16.480/552

Micro II and Embedded Systems

Introduction to PIC Microcontroller

Revised based on slides from WPI ECE2801

Moving Towards Embedded Hardware

Typical components of a PC:

- x86 family microprocessor
- Megabytes or more of main (volatile) memory
- Gigabytes or more of magnetic memory (disk)
- Operating System (w/ user interface)
- Serial I/O
- Parallel I/O
- Real-time clock
- Keyboard
- Video Display
- Sound card
- Mouse
- NIC (Network Interface Card)
- etc.

The PIC 16F87x Series Microcontroller

Some of the characteristics of the PIC 16F87x series

- High performance, low cost, for embedded applications
- Only 35 instructions
- Each instruction is exactly one word long and is executed in 1 cycle (except branches which take two cycles)
- 4K words (14bit) flash program memory
- 192 bytes data memory (RAM) + 128 bytes EEPROM data memory
- Eight level deep hardware stack
- Internal A/D converter, serial port, digital I/O ports, timers, and more!

What these changes mean when programming?

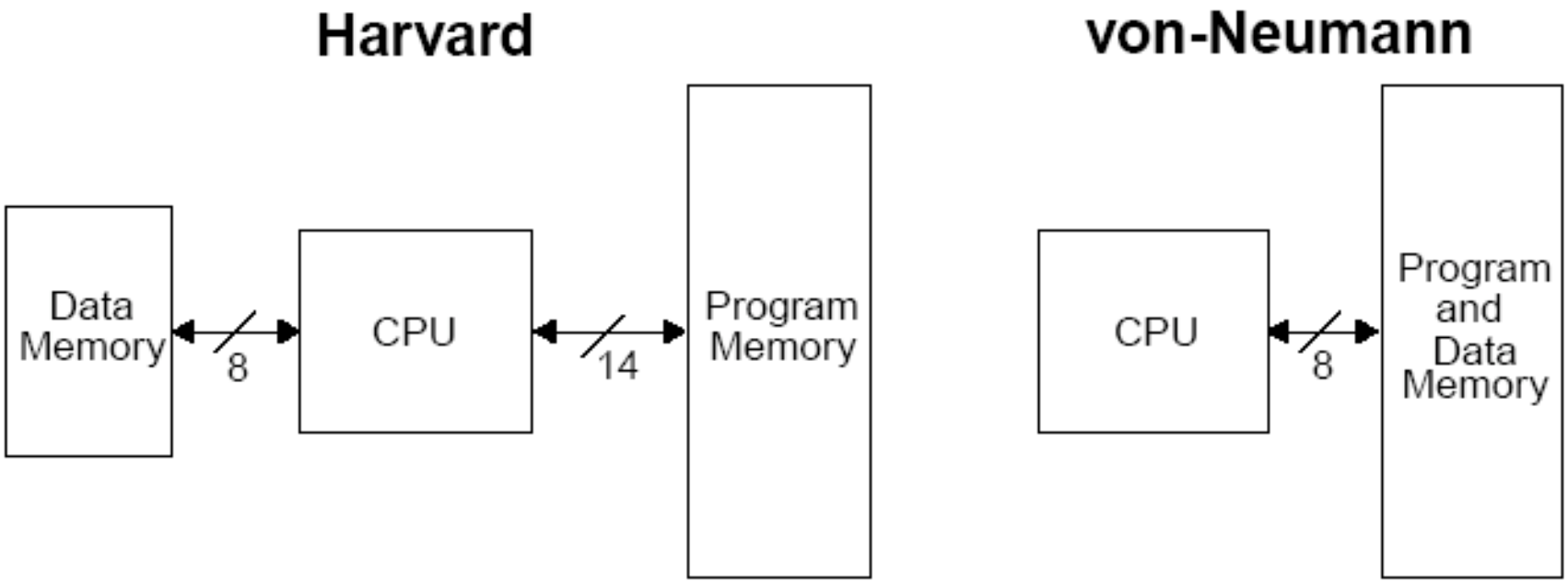
- Since there is no DOS or BIOS, you'll have to write I/O functions yourself. Everything is done directly in assembly language.
- Code design, test, and debug will have to be done in parallel and then integrated after the hardware is debugged.
- Space matters! Limitations on memory may mean being very clever about algorithms and code design.

