Microprocessor Interfacing & Programming

LECTURE 7&8

Assembly Programming

- **Terms to discuss:**

  Assembly language

  Assembler

  Machine code (object code or opcode)

- **Structure of Assembly language:** Its instruction consists of mnemonic (short, human readable word that represents a machine instruction), optionally followed by one or two operands.

Operands are the data items being manipulated and mnemonics are the commands to the processor.

Assembly language instruction

4 fields:

[label] mnemonic [operands] [;comment]

Brackets indicate that a field is optional and not all lines have them.

See the rules for label.

- The mnemonic (instruction) and operand(s) fields together perform the real work of the program.

```
MOVLW   55H
ADDLW   67H
```

ADDLW and MOVLW are mnemonics that produce opcodes.

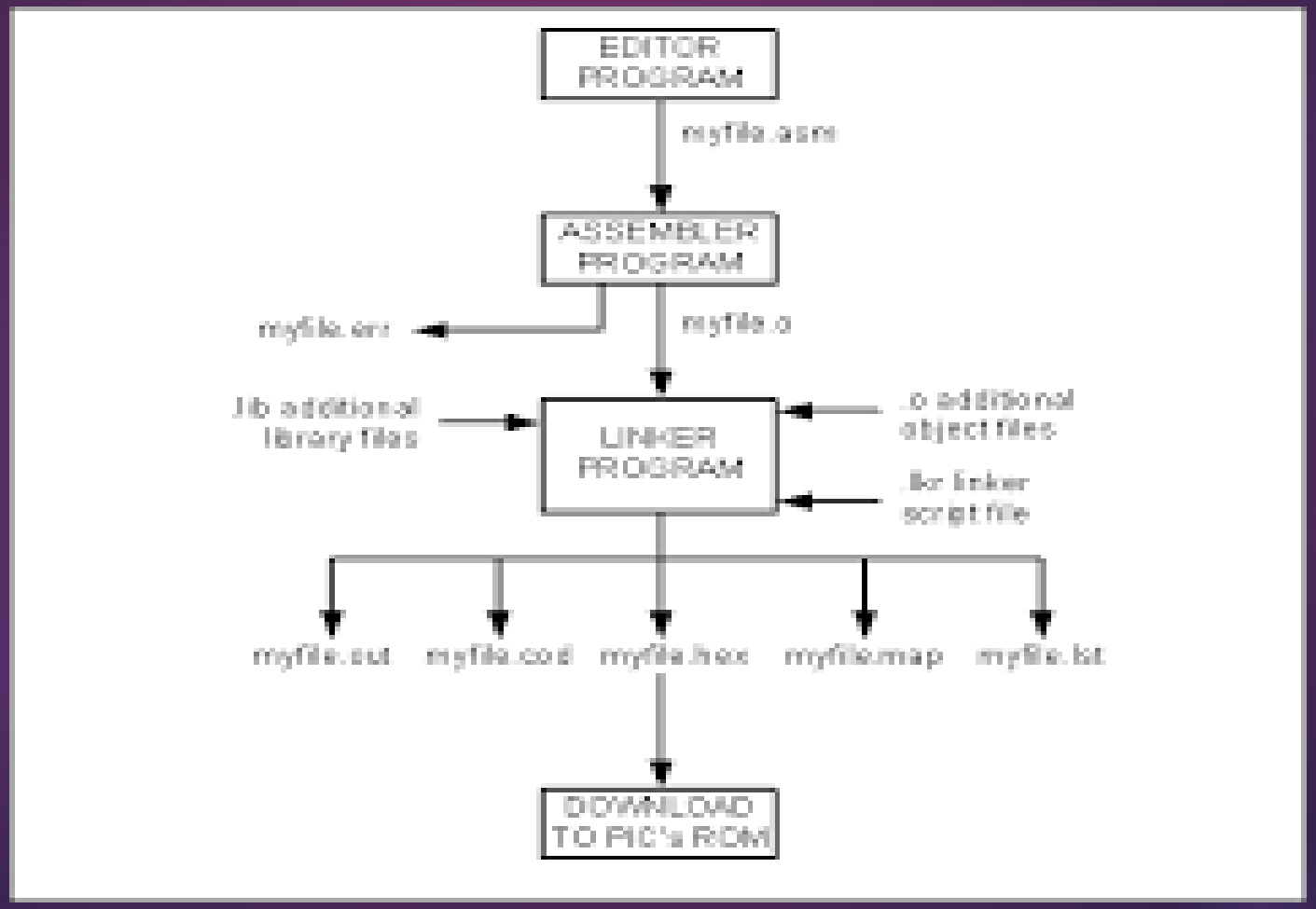
55H and 67H are the operands.

