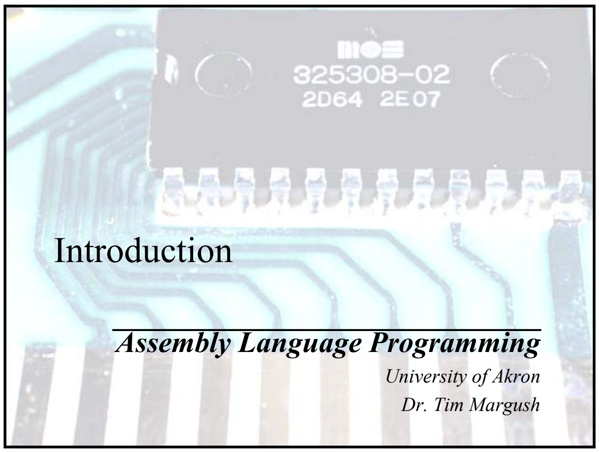
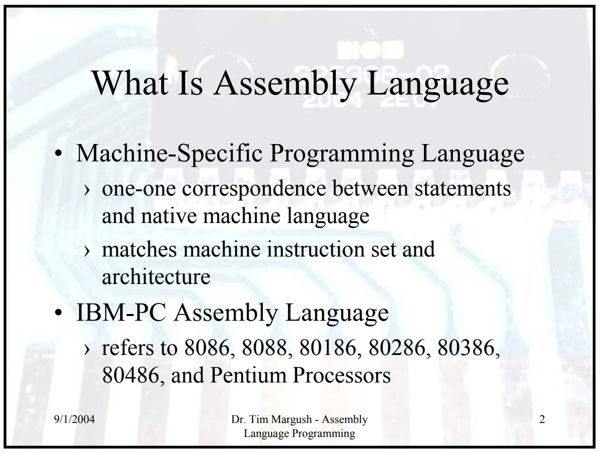


Introduction

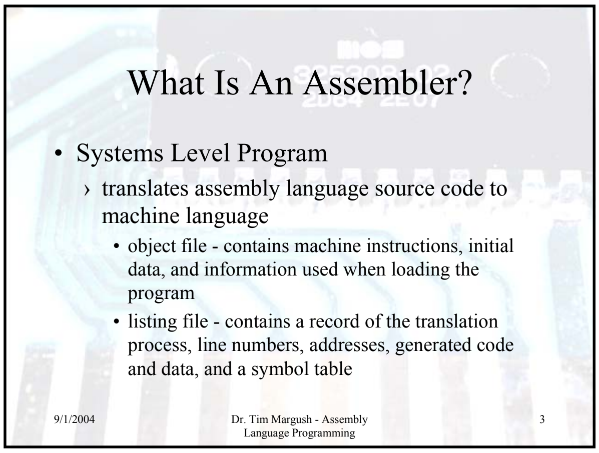
Assembly Language Programming

University of Akron
Dr. Tim Margush



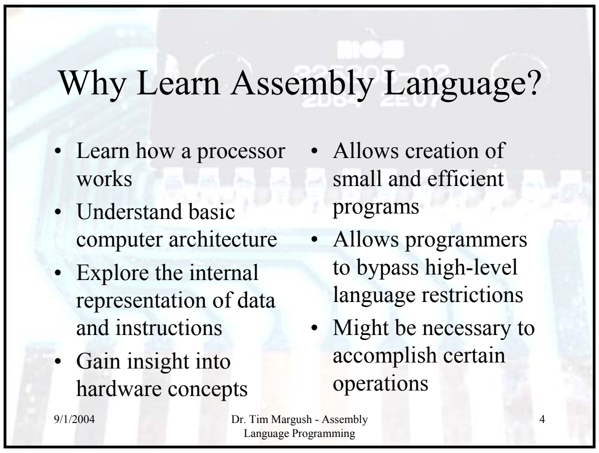
What Is Assembly Language

- Machine-Specific Programming Language
  - › one-one correspondence between statements and native machine language
  - › matches machine instruction set and architecture
- IBM-PC Assembly Language
  - › refers to 8086, 8088, 80186, 80286, 80386, 80486, and Pentium Processors