PIC16F87X

Key Features PICmicroô Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	-	PSP	-	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions

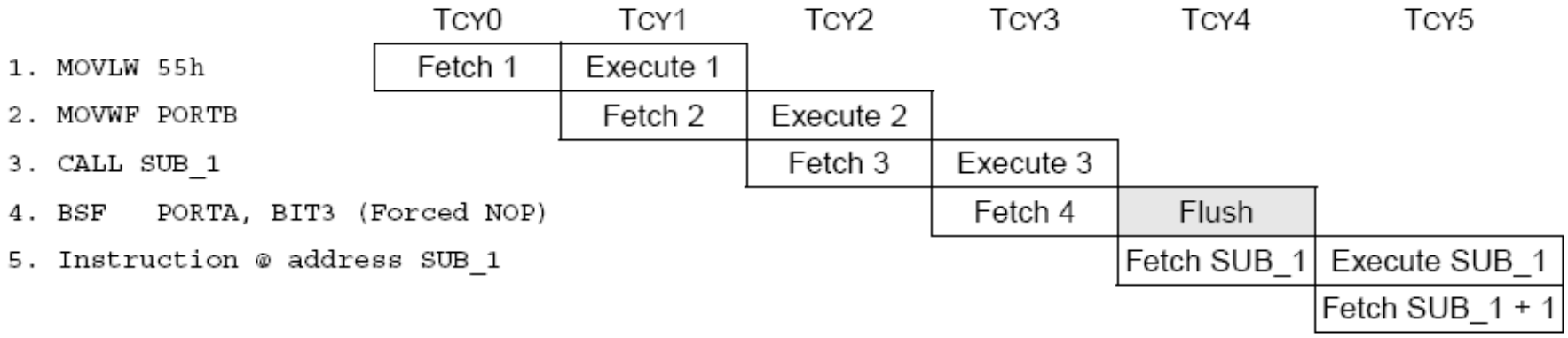
Harvard Architecture

Program memory and Data memory are separated memories and they are accessed from separated buses.



Instruction Pipelining

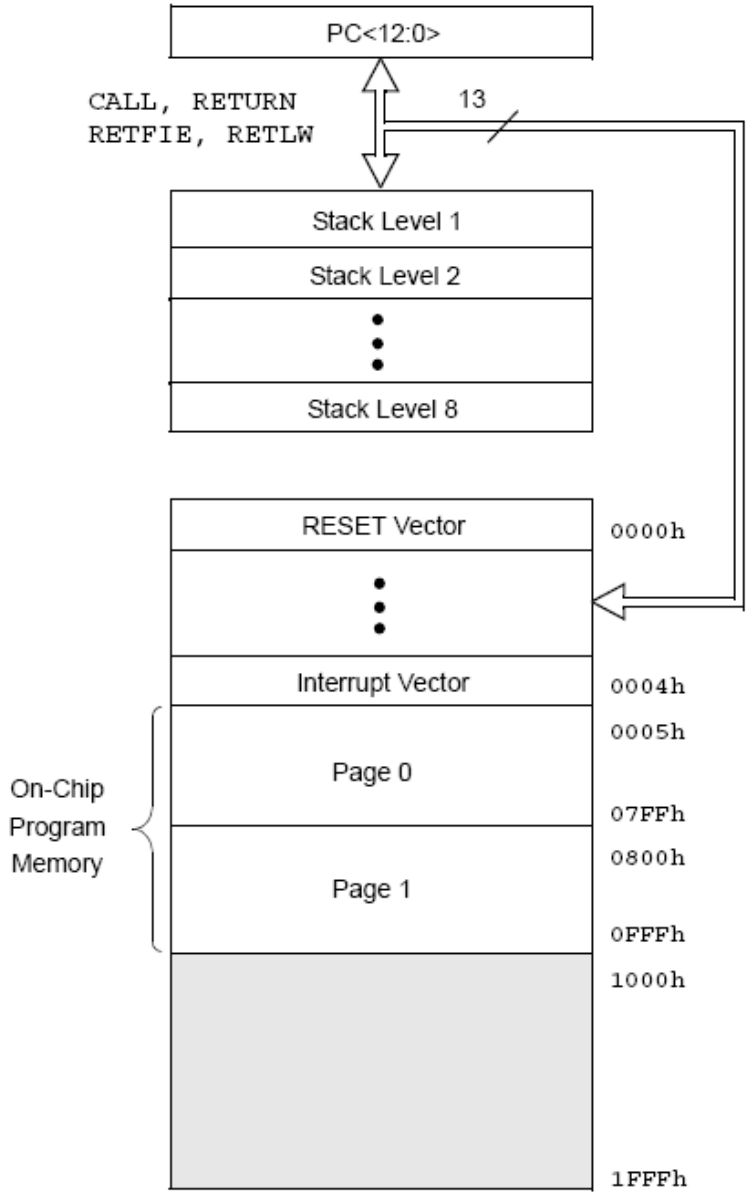
- It takes one cycle to fetch the instruction and another cycle to decode and execute the instruction
- Each instruction effectively executes in one cycle
- An instruction that causes the PC to change requires two cycles.



