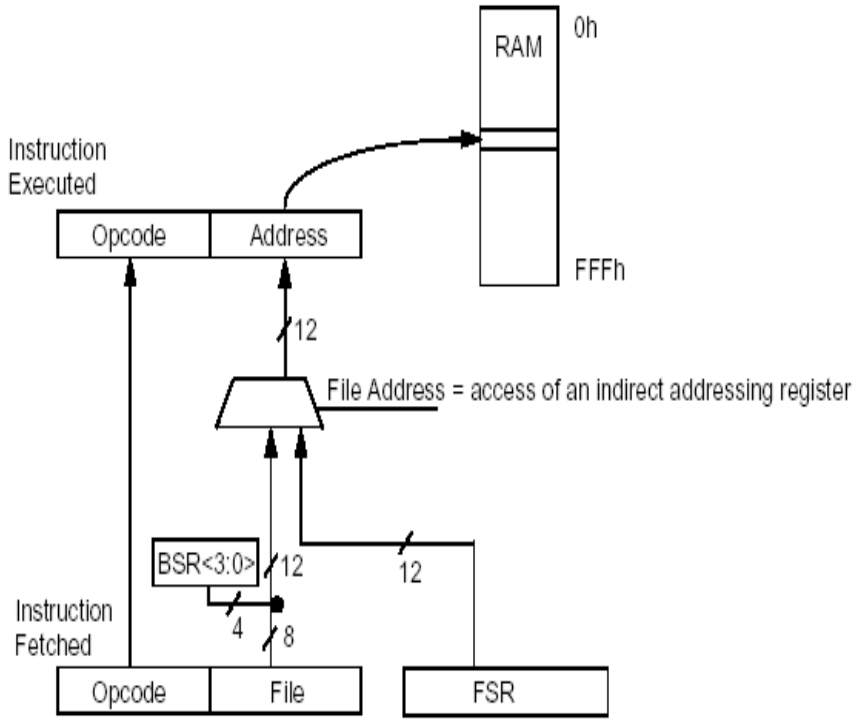
Operand is the content of data memory add. 0x055

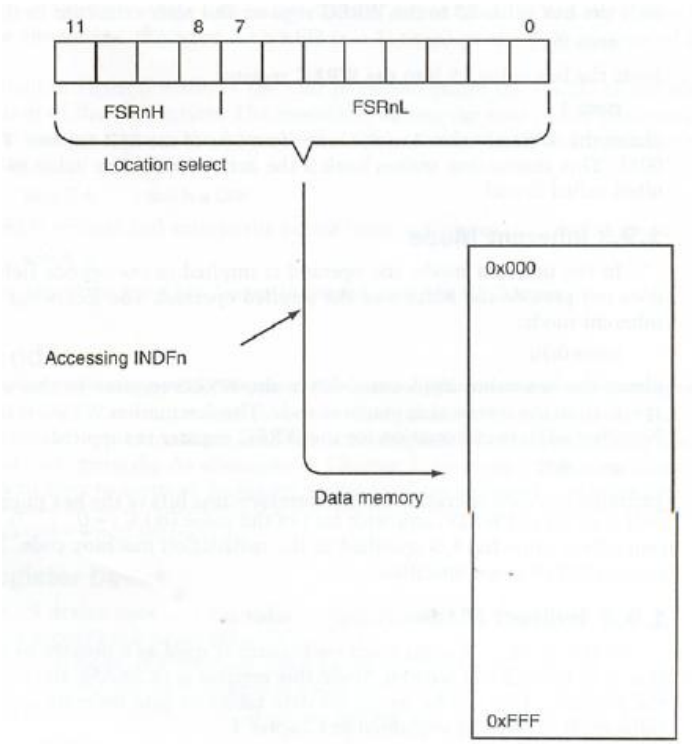
Data Memory Addressing

- *Indirect Addressing*
  - 3 File Select Registers (FSR) as a pointer to the data memory location that is to be read or written.
    1. FSR0: composed of FSR0H:FSR0L
    2. FSR1: composed of FSR1H:FSR1L
    3. FSR2: composed of FSR2H:FSR2L
  - Each FSR has an INDF register associated with it
  - The $INDF_n$ register is not a physical register. Addressing $INDF_n$ actually addresses the register whose address is contained in the FSR_n register.

