An Introduction to Microprocessors

By Larry Macklin, N8CGP

Microprocessors

- A “Computer On A Chip”
- The world’s first microprocessor- the 4004 was invented in 1971 by Intel
- In the 45 years since, microprocessors have become smaller, faster, more powerful, cheaper, more prevalent, and easier to use.

- Let’s look at what is inside a microprocessor. But first, let’s review the type of numbers that microprocessors use.

Binary Logic & Numbers

- Microprocessors operate using binary (two-state) logic
- It is much easier for electronics to detect the presence or absence of a voltage (two states) than it is to measure the magnitude of a voltage and assign a value to it from a range of discrete steps.
- Thus, microprocessors perform all internal calculations and logic operations using a Base 2 numbering system

Numbering System Basics

- The Base of a number system specifies how many unique values are contained in one digit.
- We normally use a Base 10 numbering system in our everyday lives.
- There are 10 values within a single digit (0-9)
- In a Base 2 (binary) numbering system, there are 2 values within a single digit (0-1)
- When counting upwards, when we reach the largest value for a digit, we carry a “1” into the next digit to the left.

Numbering System Basics

- Each digit position corresponds to an exponential power of the Base

For Base 10

10^4 $1 \times 10^4 = 1 \times 10000 = 10000$

10^3 $9 \times 10^3 = 9 \times 1000 = 9000$

10^2 $4 \times 10^2 = 4 \times 100 = 400$

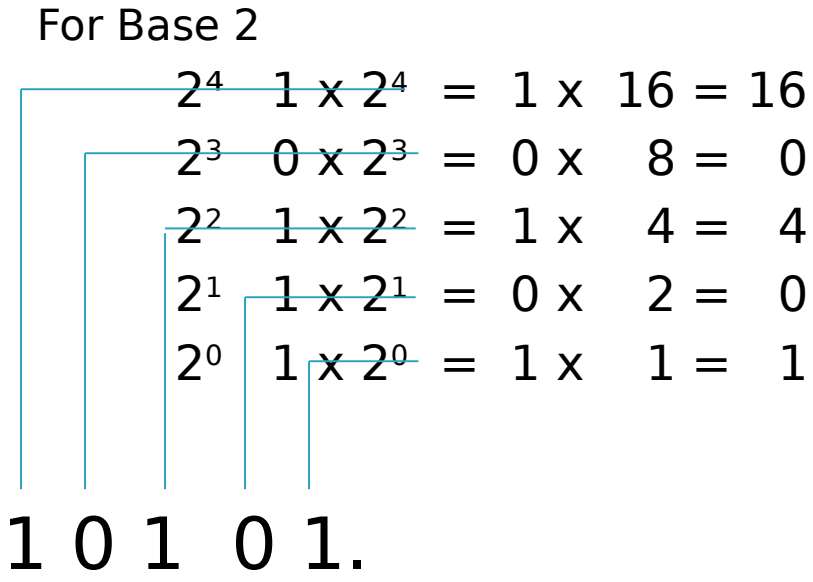
10^1 $0 \times 10^1 = 0 \times 10 = 00$

10^0 $3 \times 10^0 = 3 \times 1 = 3$

1 9 4 0 3.

Numbering System Basics

- Each digit position corresponds to an exponential power of the Base



Numbering System Basics

- Base 2 compared to Base 10
  - A Base 2 number usually requires more digits to represent the same value as a Base 10 number
  - Base 2 (binary) values are easier for the microprocessor to understand
  - Base 2 values are harder for humans to understand
  - Base 2 doesn’t use fractions! 0/0, 0/1, 1/0, 1/1

- Binary Definitions
  - One Binary digIT is called a BIT
  - A group of 4 bits is called a NIBBLE
  - A group of 8 bits is called a BYTE (2 nibbles = 1 byte)
  - A BIT may represent a logic state (on/off or true/false) or a number

Counting in Base 2

Binary	Base 10	Base 16
0	0	0
1	1	1
1 0	2	2
1 1	3	3
1 0 0	4	4
1 0 1	5	5
1 1 0	6	6
1 1 1	7	7
1 0 0 0	8	8
1 0 0 1	9	9
1 0 1 0	10	A
1 0 1 1	11	B
1 1 0 0	12	C
1 1 0 1	13	D
1 1 1 0	14	E
1 1 1 1	15	F

Binary Shorthand- Hexadecimal

- In order to reduce the length of a binary number, we can group 4 binary digits (bits) together and represent them with a single Base 16 digit.