Example:

- Directives (pseudo-instructions) do not generate any machine code (opcode) and are used only by the assembler.

```
SUM     EQU   10H
        ORG     0H
        MOVLW  25H
        ADDLW  0x34
        ADDLW  11H
        ADDLW   D'18'
        ADDLW  1CH
        ADDLW   B'00000110'
        MOVWF  SUM
        HERE  GOTO  HERE
END
```

Assembling and Linking a PIC program



- asm file is also called source file. It is created with a text editor.
- In modular programming, linker links many object files to create a ready-to-burn hex file.
- Before linking, the file must be error free.
- lst (list) file shows the binary and source code.
- map files shows the memory layout of used and unused memory locations.

lst file

- lst file is a human-readable text file that shows:
  - Program memory address of each instruction
  - The generated opcode (in hex)
  - The original source code line

Refer to book to see the sample of lst file.

Program Counter (PC) in PIC

- It is a register used by the processor to point to the address of the next instruction to be executed.
- As, processor fetches the opcode from ROM, PC is incremented automatically.
- The wider the PC, more the memory locations of processor can access. i.e; 14 bit PC can acces maximum of (2^14=16K) of code.
- 8051 have a 16 bit program counter.
- In PIC18, PC is 21 bit wide. It means it can access program addresses 000000 to 1FFFFFH. A total of (2^21=2M) of code.

- Data registers(SFRs+GPRs) = 8 bits because PIC is an 8-bit architecture (ALU works with 8-bit data).

- Program Counter = wider because it must hold the address of instructions in program memory.

Program Counter width depends on program memory size.

ROM memory map in PIC

- PC = 21 bits in all PIC18 → supports addressing up to 2 MB.
- PIC18F8722 only implements 128 KB Flash, so only the lower part of the PC space is used.
- See example 2.11

Contd....

- In PIC18F8722, PC = 21 bits, but device only has 128K words program memory.
- The assembler ensures that you never use addresses beyond actual memory.
- Still, PC is 21 bits, because Microchip wanted a uniform architecture.

Where the PIC wakes up?

- In case of PIC family, all members regardless of the family and variation, the microcontroller wakes up at memory address 0000 when it is powered up.
- It means, when the PIC is powered up,the PC has the value of 00000 in it.
- It eventually means that the first opcode is expected to be stored at ROM address 00000H.

Execution of a program byte by byte

- Refer to the book page 77.

