LSN 2 – Processor Architecture Review

- **Harvard**
  - Separate data and program memory space (busses)
- **Von-Neumann**
  - Only one bus between CPU and memory

![Harvard Architecture Diagram](image1)

![Von Neumann Architecture Diagram](image2)
LSN 2 – Processor Architecture Review

• RISC
  – A minimal set of simple instructions when combined can accomplish every needed operation

• CISC
  – A large set of complex instructions can singularly provide all needed operations
LSN 2 – Processor Architecture Review

- Instruction Cycle

- Registers
  - Special purpose (PC)
  - General purpose
LSN 2 – Microchip’s MCU Families

- 10-12 MIPS
- Up to 128 KB Program Flash
- 18–100 Pins
LSN 2 – PIC18 Families

Traditional PIC18

PIC18 J-series

PIC18 K-series

Typically products with higher memory also have higher pin-counts and higher levels of integrated peripherals

- **Traditional PIC18**
  - 40 MHz, 10 MIPS, 5V
  - Flash endurance 100k
  - EEPROM
  - Premium Features

- **PIC18 J-series**
  - 40-48 MHz, 10-12 MIPS, 3V
  - Flash endurance 1k – 10k
  - Emulate EEPROM
  - Most cost effective >32KB Flash

- **PIC18 K-series**
  - 64MHz, 16 MIPS, 3V
  - Flash endurance 10k
  - EEPROM
  - Most cost effective <32KB Flash
LSN 2 – 8-bit PIC® Architecture

Program Space

Program Flash

21-bit

Program Bus

16-bit

Table Access

12-bit

Data Bus

8-bit

8-bit CPU

Data Space

Data RAM

Peripherals

I/O Ports

sschneider@udayton.edu
LSN 2 – PIC18 Program Memory Map

21-bit Program Counter

Stack Level 1
Stack Level 2
Stack Level 30
Stack Level 31

31 Level Stack

On-chip Program Memory

Reset Vector
High Priority Interrupt Vector
Low Priority Interrupt Vector

Unimplemented Program Memory (Read as ‘0’)
LSN 2 – PIC18 Data Memory Map

PIC18F
Register File Map

ACCESS RAM
Bank 0 GPR

ACCESS RAM
Bank 1 GPR

ACCESS RAM
Bank 2 GPR

ACCESS RAM
Bank 13 GPR

ACCESS RAM
Bank 14 GPR

ACCESS RAM
Bank 15 GPR

ACCESS SFR

ACCESS SFR

ACCESS SFR

sschneider@udayton.edu
# LSN 2 – PIC18 Registers

<table>
<thead>
<tr>
<th>address</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFF</td>
<td>TOSU</td>
<td>Top of stack (upper)</td>
</tr>
<tr>
<td>0xFFE</td>
<td>TOSH</td>
<td>Top of stack (high)</td>
</tr>
<tr>
<td>0xFFD</td>
<td>TOSL</td>
<td>Top of stack (low)</td>
</tr>
<tr>
<td>0xFFC</td>
<td>STKPTR</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>0xFFB</td>
<td>PCLATU</td>
<td>Upper program counter latch</td>
</tr>
<tr>
<td>0xFFA</td>
<td>PCLATH</td>
<td>High program counter latch</td>
</tr>
<tr>
<td>0xFF9</td>
<td>PCL</td>
<td>Program counter low byte</td>
</tr>
<tr>
<td>0xFF8</td>
<td>TBLPTRU</td>
<td>Table pointer upper byte</td>
</tr>
<tr>
<td>0xFF7</td>
<td>TBLPTRH</td>
<td>Table pointer high byte</td>
</tr>
<tr>
<td>0xFF6</td>
<td>TBLPTRL</td>
<td>Table pointer low byte</td>
</tr>
<tr>
<td>0xFF5</td>
<td>TABLAT</td>
<td>Table latch</td>
</tr>
<tr>
<td>0xFF4</td>
<td>PRODH</td>
<td>High product register</td>
</tr>
<tr>
<td>0xFF3</td>
<td>PRODL</td>
<td>Low product register</td>
</tr>
<tr>
<td>0xFF2</td>
<td>INTCON</td>
<td>Interrupt control register</td>
</tr>
<tr>
<td>0xFF1</td>
<td>INTCON2</td>
<td>Interrupt control register 2</td>
</tr>
<tr>
<td>0xFF0</td>
<td>INTCON3</td>
<td>Interrupt control register 3</td>
</tr>
<tr>
<td>0xFEF</td>
<td>INDF0 (1)</td>
<td>Indirect file register pointer 0</td>
</tr>
<tr>
<td>0xFEE</td>
<td>POSTINC0 (1)</td>
<td>Post increment pointer 0 (to GPRs)</td>
</tr>
<tr>
<td>0xFE0</td>
<td>POSTDEC0 (1)</td>
<td>Post decrement pointer 0 (to GPRs)</td>
</tr>
<tr>
<td>0xFE8</td>
<td>PREINC0 (1)</td>
<td>Pre increment pointer 0 (to GPRs)</td>
</tr>
<tr>
<td>0xFE7</td>
<td>PLUSW0 (1)</td>
<td>Add WREG to FSR0</td>
</tr>
</tbody>
</table>

