PIC 18F452 Architecture

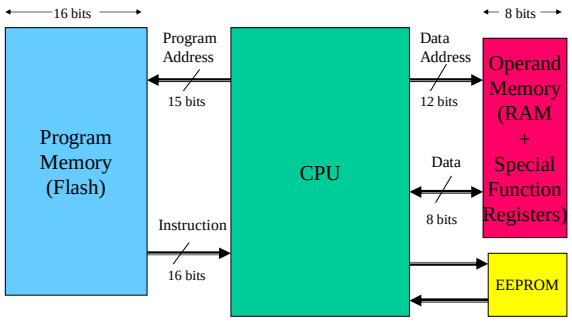
55:036

Embedded Systems and System Software

Harvard Architecture

- The PIC Processor has a *Harvard* architecture –i.e. separate instruction memory and data memory.
- For the 18F452:
  - 32KByte Program Memory on chip, (Flash)
  - 1792 bytes of Data Memory on chip
    - 1536 bytes of static RAM
    - 256 bytes of EEPROM
- Data RAM serves the role of registers and main memory
  - i.e. there is no distinction between data accesses to registers and memory
  - Can think of the processor as having lots of registers and no additional data memory

18F452 Architecture



PIC Program Memory

- 2^{15} = 32K = 32768 program memory locations (bytes).
- Most PIC instructions occupy two bytes (a few occupy 4 bytes)
- CPR can read two bytes at a time from Program memory
- This is non-volatile *flash* memory in the 18F452.
  - Can be read (in 8 or 16 bit units) at processor speed
  - Limited write/erase ability

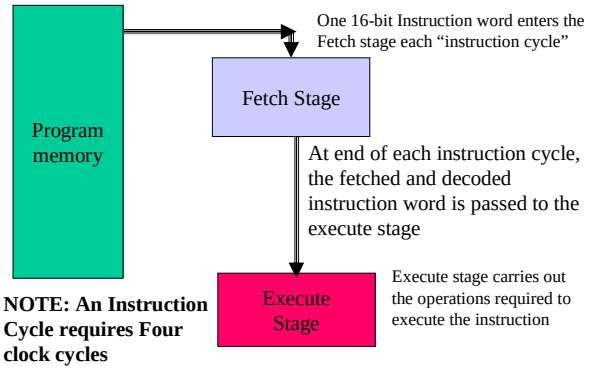
PIC (RAM) Data Memory

- 12 data address bits allow for 2^{12} = 4K = 4096 data memory locations (bytes).
  - PIC 18F452 implements only 1536 bytes of data memory
- This is volatile *static RAM* memory which can both be read and written.
- The non-volatile EEPROM data memory is accessed via a separate mechanism.
  - EEPROM can be read at near processor speed
  - Can be written to in byte units
  - Write times are MUCH slower (measured in msec)

PIC Two-Stage Pipeline

- The two stages of the PIC pipeline run simultaneously to improve speed.
- The *instruction fetch* stage gets the next instruction machine code from program memory.
- The *execution* stage does whatever the machine code calls for.

Two-stage Pipeline



NOTE: An Instruction Cycle requires Four clock cycles

Standard Pipeline Operation

Cycle:	1	2	3	
	fetch n	fetch n+1	fetch n+2	
	exec. n-1	exec. n	exec. n+1	

Note: Net instruction "throughput of 1 instr./cycle

Standard Pipeline Operation

Note: these are Instruction Cycles (= 4 clock cycles)

Cycle:	1	2	3	
	fetch n	fetch n+1	fetch n+2	
	exec. n-1	exec. n	exec. n+1	

Note: Net instruction "throughput of 1 instr./cycle

Two-Word Instructions

- Most PIC instructions have a one-word (16-bit) machine code.
- A few PIC instructions have a two-word (32-bit) machine code.
- Two-word instructions require two fetches from two consecutive addresses.
- Two-word instructions therefore always require two instruction clocks to process.

Pipeline operation if Instruction n requires two words

Cycle	1	2	3	4	
	fetch n (word 1)	fetch n (word 2)	fetch n+1	fetch n+2	
	exec. n-1		exec. n	exec. n+1	

Note: Each two word instruction "costs" one extra cycle

Branch Instruction Timing

- Branch instructions are those that change the program counter (PC).
- Since the fetch stage incorrectly gets the machine code for the next sequential instruction when branching, this machine code must be dumped.
- This dumping results in taken branches (including unconditional branches) requiring two instruction clocks to process.
- Branches that are not taken require only one machine cycle.