Data Memory Addressing

- *Indirect Addressing*

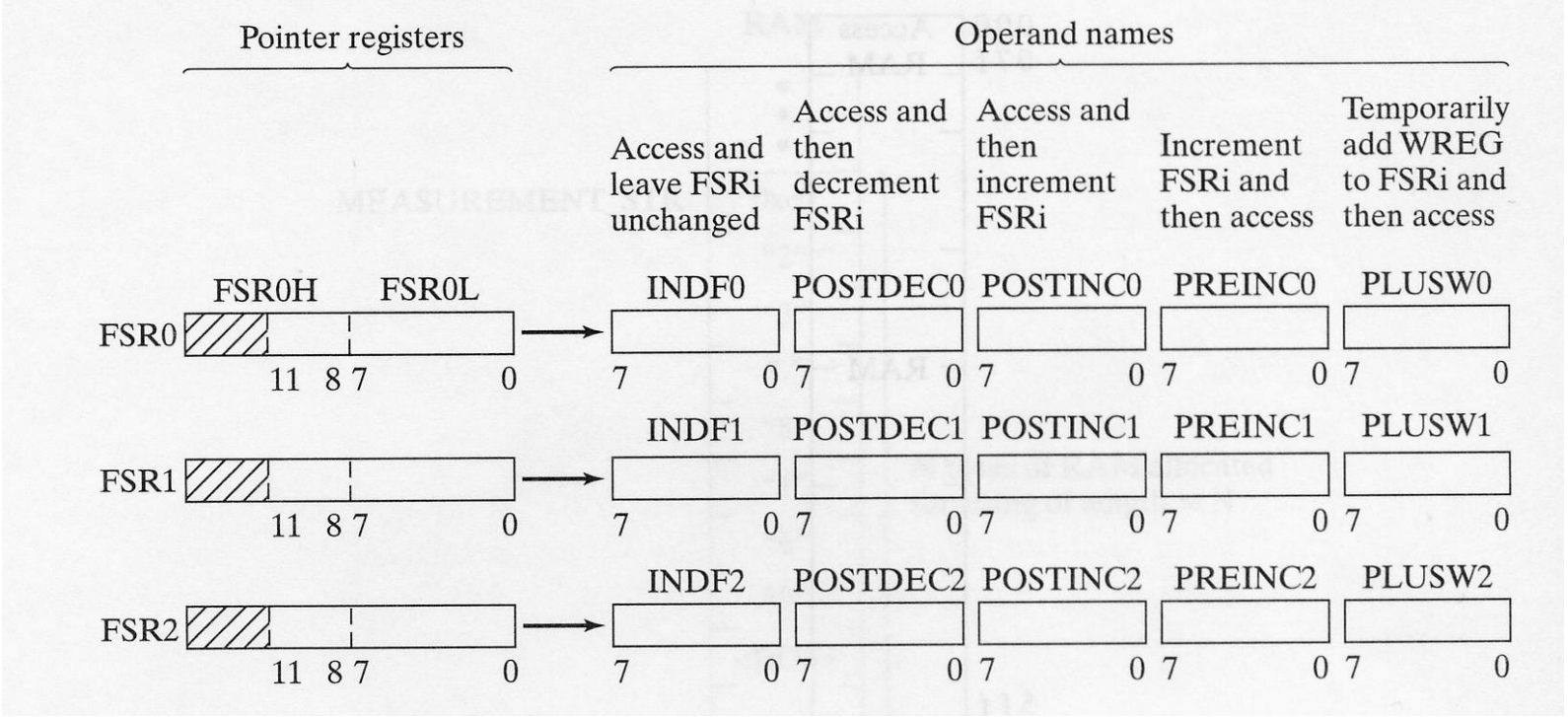





```
LFSR  02, num1       ;load FSR2 with the add. of num1
MOVWF INDF2, W       ; move WREG to the register
                     ; pointed by FSR2
```