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

Program Memory Organization

- 13-bit program counter (PC) $\Rightarrow$ addresses for 2^{13} = 8K locations
- Each location is 14 bits (All the instructions are 14 bits long)
- PIC16F874 has only 4K (x14 bit) program memory - the upper bit is simply ignored during fetches from program memory
- Paging: 2K / page



Two Special addresses

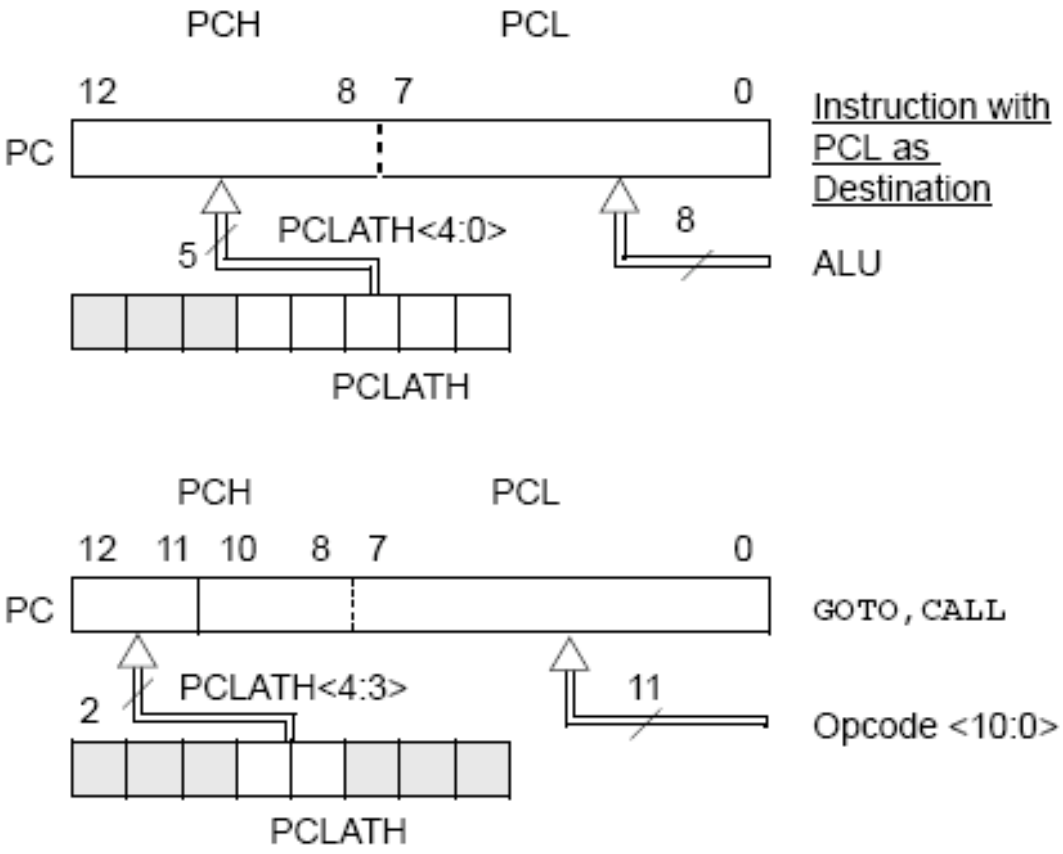
- Reset Vector Address (0000h)
  - When the CPU starts up from its reset state, its PC is automatically cleared to zero.
  - Assign ***goto Mainline*** instruction

```
Mainline
    callInitial                 ;Initialize everything
MainLoop
    call        Task1           ;Deal with task1
    callTask2                   ;Deal with task2
    …
    call        LoopTime        ;Force looptime to a fixed value
    goto        MainLoop        ;repeat
```