Note 1. This is not a physical register

<table>
<thead>
<tr>
<th>address</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE8</td>
<td>WREG</td>
<td>Working register</td>
</tr>
<tr>
<td>0xFE7</td>
<td>INDF1 (1)</td>
<td>Indirect file register pointer 1</td>
</tr>
<tr>
<td>0xFE6</td>
<td>POSTINC1 (1)</td>
<td>Post increment pointer 1 (to GPRs)</td>
</tr>
<tr>
<td>0xFE5</td>
<td>POSTDEC1 (1)</td>
<td>Post decrement pointer 1 (to GPRs)</td>
</tr>
<tr>
<td>0xFE4</td>
<td>PREINC1 (1)</td>
<td>Pre increment pointer 1 (to GPRs)</td>
</tr>
<tr>
<td>0xFE3</td>
<td>PLUSW1 (1)</td>
<td>Add WREG to FSR1</td>
</tr>
<tr>
<td>0xFE2</td>
<td>FSR1H</td>
<td>File select register 1 high byte</td>
</tr>
<tr>
<td>0xFE1</td>
<td>FSR1L</td>
<td>File select register 1 low byte</td>
</tr>
<tr>
<td>0xFE0</td>
<td>BSR</td>
<td>Bank select register</td>
</tr>
<tr>
<td>0xFD9</td>
<td>INDF2 (1)</td>
<td>Indirect file register pointer 2</td>
</tr>
<tr>
<td>0xFD8</td>
<td>POSTINC2 (1)</td>
<td>Post increment pointer 2 (to GPRs)</td>
</tr>
<tr>
<td>0xFD7</td>
<td>POSTDEC2 (1)</td>
<td>Post decrement pointer 2 (to GPRs)</td>
</tr>
<tr>
<td>0xFD6</td>
<td>PREINC2 (1)</td>
<td>Pre increment pointer 2 (to GPRs)</td>
</tr>
<tr>
<td>0xFD5</td>
<td>PLUSW2 (1)</td>
<td>Add WREG to FSR2</td>
</tr>
<tr>
<td>0xFD4</td>
<td>FSR2H</td>
<td>File select register 2 high byte</td>
</tr>
<tr>
<td>0xFD3</td>
<td>FSR2L</td>
<td>File select register 2 low byte</td>
</tr>
<tr>
<td>0xFD2</td>
<td>STATUS</td>
<td>Status register</td>
</tr>
</tbody>
</table>

sschneider@udayton.edu
LSN 2 – PIC18 Registers

- Status register
  - Contains arithmetic status of the ALU
  - Bits set or cleared according to the device logic

```
<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>--</td>
<td>--</td>
<td>N</td>
<td>OV</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
</tr>
</tbody>
</table>
```
LSN 2 – PIC18 Programmer’s Model
LSN 2 – PIC18 Pipelining

- Allows processor to overlap the execution of several instruction to achieve higher instruction throughput
  - Utilizes the fact that different processor components are not fully utilized during the instruction execution process
  - Prefetches instructions during execution of current instructions

![Pipelining Diagram]

sschneider@udayton.edu
LSN 2 – PIC18 Pipelining

- PIC18 utilizes a two-stage pipeline for instruction fetch and instruction execution

- Problems can arise from pipelining (Hazards)
  - Data dependency hazard
  - Control hazard
  - Beyond scope of this class
LSN 2 – PIC18 Instruction Format

Decoded Instruction from Program Memory:
- Opcode
- d
- a
- Address

Data Memory (Register File):
- 07h
- 08h
- 09h
- 0Ah
- 0Bh
- 0Ch
- 0Dh
- 0Eh
- 0Fh
- 10h

Arithmetic/Logic Function to be Performed

Result Destination
LSN 2 – PIC18 Instruction Format

- Byte oriented file register instructions

```
15 10 9 8 7 0

| opcode | d | a | f |
```

- d = 0 for result destination to be WREG register.
- d = 1 for result destination to be file register (f)
- a = 0 to force Access Bank
- a = 1 for BSR to select bank
- f = 8-bit file register address

- Byte-to-byte move operations (2 words)

```
15 12 11 0

| opcode | f (source file register) |
```

```
15 12 11 0

| 1111 | f (destination file register) |
```

f = 12-bit file register address
LSN 2 – PIC18 Instruction Format

- **Bit-oriented file register operations**

  ```
  15 12 11 9 8 7 0
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
  f
  ```