Data Memory Addressing

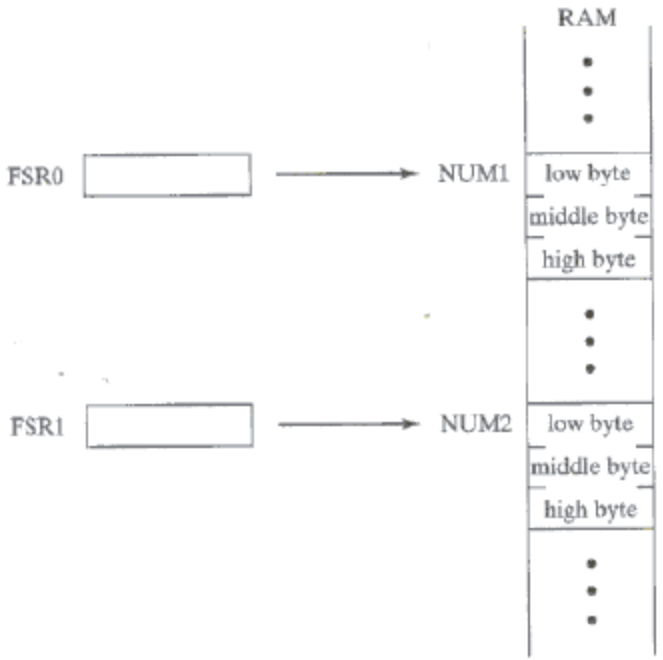
- *Indirect Addressing Operations*



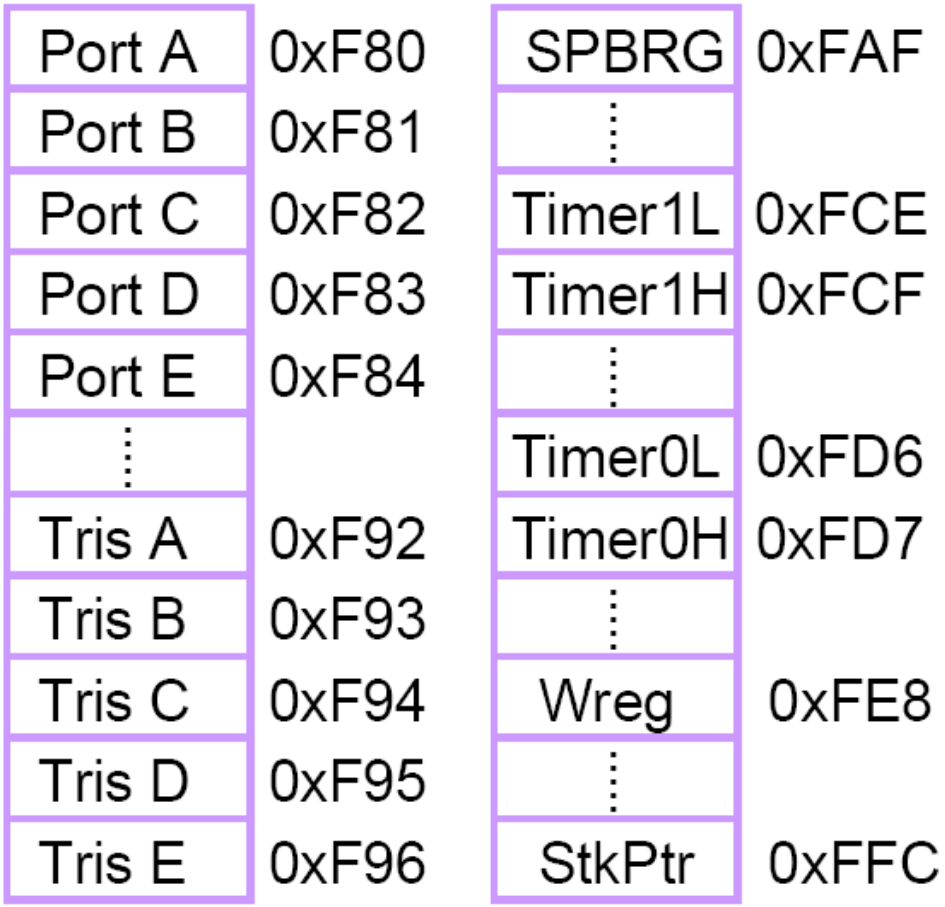
Data Memory Addressing

- *Indirect Addressing Example*

```
        count set 0x02
        lfsr 0, num1
        lfsr 1, num2
        movlw 3
        movwf count, A
        bcf  STATUS, c
Again:  movf  POSTINC1, W
        addwfc POSTINC0, F
        decfsz count, 1
        bra Again
```