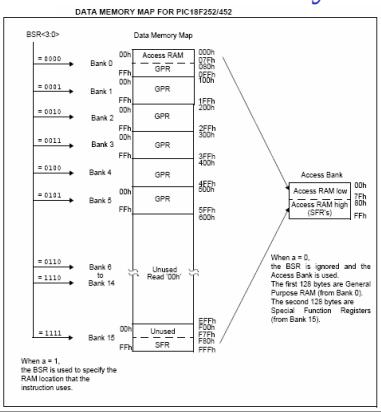
Pipeline operation if Instruction n is a "taken" branch instruction

	1	2	3	4	
	fetch n	fetch n+1 (dumped)	fetch bt	fetch bt+1	
	exec. n-1	exec n		exec. bt	

bt = branch target address

Note: Each "taken" branch instruction "costs" one extra cycle

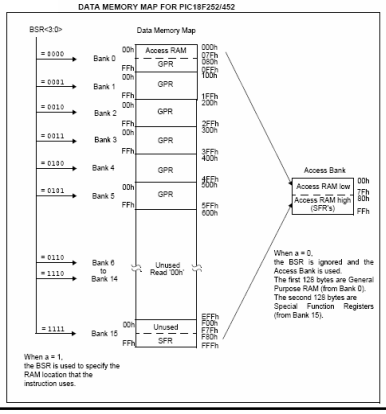
18F452 Data Memory



Instruction Format

- Most PIC instructions employ a one operand format
  - Like an "accumulator" architecture
- There is a special "working register" WREG (or just W) that is used as the second operand for many instructions that require two operands.
  - Unlike a traditional accumulator architecture, the WREG is not necessarily the default destination operand

18F452 Data Memory



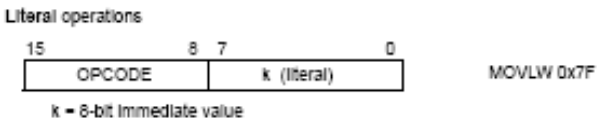
Data Addressing Modes

- *Literal* (immediate) addressing: The data is contained in the machine code instruction.
- *Direct* addressing: The address of the data is contained in the machine code instruction.
- *Indirect* addressing: The address of the data is contained in a special indirect addressing register.

Literal Addressing

- Instructions using a literal (immediate) value mostly operate on the WREG for the second operand.
- Since the 8-bit literal value is part of the machine code there is no room to also have an 8-bit register number.
- Example: `addlw 7` ; WREG←WREG+7

Instruction Format--Literal Addressing



Direct Addressing of Data Memory (Registers)

- Data memory addresses are 12 bits
- Instructions only have room for 8 bit addresses
- Remaining address bits can be supplied by a special Bank Select Register (BSR)

Access Bank Direct Addressing

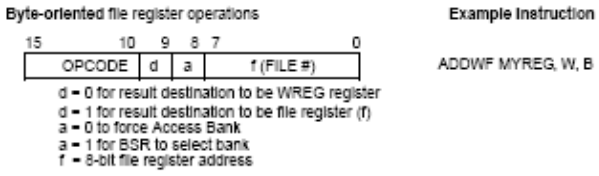
- When accessing an address (register) in the range 0x000-0x07F or any special function register (SFR), unbanked addressing (also called **"access bank"** addressing is automatically used by the assembler.
- BSR is not used—Upper 4 bits of address assumed to be 0
- Example: clrf 0x050 ; reg[0x050]←0x00

Banked Direct Addressing

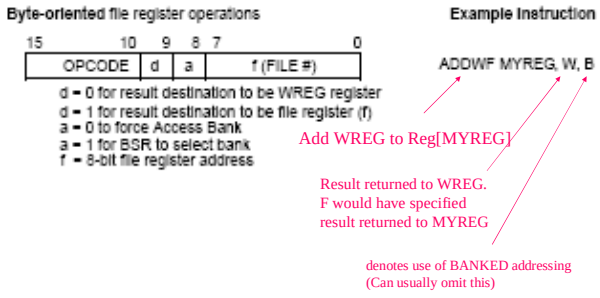
- When accessing registers 0x800-0xF7F, banked addressing is automatically used.
- Banked addressing combines the lower nibble of the *bank select register* (BSR) with the least significant 8 bits of the register specified.
- Ex:

```
movlb 5      ; BSR ← 5
clrf 0x29B   ; reg[BSR+0x09B]←0x00
             ; reg[0x59B] ←0x00
```

Instruction Format—Direct Addressing



Instruction Format—Direct Addressing



Special Function Registers (SFRs)

- 18F452 contains a large number of Special Function registers (SFRs)
  - Used to configure the processor and configure/control the operation of various on-chip peripheral modules
- SFRs are implemented and addressed as Data RAM
  - Although they are physically distributed among the modules they control

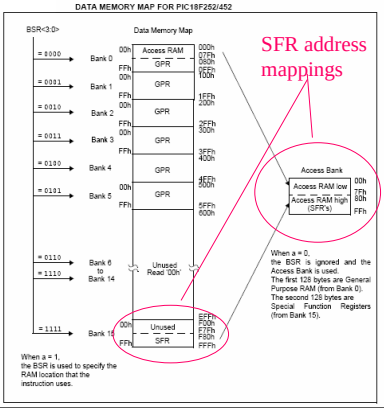
PIC 18F452 SFRs

TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F2X2 devices.