One program memory location = 1 instruction word = 16 bits = 2 bytes

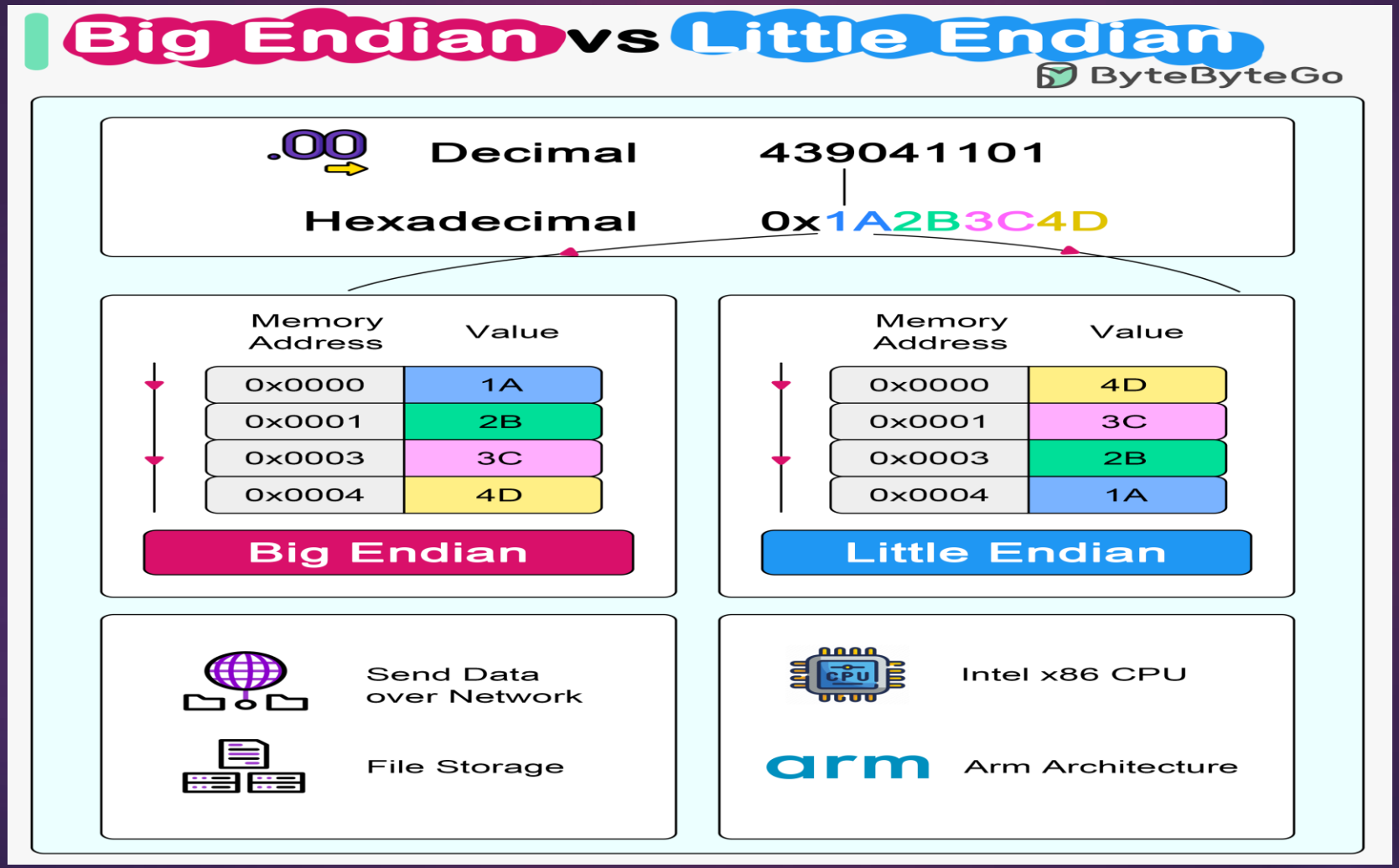
Program memory is word-addressable (not byte-addressable).

Difference?

- Data memory needs to handle small units (like 8-bit registers, SFRs, variables), so byte addressing makes it flexible.

- Program memory stores instructions, which are mostly 16 bits, so word addressing avoids wasting memory (PC increments per instruction, not per byte).

Little Endian vs Big Endian Method



- PIC18 is little-endian.
- The least significant byte (LSB) of a multi-byte value is stored at the lowest RAM address.
- The most significant byte (MSB) goes to the next higher address.

WORD ADDRESS	HIGH BYTE	LOW BYTE
000000h	0Eh	25h
000002h	0Fh	34h
000004h	0Fh	11h
000006h	0Fh	12h
000008h	0Fh	1Ch
00000Ah	0Fh	06h
00000Ch	6Eh	10h
00000Eh	EFh	07h
000010h	0Fh	00h

Types of Architecture for µP

- Von Neumann Architecture
- Harvard Architecture

- While code provides instructions to processor, data provides the information to be processed. µP uses buses (wire traces) to access the code ROM and data RAM memory spaces.
- Early computers used the same bus for accessing both the code and data. Such an architecture is **Von Neumann**.

Drawback of Von Neumann???

- For Von Neumann systems, the process of accessing the code or datacould cause them toget in each other's way and slow down the processing speed of the processor, because each had to wait for the other to finish fetching.

Alternate Solution could be.....

Harvard Architecture to speed up the process of program execution.