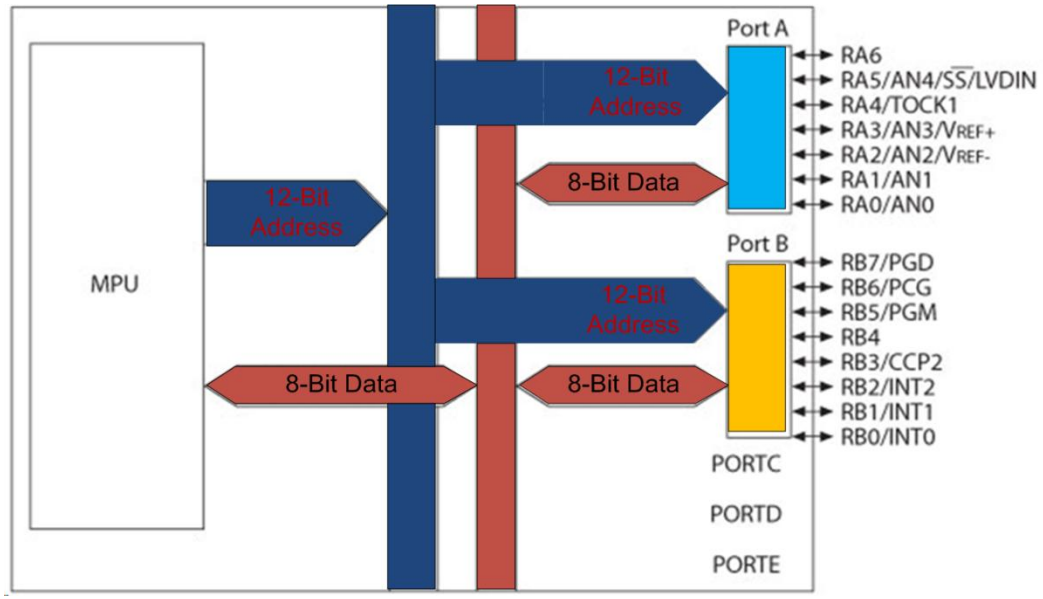
SFRs Examples



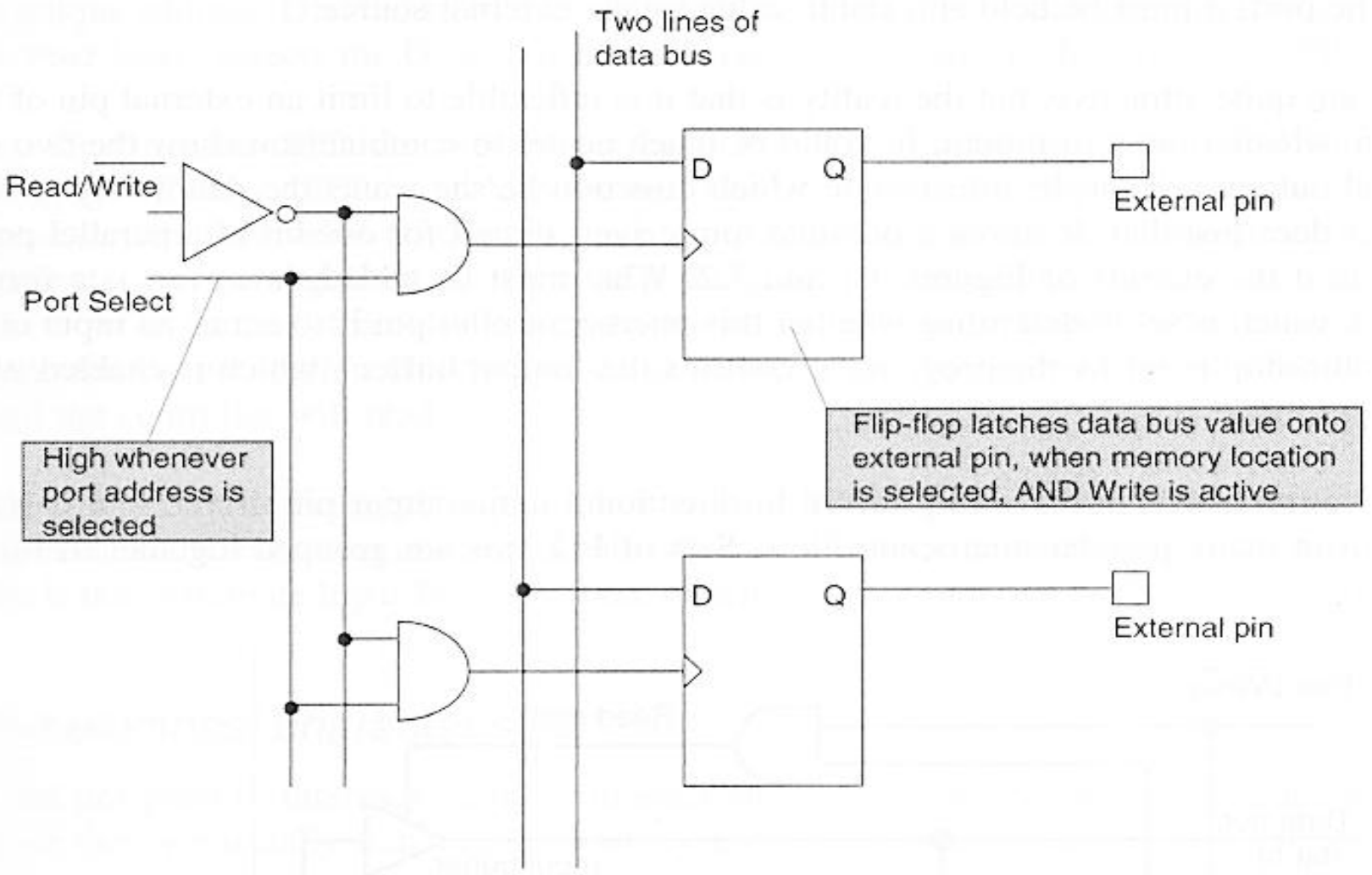
I/O Ports

PIC18F452 I/O Ports

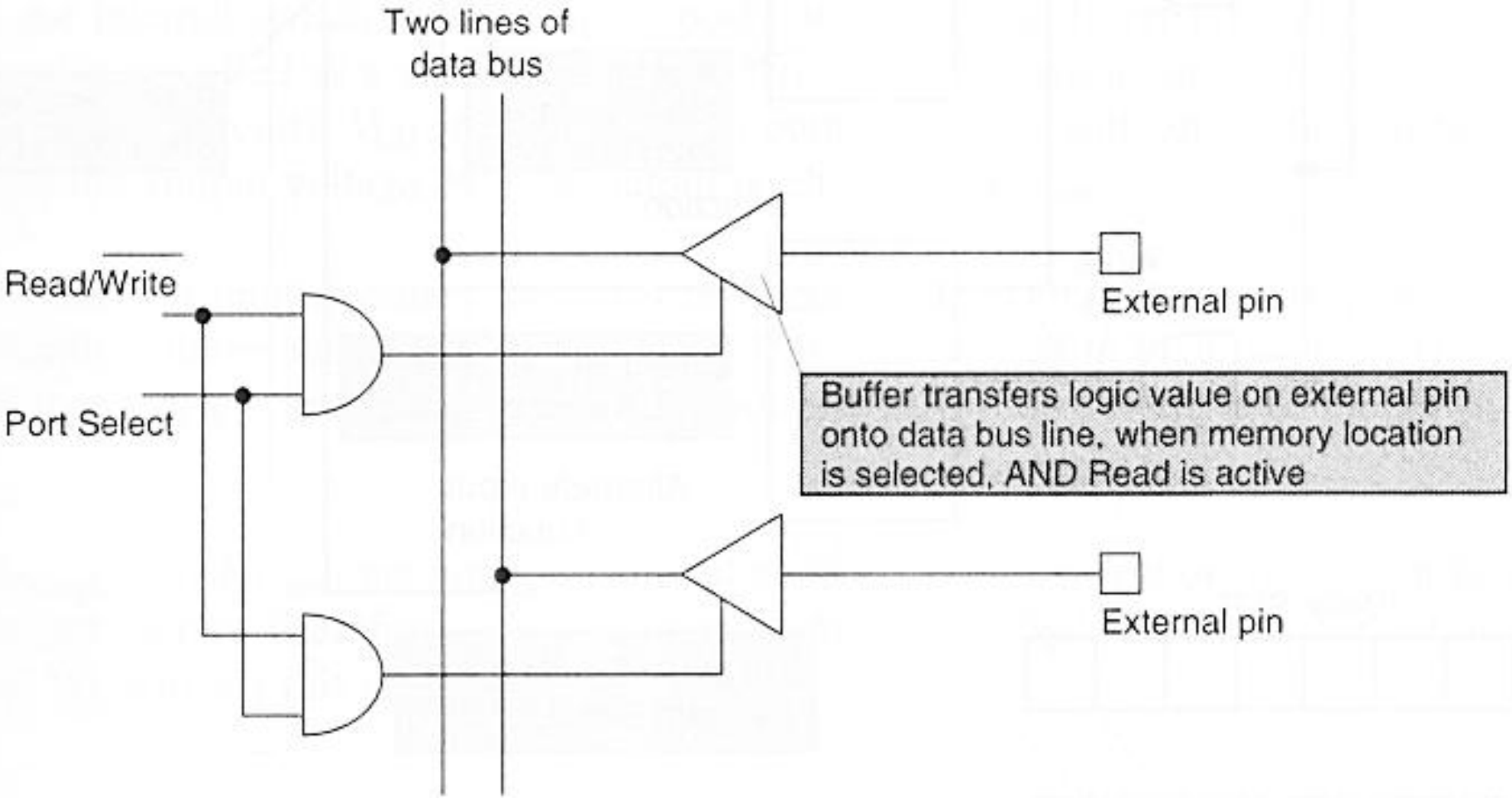
- Five I/O ports
  - PORT A through PORT E
  - Most I/O pins are multiplexed
  - Generally have eight I/O pins with a few exceptions
  - Addresses already assigned to these ports in the design stage
  - Each port is identified by its assigned SFR