  - b = 3-bit position of bit in the file register (f).
  - a = 0 to force Access Bank
  - a = 1 for BSR to select bank
  - f = 8-bit file register address

- **Literal operations**

  ```
  15 8 7 0
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>k</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
  ```

  - k = 8-bit immediate value
LSN 2 – PIC18 Instruction Format

- Control operations

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Control operations</td>
</tr>
<tr>
<td>0x1111</td>
<td>GOTO label</td>
</tr>
<tr>
<td>0x00</td>
<td>CALL funct_name</td>
</tr>
<tr>
<td>0x00</td>
<td>BRA label</td>
</tr>
<tr>
<td>0x00</td>
<td>BC label</td>
</tr>
</tbody>
</table>

- n = 20-bit immediate value
- S = fast bit
LSN 2 – PIC18 Instruction Format

• Mnemonic notation
  – ‘F’ or ‘W’ indicate that the source or destination address is the original register file location (F) or the working register (W)

• Number formats
  – Hex 0x7f, 20 (default), H’7f’ (alternative)
  – Binary B’10011100’
  – Decimal D’32’
  – Octal O’777’
  – ASCII A’C’, ‘C’ (alternative)
LSN 2 – PIC18 Instruction Format

- Instruction formats presented use 8-bits to specify a register file (f field)
  - Uses BSR to select only one bank at a time
  - When operating on a data register in a different bank, bank switching is needed
- Access Register
  - When operands are in the access bank, no bank switching is needed
  - Most SFRs are in the access bank
LSN 2 – PIC18 Addressing Modes

- All MCUs use addressing modes to specify the operand to be operated on

- Register direct mode
  - Use an 8-bit value to specify a data register
    - MOVWF 0x25, A
    - MOVFF 0x40, 0x50

- Immediate mode
  - Actual operand provided, no need to access memory
    - MOVILW 0x25
    - ANDLW 0x40
LSN 2 – PIC18 Addressing Modes

• Inherent mode
  – Operand is implied in the opcode field, opcode does not provide address

• Indirect mode
  – A special function register (FSRx) is used as a pointer to the actual data register

  LFSR  FSR0, 0x25
  MOVWF  INDF0
  MOVWF  PREINC0
LSN 2 – PIC18 Addressing Modes

• Bit-direct mode
  – Five instructions to deal with bits (BCF, BSF, BTFSC, BTFSS, BTG)
    \[
    \text{BTG PORTB, 2}
    \]
LSN 2 – PIC18 Instructions

• PIC18 has 77 instructions

• Data movement instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>16-bit instruction word</th>
<th>Status affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>lfsr f, k</td>
<td>Load FSR</td>
<td>1110 1110 00ff k_{1,3}</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1111 0000 k_{2,3} kkk</td>
<td></td>
</tr>
<tr>
<td>movf f, d, a</td>
<td>Move f</td>
<td>0101 00da fff fff fff</td>
<td>Z, N</td>
</tr>
<tr>
<td>movf fs, fd</td>
<td>Move fs (source) to fd</td>
<td>1100 fff fff fff fff</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1111 fff fff fff fff</td>
<td></td>
</tr>
<tr>
<td>movwf f, a</td>
<td>Move WREG to f</td>
<td>0110 111a fff fff fff</td>
<td>None</td>
</tr>
<tr>
<td>swapf f, d, a</td>
<td>Swap nibbles in f</td>
<td>0011 10da fff fff fff</td>
<td>None</td>
</tr>
<tr>
<td>movlw k</td>
<td>Move literal to BSR&lt;3:0&gt;</td>
<td>0000 0001 kkkk kkkk</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Move literal to WREG</td>
<td>0000 1110 kkkk kkkk</td>
<td>None</td>
</tr>
</tbody>
</table>
LSN 2 – PIC18 Instructions

• Add instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addwf f, d, a</td>
<td>Add WREG and f</td>
</tr>
<tr>
<td>addwfc f, d, a</td>
<td>Add WREG and carry bit to f</td>
</tr>
<tr>
<td>addlw k</td>
<td>Add literal and WREG</td>
</tr>
</tbody>
</table>

• Subtraction instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subfwb f, d, a</td>
<td>Subtract f from WREG with borrow</td>
</tr>
<tr>
<td>subwf f, d, a</td>
<td>Subtract WREG from f</td>
</tr>
<tr>
<td>suwfb f, d, a</td>
<td>Subtract WREG from f with borrow</td>
</tr>
<tr>
<td>sublw k</td>
<td>Subtract WREG from literal</td>
</tr>
</tbody>
</table>
LSN 2 – Homework

• Reading
  – 1.4 – 1.10

• Assignment – HW1
  – E1.4, E1.5, E1.6, E1.7, E1.11, E1.12, E1.15, and E1.19

• References
  – Microchip MCU2121 Course Notes