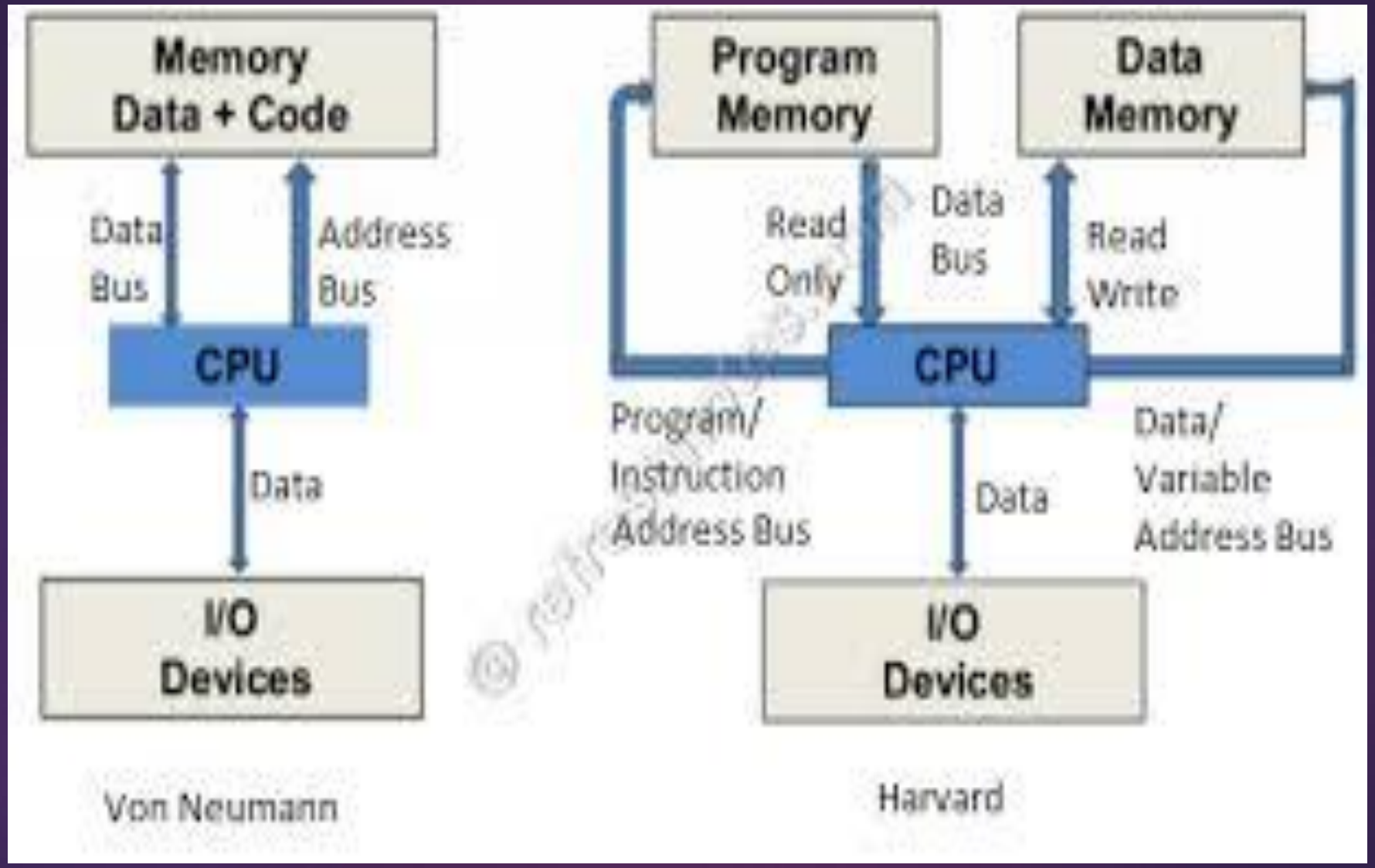
- PIC18 has Harvard Architecture.