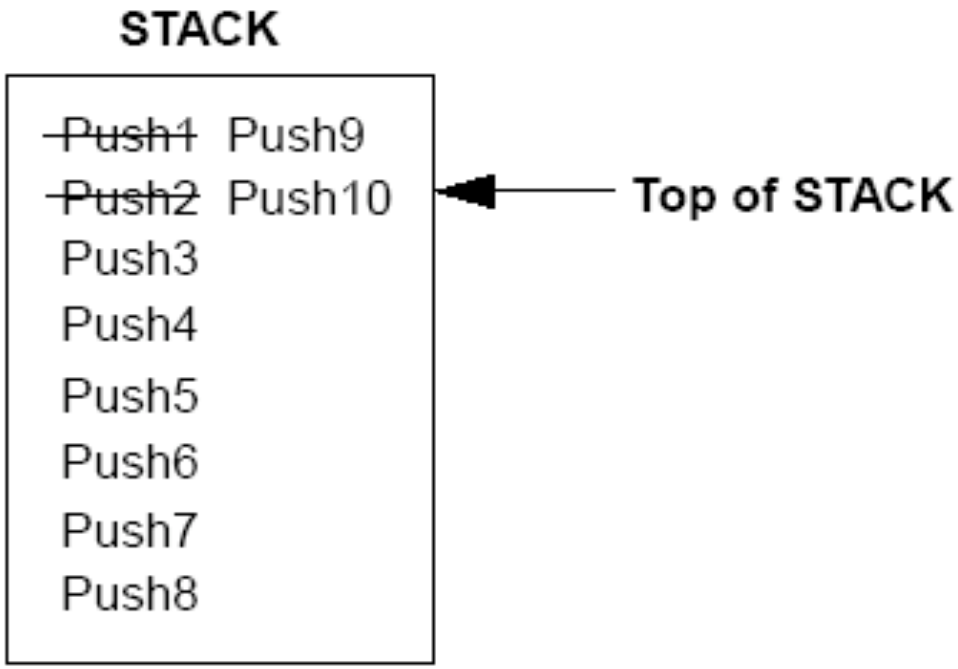
- Interrupt Vector Address (0004h)
  - 0004h is automatically loaded into the program counter when an interrupt occurs.
  - Assign ***goto IntService*** instruction there: cause the CPU to jump to the beginning of the interrupt service routine, located elsewhere in the memory space.