2^{15} 2^{12} 2^{11} 2^{8} 2^{7} 2^{4} 2^{3} 2^{0}

1 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1

9 F 7 1

16^{3} 16^{2} 16^{1} 16^{0}

$= 9 \times 4096 = 36864$

$= F \times 256 = 3840$

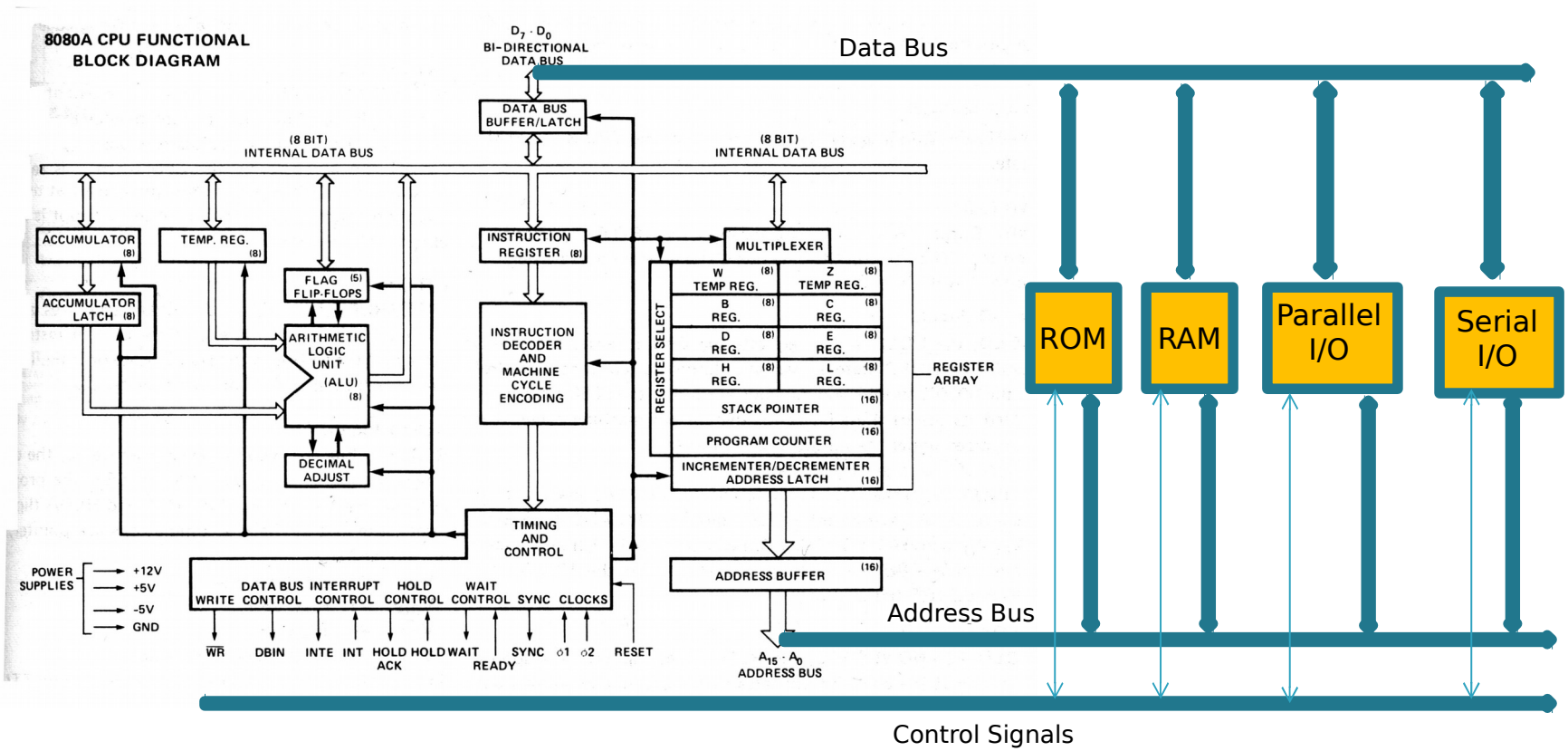
$= 7 \times 16 = 112$

$= 1 \times 1 = 1$

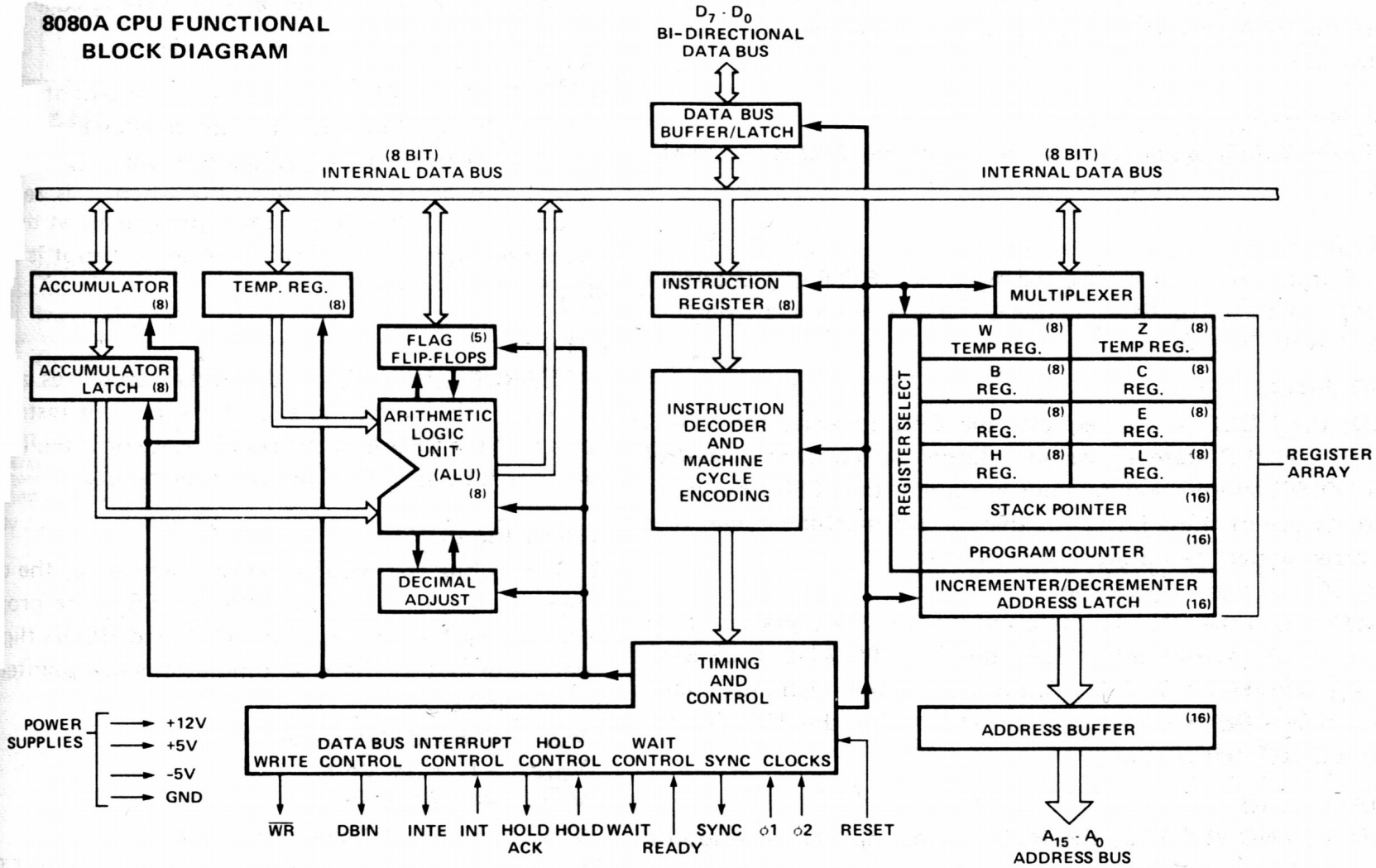
Components of a Microprocessor System

- Data Bus
- Address Bus
- Program Memory
- Data Memory
- Program Counter
- Instruction Register & Decoder
- Arithmetic Logic Unit
- Accumulator Register
- Flag Register
- General Purpose Registers
- Stack Pointer
- Clock
- Control Inputs & Outputs
- Program Instructions
- Input & Output Peripherals