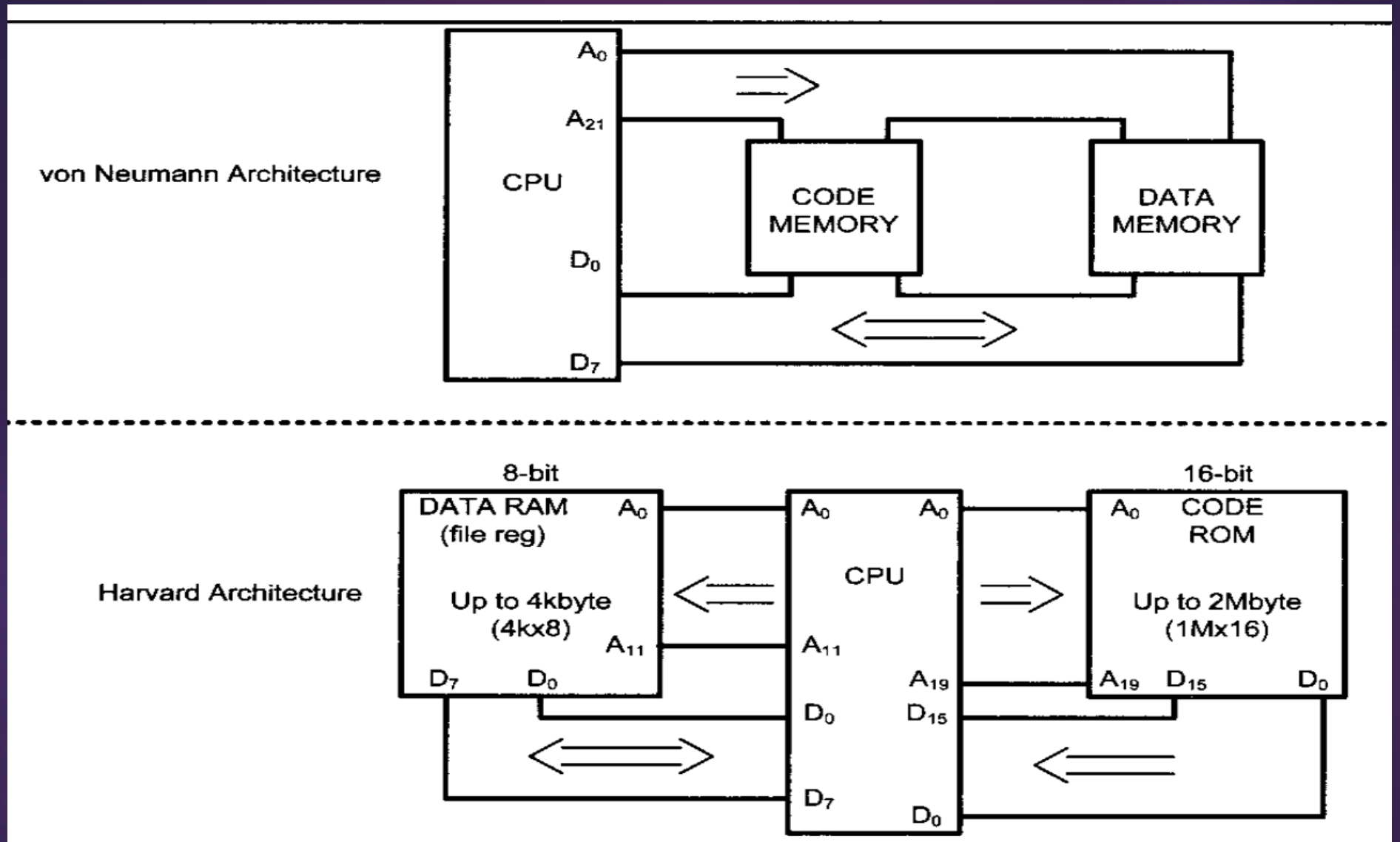
Harvard Architecture

- In Harvard Architecture, we have separate buses for code and data memory.
- It means we need 4 set of buses:

- Program Address Bus
- Program Data Bus
- Data Address Bus
- Data Data Bus

- Each memory has its own address bus and data bus. This allows the CPU to fetch instructions and read/write data simultaneously.


Memory
Data + Code
Data
Bus
Address
Bus
CPU
Data
I/O
Devices
Von Neumann
Program
Memory
Data
Memory
Read
Only
Data
Bus
Read
Write
CPU
Program/
Instruction
Address Bus
Data
Data/
Variable
Address Bus
I/O
Devices
Harvard



gure 2-14. von Neumann vs. Harvard Architecture

Example:

- Example: MOVWF PORTA

- Program Address Bus → PC value points to instruction in ROM
- Program Data Bus → fetched 16-bit instruction delivered to CPU
- Data Address Bus → points to the SFR address of PORTA
- Data Data Bus → writes WREG value into PORTA

PIC18 Architecture

- **Program memory**

Flash ROM

Stores 16-bit instruction words

Addressed by a 21-bit Program Counter

- **Data memory**

RAM + Special Function Registers (SFRs)

Byte-oriented (8 bits wide)

Accessed through separate data buses

- **RISC and Harvard Architecture**