PCL and PCLATH registers

- PC: Program Counter, 13 bits
- PCL (02h): 8 bits, the lower 8 bits of PC
- PCLATH (0Ah): PC Latch, provides the upper 5 (or 2) bits of PC when PCL is written to



Stack



- 8 level deep x 13 bit wide hardware stack
- The stack operates as a circular buffer. After the stack has been pushed eight times, the ninth push will overwrite the value that was stored from the first push. There are no status bits to indicate stack overflow or stack underflow conditions.
- No PUSH or POP instructions. The PC is pushed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is popped when a RETURN, RETLW, or a RETFIE instruction is executed.

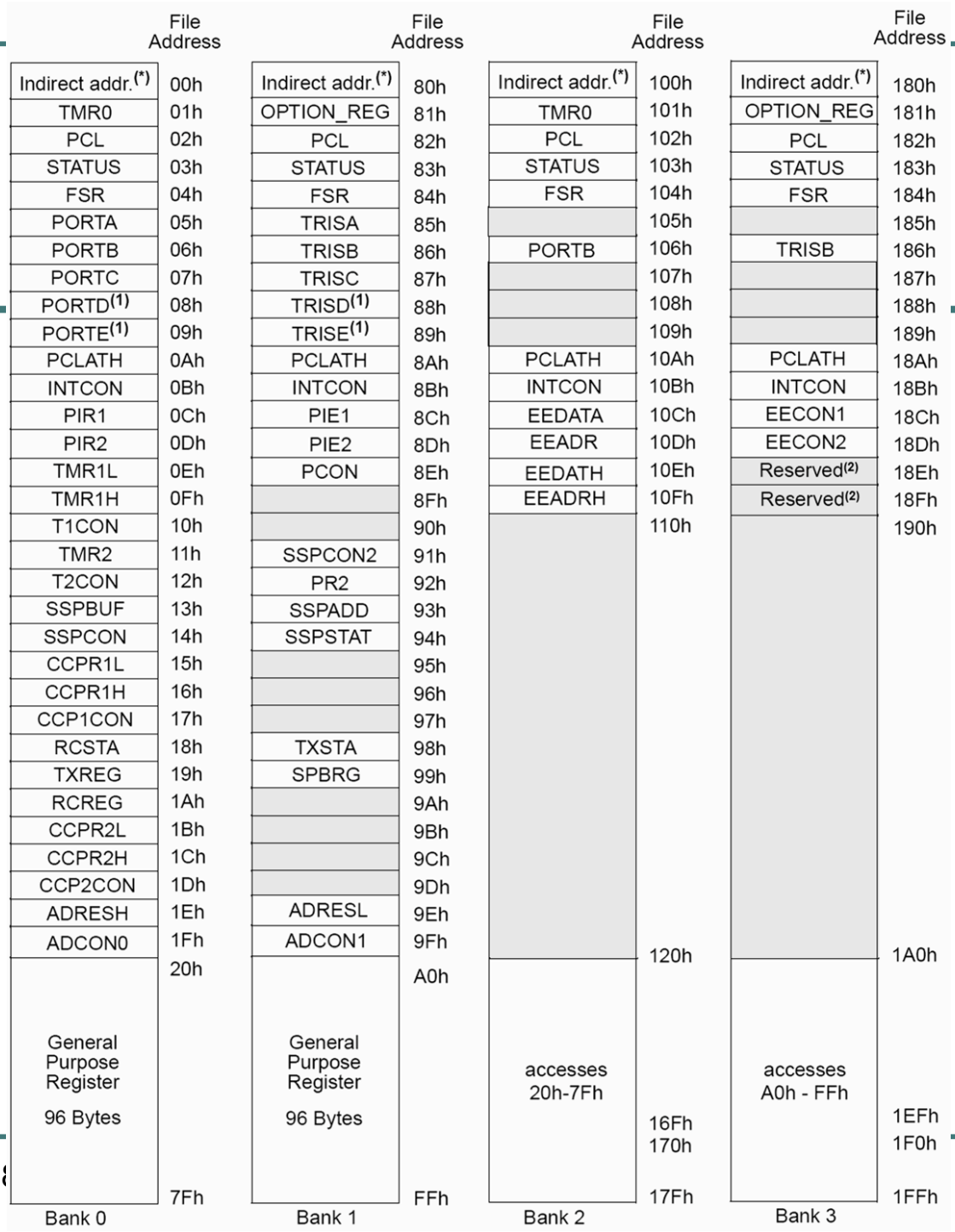
Introduction to PIC Microcontroller: Data Memory Organization