Microprocessor System



Microprocessor Internal Diagram



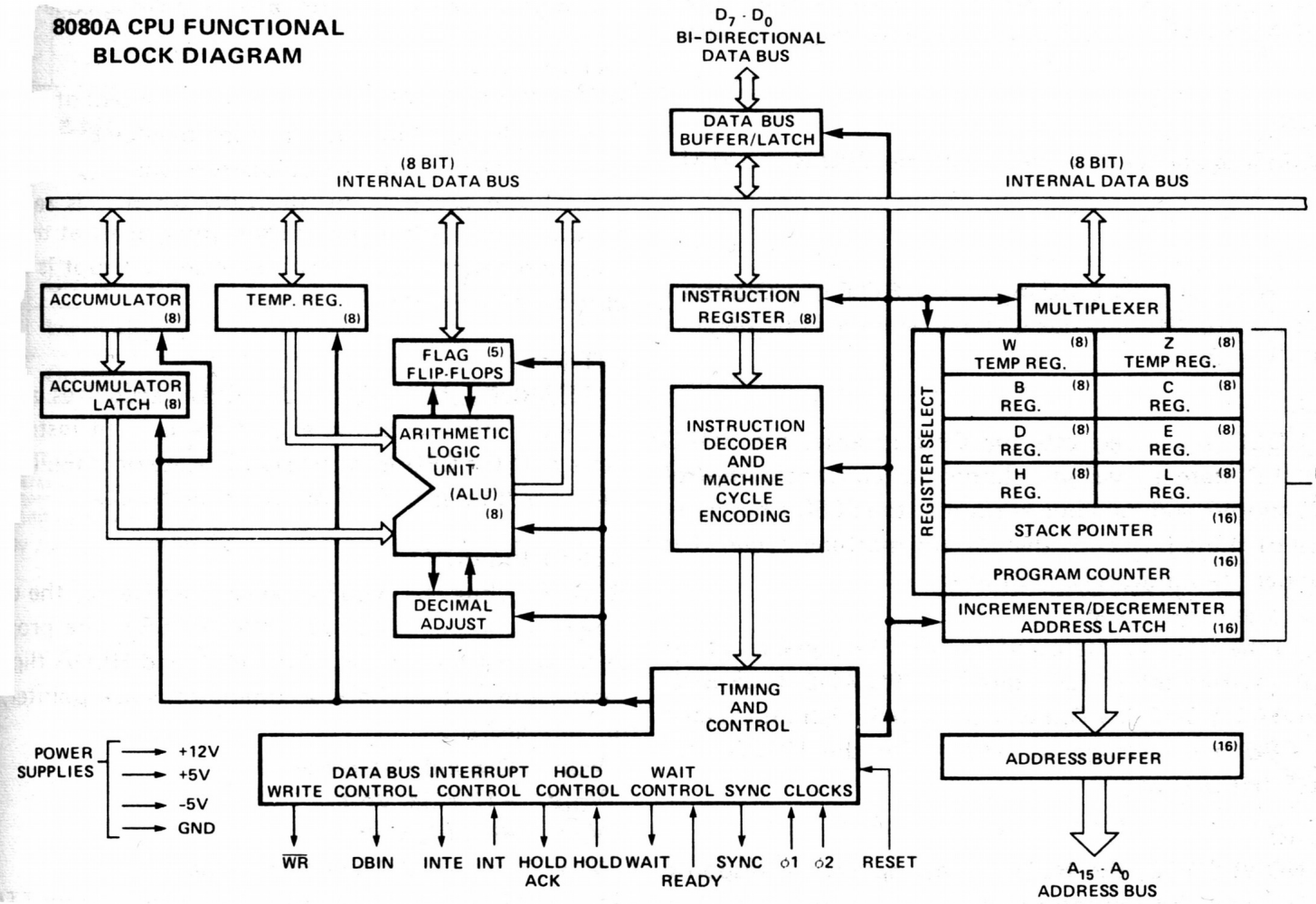
Microprocessor Instructions

- Microprocessor instructions are very low-level and simple.
- The instruction set is specific to the hardware architecture of the microprocessor. Thus, each microprocessor model has a unique instruction set.
- Instructions are coded as groups of binary digits. The groups are usually a multiple of 8 bits.
- The binary value for each instruction is called its operational code, or opcode.

Microprocessor Instructions