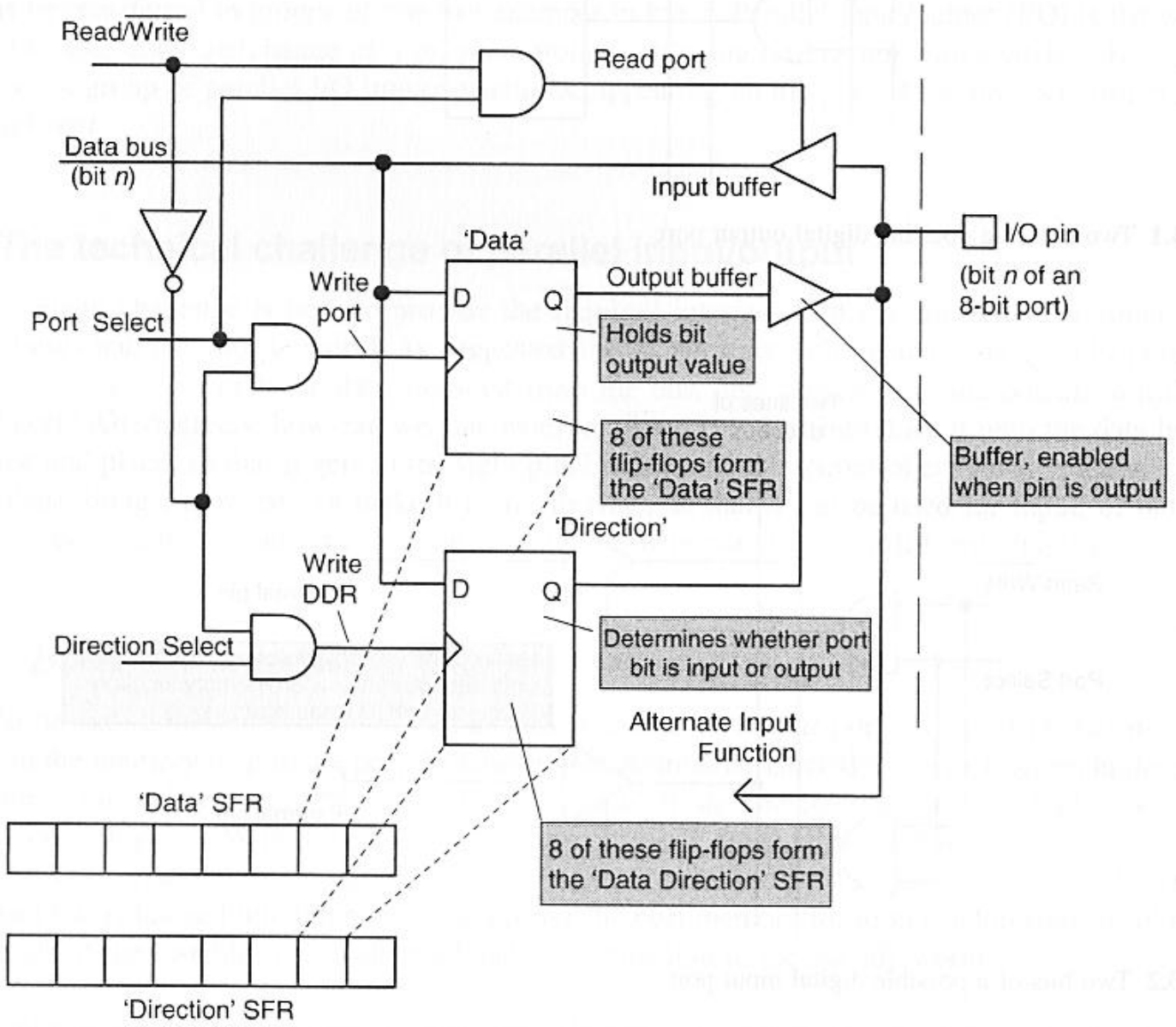
Parallel I/O Output Structure



Parallel I/O Input Structure



Parallel I/O Combined I/O Structure



Parallel I/O ports Main Features

- Simple memory mapped access
- Can be configured through software as either input or output
- Ability to set or reset individual bits
- Can have internal pull-ups
- Can drive small loads like LEDs
- Can be multifunction
- Different capability for pins (i.e. larger current)

Parallel I/O ports

- For most ports, the I/O pin's direction (input or output) is controlled by the data direction register **TRISx** (x=A,B,C,D,E): a '1' in the TRIS bit corresponds to that pin being an input, while a '0' corresponds to that pin being an output
- The **PORTx** register is the latch for the data to be output. Reading PORTx register read the status of the pins, whereas writing to it will write to the port latch.
- **Example: Initializing PORTB** (PORTB is an 8-bit port. Each pin is individually configurable as an input or output).

```
bcf     STATUS, RP0          ; select bank0
bcf     STATUS, RP1
clrf    PORTB                ; clear PORTB output data latches
bsf     STATUS, RP0          ; select bank1
movlw   0xCF                 ; value used to initialize data direction
movwf   TRISB          ; PORTB<7:6>=inputs, PORTB<5:4>=outputs,
                       ; PORTB<3:0>=inputs
```