Review

- Harvard architecture
  - Separated Program memory and Data memory, separated access
  - Instruction pipelining: while the first instruction is executed (which might involves data memory access), the second one is fetched from the program memory
- Program memory
  - Up to 8K words (13 bit PC, 2^{13}=8K)
  - Each word is 14 bits
  - Each instruction is exactly 1 word long.
  - General program structure – two special addresses: Reset vector address (0000h) and Interrupt vector address (0004h)
  - 8 level deep hardware stack

Data Memory Organization

- Data memory is made up of
  - Special Function Registers (SFR) area
  - General Purpose Registers (GPR) area
- SFRs control the operation of the device, e.g.
  - I/O ports and associated control registers used to establish each bit of a port as wither an input or an output
  - Registers that provide the data input and data output to the variety of resources on the chip, such as the timers, the serial ports, and the analog-to-digital converter.
  - Registers that contain control bits for selecting the mode of operation of a chip resource as well as enabling or disabling its operation
  - Registers that contain status bits which denote the state of one of these chip resources
- GPRs are the other name for the microcontroller's RAM, are the general area for data storage and scratch pad operations
- Register File is a PIC terminology, denote memory locations that an instruction can access via an address.


File Address
Indirect addr.(*) 00h
TMR0 01h
PCL 02h
STATUS 03h
FSR 04h
PORTA 05h
PORTB 06h
PORTC 07h
PORTD(1) 08h
PORTE(1) 09h
PCLATH 0Ah
INTCON 0Bh
PIR1 0Ch
PIR2 0Dh
TMR1L 0Eh
TMR1H 0Fh
T1CON 10h
TMR2 11h
T2CON 12h
SSPBUF 13h
SSPCON 14h
CCPR1L 15h
CCPR1H 16h
CCP1CON 17h
RCSTA 18h
TXREG 19h
RCREG 1Ah
CCPR2L 1Bh
CCPR2H 1Ch
CCP2CON 1Dh
ADRESH 1Eh
ADCON0 1Fh
20h
General Purpose Register
96 Bytes
7Fh
Bank 0
File Address
Indirect addr.(*) 80h
OPTION_REG 81h
PCL 82h
STATUS 83h
FSR 84h
TRISA 85h
TRISB 86h
TRISC 87h
TRISD(1) 88h
TRISE(1) 89h
PCLATH 8Ah
INTCON 8Bh
PIE1 8Ch
PIE2 8Dh
PCON 8Eh
8Fh
90h
SSPCON2 91h
PR2 92h
SSPADD 93h
SSPSTAT 94h
95h
96h
97h
TXSTA 98h
SPBRG 99h
9Ah
9Bh
9Ch
9Dh
ADRESL 9Eh
ADCON1 9Fh
A0h
General Purpose Register
96 Bytes
FFh
Bank 1
File Address
Indirect addr.(*) 100h
TMR0 101h
PCL 102h
STATUS 103h
FSR 104h
105h
PORTB 106h
107h
108h
109h
PCLATH 10Ah
INTCON 10Bh
EEDATA 10Ch
EEADR 10Dh
EEDATH 10Eh
EEADRH 10Fh
110h
120h
accesses 20h-7Fh
16Fh
170h
17Fh
Bank 2
File Address
Indirect addr.(*) 180h
OPTION_REG 181h
PCL 182h
STATUS 183h
FSR 184h
185h
TRISB 186h
187h
188h
189h
PCLATH 18Ah
INTCON 18Bh
EECON1 18Ch
EECON2 18Dh
Reserved(2) 18Eh
Reserved(2) 18Fh
190h
1A0h
accesses A0h - FFh
1EFh
1F0h
1FFh
Bank 3