- For our example 8080 processor, the instructions are classified as follows:
  - Move, Load, & Store
  - Stack Operations
  - Jump Instructions
  - Call & Return
  - Increment & Decrement
  - Add & Subtract
  - Logical
  - Rotate/Shift
  - Input & Output
  - Special & Control

Instruction Execution



Program Memory

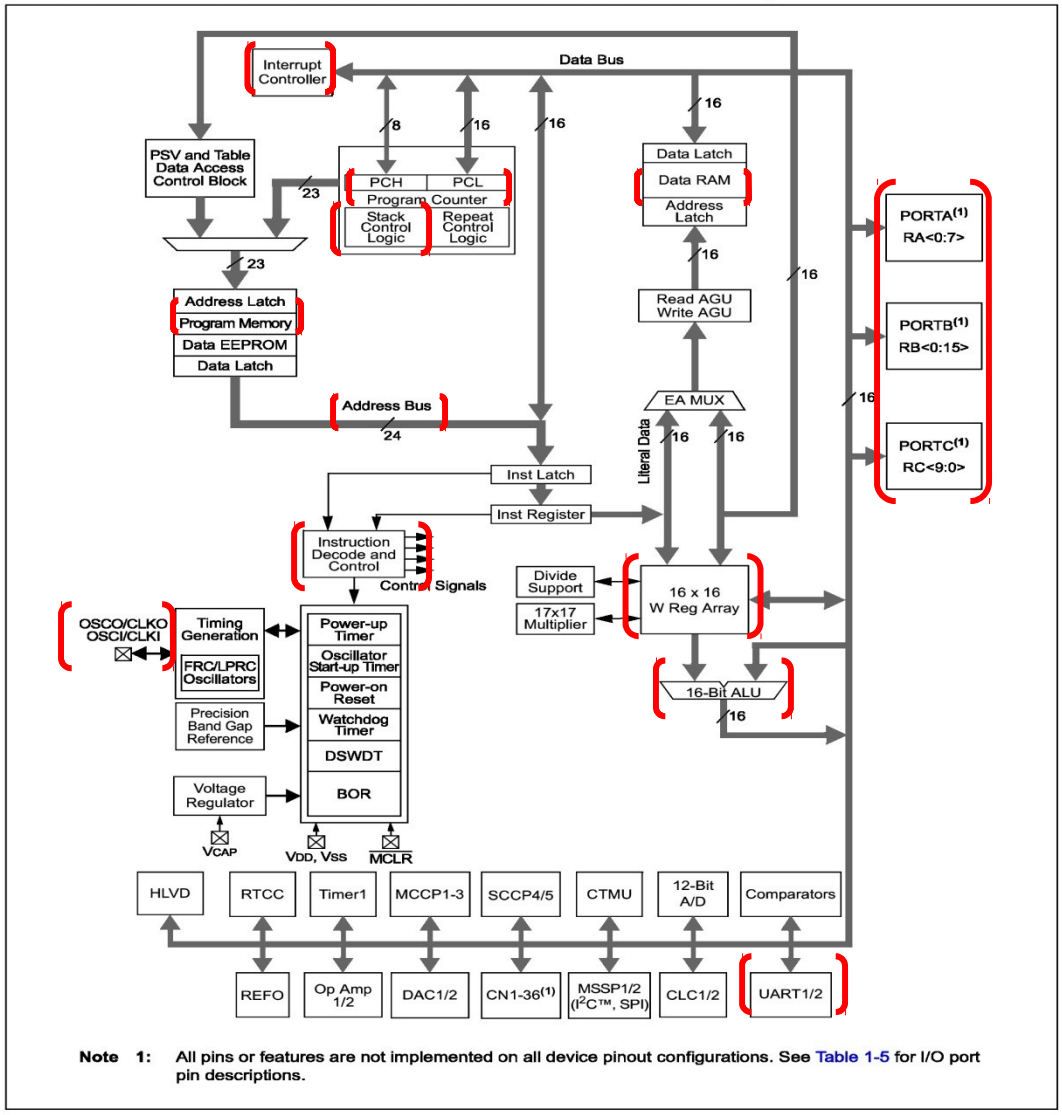
→	1000	06	MVIB,00
	1001	00	
	1002	3E	MVIA,00
	1003	00	
	1004	3C	INRA
	1005	C2	JNZ
	1004		
	1006	10	
	1007	04	
	1008	CD	Call
	100E		
	1009		
	100A		
	100B	C3	JMP
	1002		
	100C	10	
	100D	02	
	100E	04	INRB
	100F	78	MOV A,B
	1010	D3	OUT,20
	1011	20	
	1012	C9	RET