Relationship between TRIS and PORT Registers

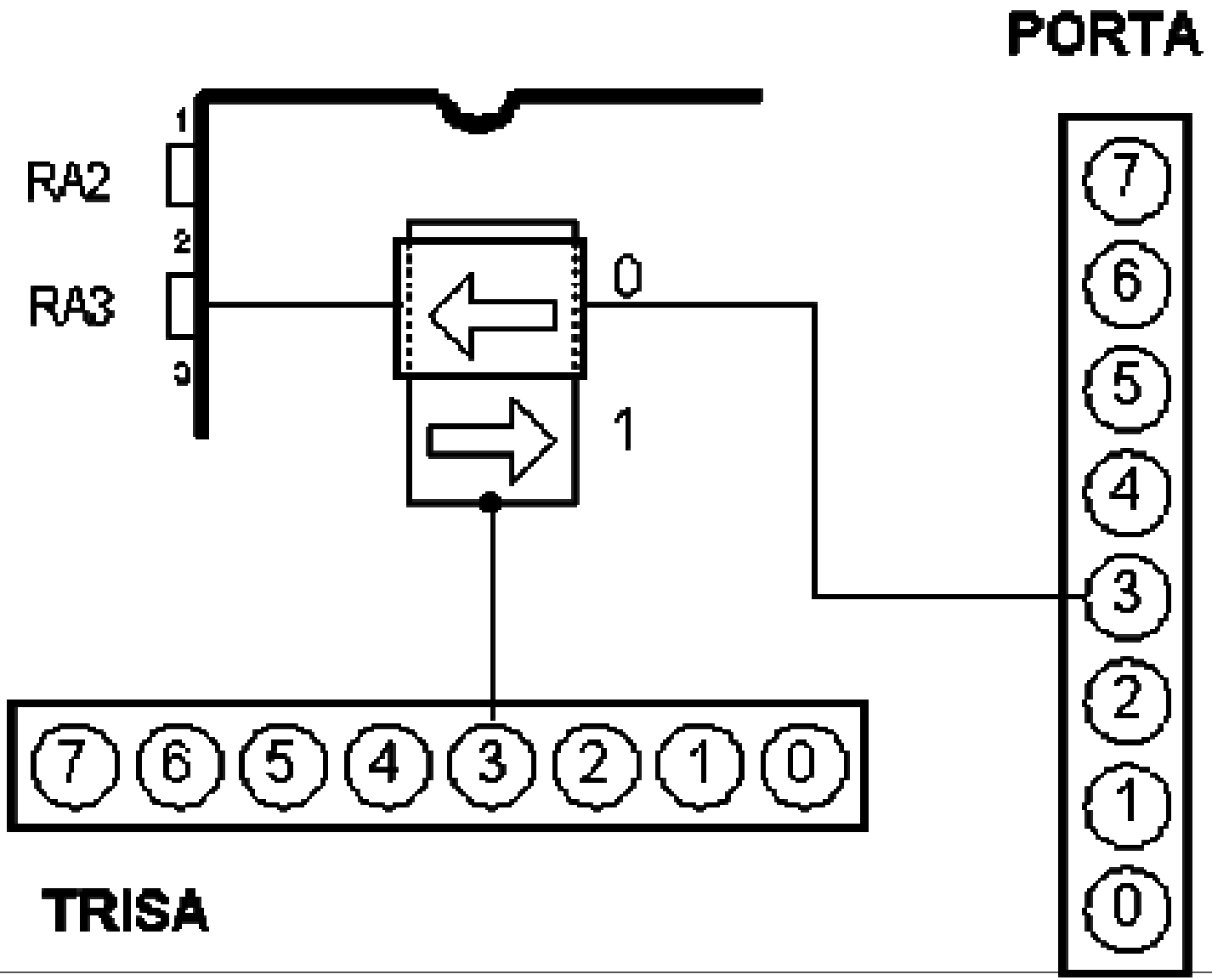


Illustration: Displaying a Byte at an I/O Port (1 of 5)

- Problem statement:
  - Write instructions to light up alternate LEDs at PORTC.
- Hardware:
  - PORTC
    - bidirectional (input or output) port; should be setup as output port for display
  - Logic 1 will turn on an LED in Figure 2.10.

Illustration (2 of 5)

- Interfacing LEDs to PORTC
- Port C is F82H
- Note that PORT C is set to be an output!
- Hence, TRISC (address 94H) has to be set to 0