UML 16.4

Banking

- Data memory is partitioned into banks
- Each bank extends up to 7Fh (128) bytes
  - 4 banks : 4*128 bytes = 512 bytes
  - 2 banks : 2*128 bytes = 256 bytes
- Lower locations of each bank are reserved for SFRs. Above the SFRs are GPRs.
- Implemented as Static RAM
- Some “high use” SFRs from bank0 are mirrored in the other banks (e.g., INDF, PCL, STATUS, FSR, PCLATH, INTCON)
- RP0 and RP1 bits in the STATUS register selects the bank when using direct addressing mode.

Direct Addressing

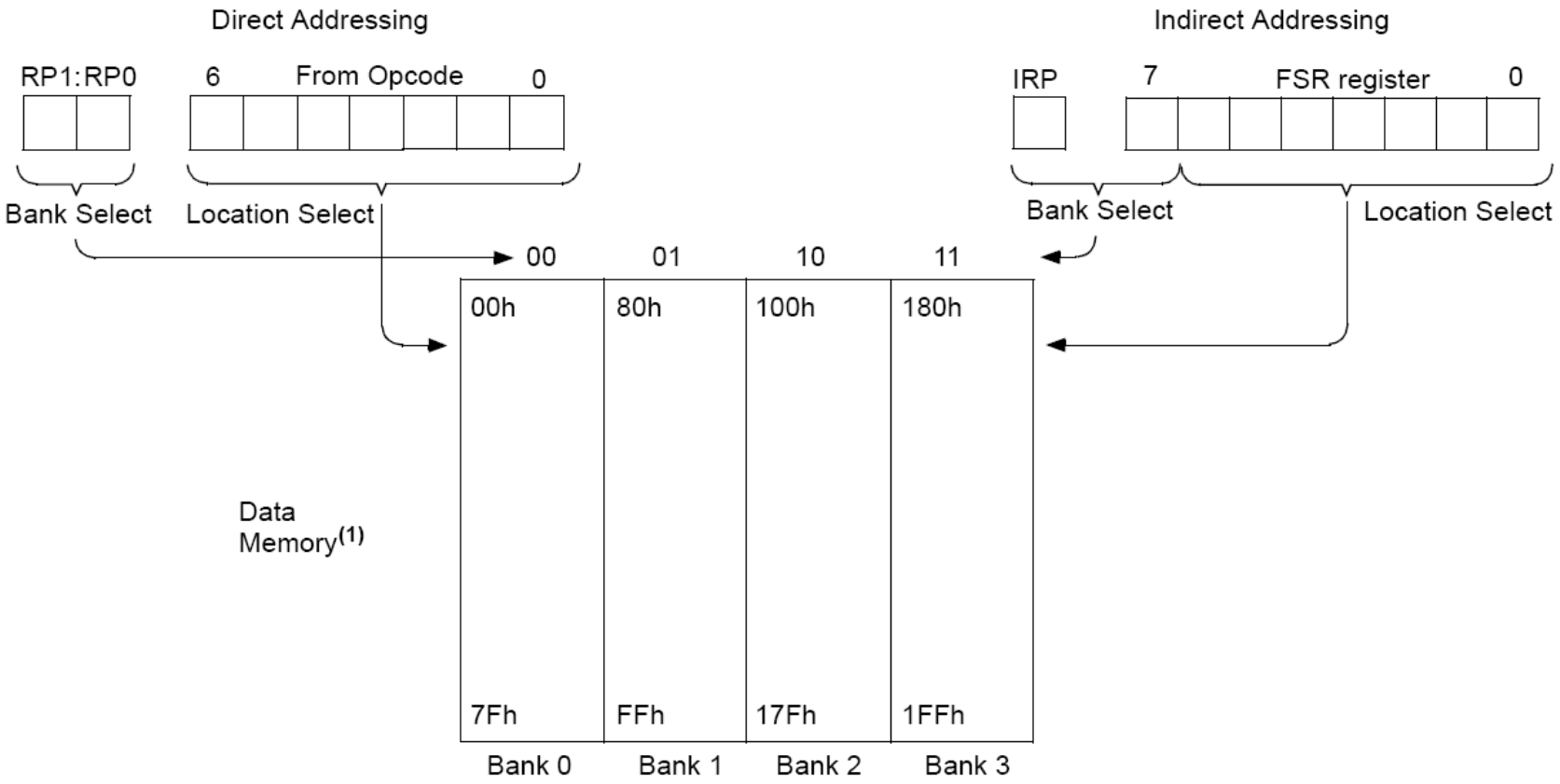
- Use only 7 bits of instruction to identify a register file address
- The other two bits of register address come from RP0 and RP1 bits in the STATUS register