Stack

2FFC	
2FFD	
2FFE	
→ 2FFF	

PIC 24FV16KM204 Block Diagram

FIGURE 1-1: PIC24FXXXXX FAMILY GENERAL BLOCK DIAGRAMS



Note 1: All pins or features are not implemented on all device pinout configurations. See Table 1-5 for I/O port pin descriptions.

Generating Program Instructions

- Three methods for creating the software instructions (a.k.a. The Program)

- Manual Machine Coding
  - - You supply the Binary Operation Codes
  - - You keep track of the memory location for each instruction
  - - You keep track of register and stack usage
  - - Program jumps are to absolute addresses, not labels
  - - Requires knowledge of the processor architecture
  - - Very specific to one microprocessor model
  - - Tedious and error-prone
  - - Difficult to make changes
  - - Not easy to debug & document

Generating Program Instructions

- Assembler Software
  - - You write the program steps as symbols or Mnemonics
  - - The assembler provides the op codes
  - - The assembler software keeps track of the addresses
  - - You keep track of register and stack usage
  - - Labels are used identify specific instructions
  - - Program jumps are made to labels, not addresses
  - - Requires knowledge of the processor architecture
  - - Usually specific to one microprocessor family
  - - Less tedious and error-prone than machine coding
  - - Easier to make changes
  - - Easier to debug & document
  - - Assembler software is specific to the microprocessor model
  - - Debugging and emulator software tools are often available as part of the assembler software package.
  - - Typically used for timing or resource-critical applications

Generating Program Instructions

- High-Level Language & Interpreter/Compiler Software
  - - Program steps are written using languages such as Basic, Fortran, C, etc.
  - - Program steps use standard verbs, commands, and syntax for that language.
  - - Labels and variables are used rather than referring to specific instructions, registers, or memory locations.
  - - Program steps are not specific to any microprocessor model or family.
  - - Author needs no knowledge of the underlying processor architecture
  - - Requires knowledge of the language vocabulary & syntax
  - - Faster to write, modify, debug, and document
  - - Only the compiler portion of the software is specific to the microprocessor model or family.
  - - Debugging and emulator software tools are usually available as part of the compiler software package.

Questions?

- Thanks for your attention!