18F452 Data Memory



SFR Direct Addressing

- Special function registers are double mapped to addresses 0x080-0x0FF (unbanked only) and 0xFFF-0xFFF.
- SFRs are normally referred to by their *symbolic name*. If this is done, there is no need to worry about the BSR for SFR access.
- Ex: `clrf PORTB` ; reg[PORTB]←0x00

0x080-0x0FF Addressing

- Banked addresses 0x080-0x0FF are not the SFRs, they are general purpose registers.
- On power-up, the PIC has BSR=0x00, so access to these addresses will occur correctly (automatically).
- Access to addresses 0x100-0xF7F requires first setting the BSR.

Available Addresses

- The 18F452 has 1536 bytes of general-purpose RAM (6 banks of 256 bytes).
- The available addresses are therefore 0x000-0x5FF plus 128 SFR addresses.
- The cheaper 18F442 part has 768 bytes of RAM (3 banks of 256 bytes) so addresses 0x000-0x2FF plus SFRs are available.

Indirect Addressing

- Indirect addressing accesses the register whose address is in one of three FSRs (F select registers) called FSR0, FSR1, and FSR2.
  - These registers are (logicaly) 12 bits in size so that they can store a complete 12-bit register file address
- Variants of indirect addressing allow for automatically incrementing, decrementing, or adding WREG to the FSR with the access.

Plain Indirect Addressing

- To leave the FSR register unchanged, use INDF0, INDF1, or INDF2 for indirect access using FSR0, FSR1, or FSR2 respectively.
- Ex:
```
lfsr 2, 0x3F2   ; reg[FSR2]←0x3F2
clrf INDF2      ; reg[0x3F2]←0x00
```
Note that FSR2 is unchanged by the clrf instruction.

Post-decrement Addressing

- To decrement the FSR after the access, use POSTDEC0, POSTDEC1, or POSTDEC2.
- Ex: lfsr 2, 0x3F2 ; reg[FSR2]←0x3F2
  clrf POSTDEC2 ; reg[0x3F2]←0x00

  After 0x3F2 is cleared, FSR2 contains 0x3F1.

Post-increment Addressing

- To increment the FSR after the access, use POSTINC0, POSTINC1, or POSTINC2.
- Ex: lfsr 0, 0x1BA ; reg[FSR0]←0x1BA
  clrf POSTINC0 ; reg[0x1BA]←0x00

  After 0x1BA is cleared, FSR0 contains 0x1BB.

Pre-increment Addressing

- To increment the FSR before the access, use PREINC0, PEINC1, or PREINC2.
- Ex: lfsr 1, 0x22C ; reg[FSR1]←0x22C
  clrf PREINC1 ; reg[0x22D]←0x00

  FSR1 is incremented to 0x22D and then the clear is executed.

Pre-add-W Addressing

- To add W to the FSR before the access, use PLUSW0, PLUSW1, or PLUSW2.
- Ex: lfsr 1, 0x22C ; reg[FSR1]←0x22C
  movlw 3 ; reg[WREG]←0x03
  clrf PLUSW1 ; reg[0x22F]←0x00

  This addition is temporary. The FSR value remains unchanged after the access.

Table Reading

- Bytes from **program memory** can be copied into the TABLAT special function register.
- This permits read-only (constant) data structures, such as look-up tables, to be kept in program memory.
- The program memory address of the byte to be read is specified by the TBLPTRH and TBLPTRL (table pointer) SFRs.
- The four table read instructions allow for various auto-increment/decrement combinations.

Table Read Instructions

- tblrd* : plain table read
- tblrd*+ : table read, then increment
- tblrd*- : table read, then decrement
- tblrd+* : increment, then table read

Special Program Addresses

- Reset vector: 0x0000
- High-priority interrupt vector: 0x0008
- Low-priority interrupt vector: 0x0018

- More about this later

Status Register

- The SFR called STATUS has 5 active bits and 3 unimplemented bits.
- The active bits are called N, OV, Z, DC, and C.
- These bits are altered by some instructions. Inside front cover of book shows which instructions alter what.

Carry/Borrow Bit

- Addition: C=0 if no carry, C=1 if carry
- Subtraction: C=1 if no borrow, C=0 borrow
- Rotate through carry: C is part of 9-bit rotate involving a register and the carry flag.
- The C bit normally makes sense when doing unsigned arithmetic.

Overflow Bit

- OV=0 if the result is between -128 and +127, otherwise OV=1.
- The OV bit normally makes sense when doing signed arithmetic.

Zero Bit and Negative Bit

- Z=1 if all 8 bits of a result are binary 0, otherwise Z=0.
- N=1 if the result has a binary 1 in the most significant bit, otherwise N=0. This flag normally makes sense when doing signed arithmetic.

Decimal Carry Bit

- The DC bit is normally used when doing packed binary-code decimal (BCD) arithmetic.
- The DC bit is a carry/borrow bit (like C) except that the DC bit deals with carry/borrow from the least-significant nibble rather than the whole byte.