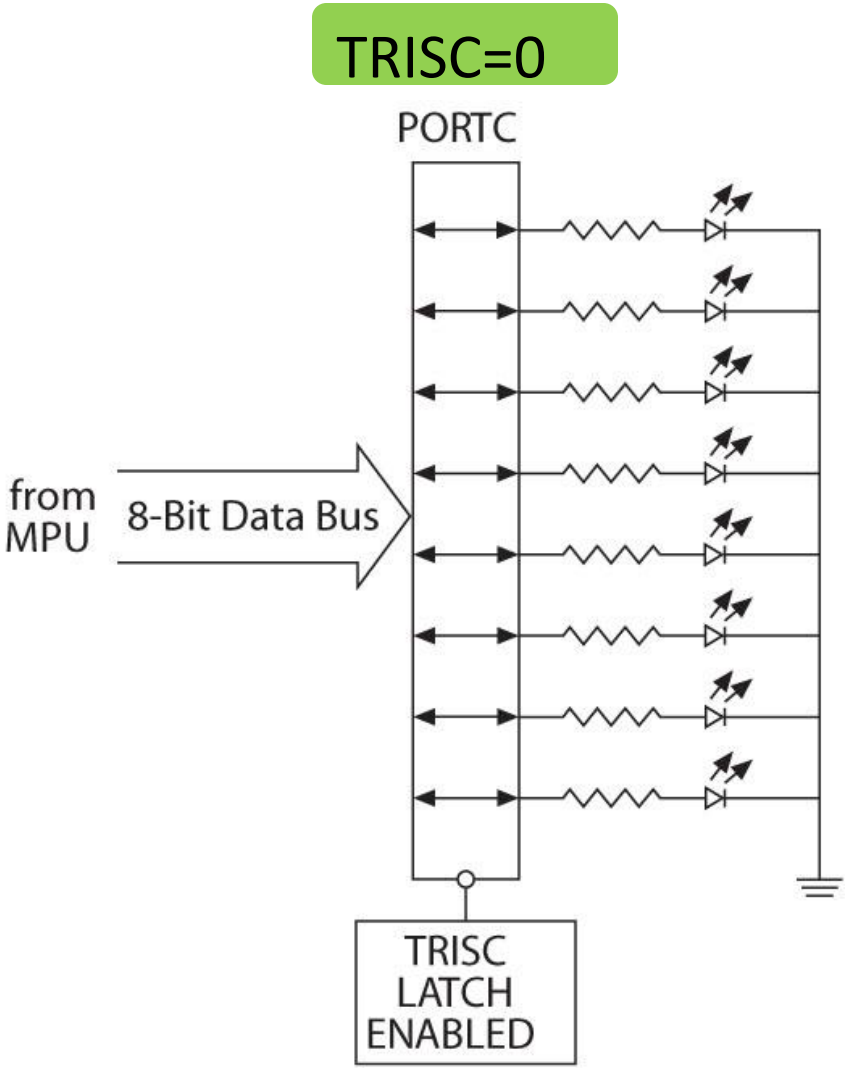


Illustration (3 of 5)

- Program (software)
  - Logic 0 to TRISC sets up PORTC as an output port
  - Byte 55H turns on alternate LEDs
    - MOVLW 00 ;Load W register with 0
    - MOVWF TRISC, 0 ;Set up PORTC as output
    - MOVLW 0x55 ;Byte 55H to turn on LEDS
    - MOVWF PORTC,0 ;Turn on LEDs
    - SLEEP ;Power down

Register Addressing Modes

- There are 3 types of addressing modes in PIC
  - Immediate Addressing
    - Movlw H’0F’
  - Direct Addressing
  - Indirect Addressing

Direct Addressing

- Uses 7 bits of 14 bit instruction to identify a register file address
- 8^{th} and 9^{th} bit comes from RP0 and RP1 bits of STATUS register.
- Exp:

```
Z equ  D'2'
       btfss STATUS, Z
```