Accessed Bank	Direct (RP1:RP0)	Indirect (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Example: Bank switching

```
CLRF        STATUS          ; Clear STATUS register (Bank0)
: ;
BSF         STATUS, RP0     ; Bank1
: ;
BCF         STATUS, RP0     ; Bank0
: ;
MOVLW       0x60            ; Set RP0 and RP1 in STATUS register, other
XORWF       STATUS, F       ; bits unchanged (Bank3)
: ;
BCF         STATUS, RP0     ; Bank2
: ;
BCF         STATUS, RP1     ; Bank0
```

Direct/Indirect Addressing



Indirect Addressing

- The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.
- Any instruction using the INDF register actually access the register pointed to by the File Select Register (FSR).
- The effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit in STATUS register.

Example

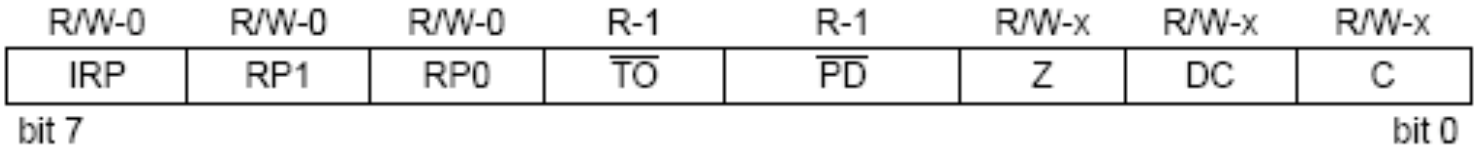
```
        MOVLW   0x20    ;initialize pointer
        MOVWF   FSR     ;to RAM
NEXT    CLRF    INDF    ;clear INDF register
        INCF    FSR,F   ;inc pointer
        BTFSS   FSR,4   ;all done?  (to 0x2F)
        GOTO    NEXT    ;no clear next
CONTINUE
        :               ;yes continue
```

Special Function Registers (1)

- W, the working register
  - To move values from one register to another register, the value must pass through the W register.
- FSR (04h,84h,104h,184h), File Select Register
  - Indirect data memory addressing pointer
- INDF (00h,80h,100h,180h)
  - accessing INDF accesses the location pointed by IRP+FSR
- PC, the Program Counter, PCL (02h, 82h, 102h, 182h) and PCLATH (0Ah, 8Ah, 10Ah, 18Ah)

Special Function Registers (2)

- STATUS (03h, 83h, 103h, 183h)



  - IRP: Register bank select bit (indirect addressing)
  - RP1:RP0 – Register bank select bits (direct addressing)
  - NOT_TO: Time Out bit, reset status bit
  - NOT_PD: Powel-Down bit, reset status bit
  - Z: Zero bit ~ ZF in x86
  - DC: Digital Carry bit ~ AF in x86
  - C: Carry bit ~ CF in x86 (note: for subtraction, borrow is opposite)

I/O Ports

- General I/O pins are the simplest of peripherals used to monitor and control other devices.
- 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 PORTD (PORTD is an 8-bit port. Each pin is individually configurable as an input or output).

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