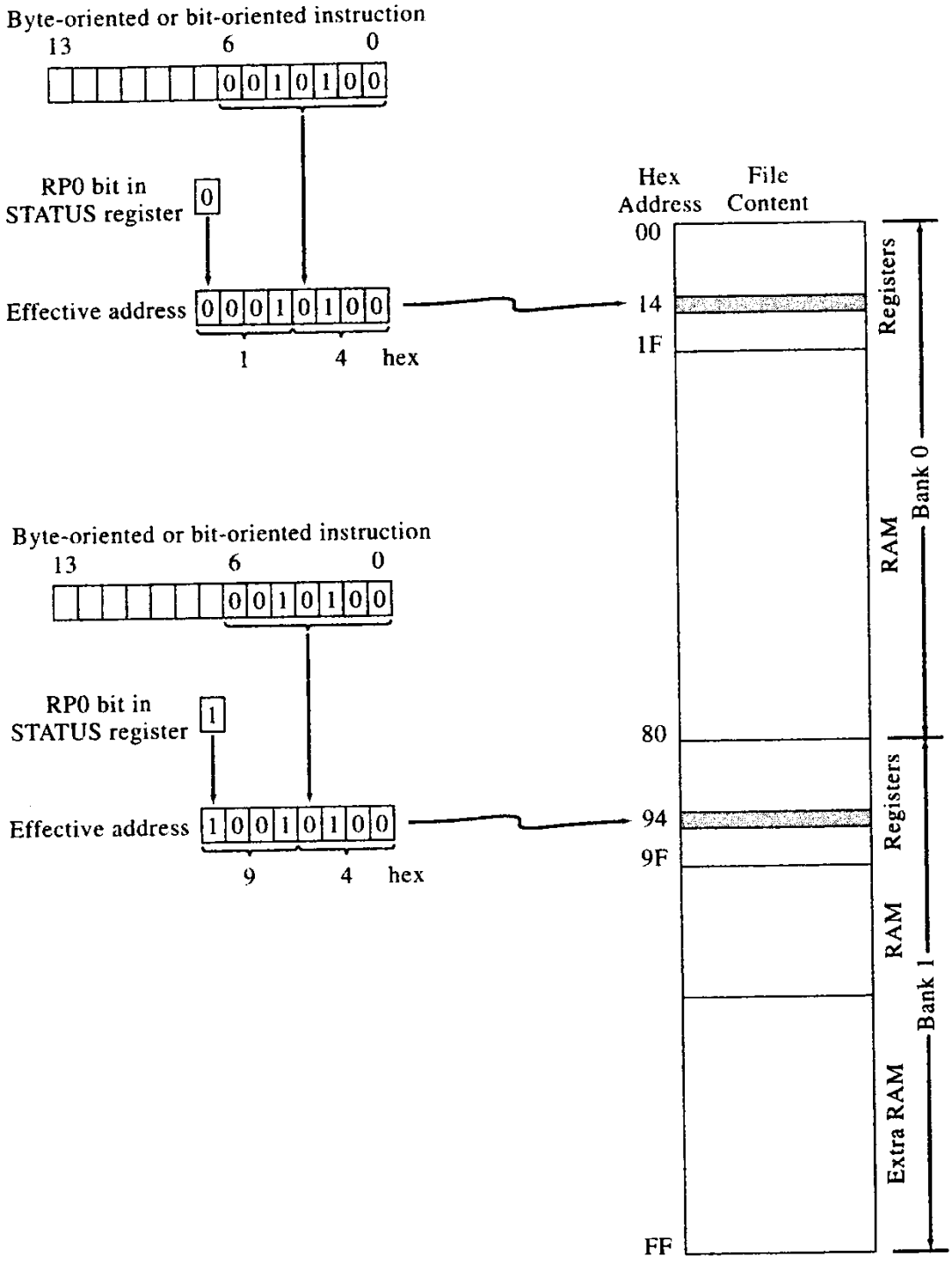
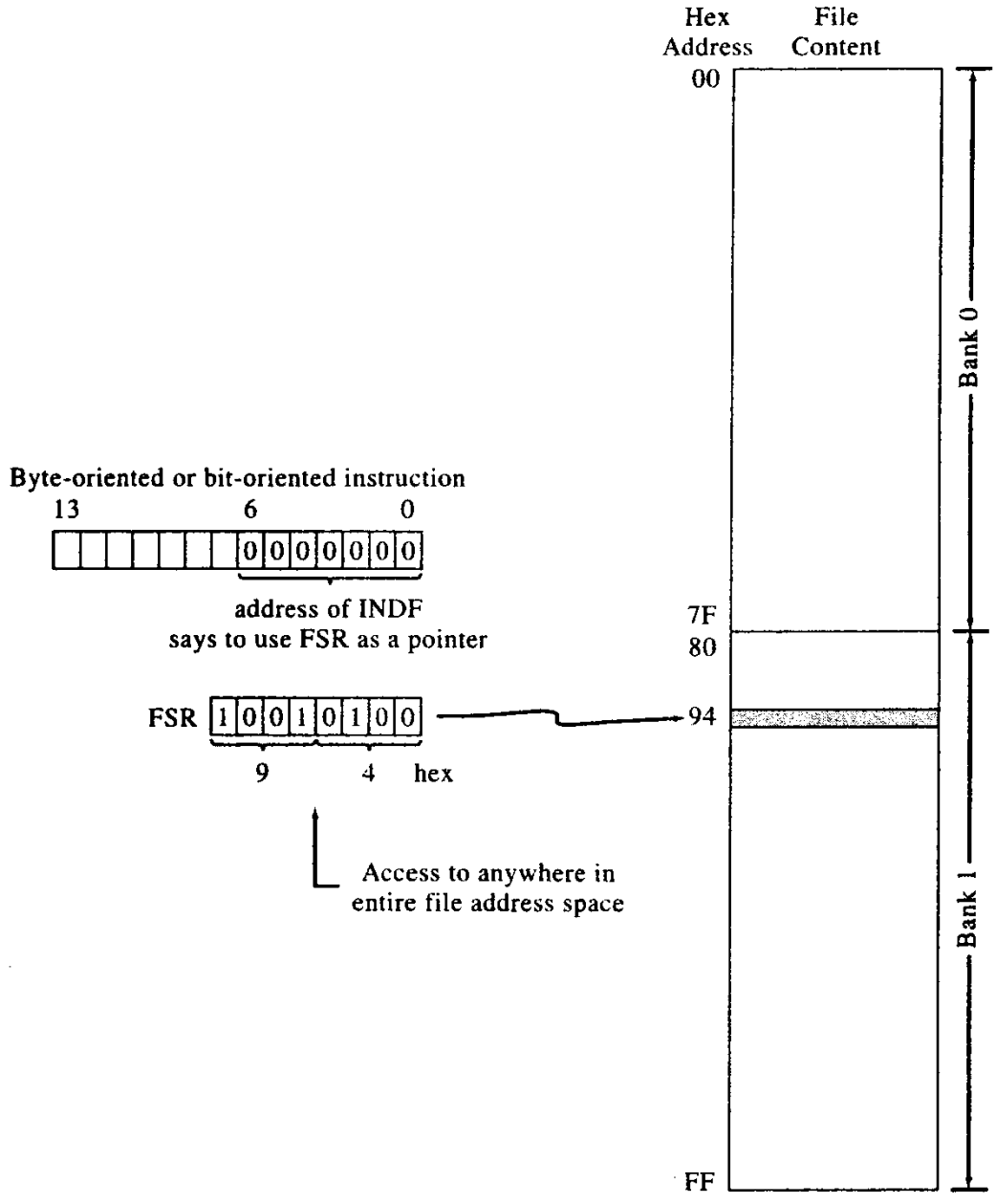


Figure 6 Direct addressing mode.

Indirect Addressing

- Full 8 bit register address is written the special function register FSR
- INDF is used to get the content of the address pointed by FSR
- Exp : A sample program to clear RAM locations H’20’ – H’2F’ .



igure 7 **Indirect addressing mode.**

Some CPU Registers

- STATUS
- PC
- W
- PCL
- PCLATH

Instruction Set

- Every Instruction is coded in a 14 bit word
- Each instruction takes one cycle to execute
- Only 35 instructions to learn (RISC)

Instruction Set

- Uses 7 bits of 14 bit instruction to identify register file address
- For most instructions, W register is used as a source register
- The result of an operation can be stored back to the W register or back to source register

Some Arithmetic Operations

- addwf FSR, w ; Add w to FSR and put result in w
- iorwf TMR0, f ; Inclusive OR w with TMR0 and store result in TMR0
- addwf reg ; Add content of the reg to content of the w and store the result back into reg (source)