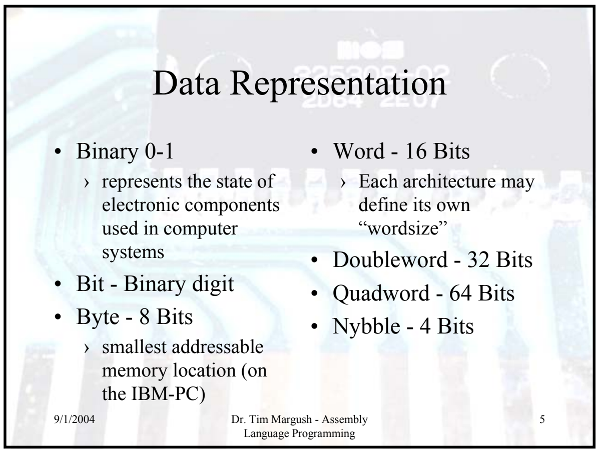
What Is An Assembler?

- Systems Level Program
  - › translates assembly language source code to machine language
    - object file - contains machine instructions, initial data, and information used when loading the program
    - listing file - contains a record of the translation process, line numbers, addresses, generated code and data, and a symbol table



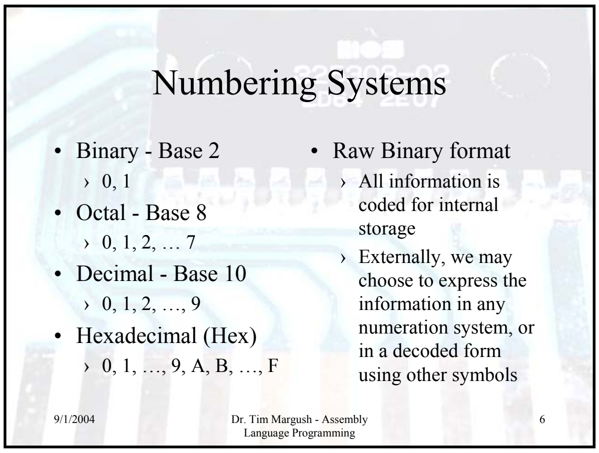
Why Learn Assembly Language?

- Learn how a processor works
- Understand basic computer architecture
- Explore the internal representation of data and instructions
- Gain insight into hardware concepts
- Allows creation of small and efficient programs
- Allows programmers to bypass high-level language restrictions
- Might be necessary to accomplish certain operations



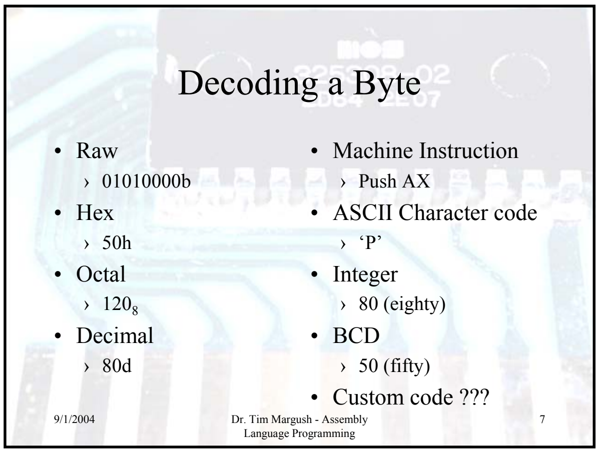
Data Representation

- Binary 0-1
  - › represents the state of electronic components used in computer systems
- Bit - Binary digit
- Byte - 8 Bits
  - › smallest addressable memory location (on the IBM-PC)
- Word - 16 Bits
  - › Each architecture may define its own “wordsize”
- Doubleword - 32 Bits
- Quadword - 64 Bits
- Nybble - 4 Bits



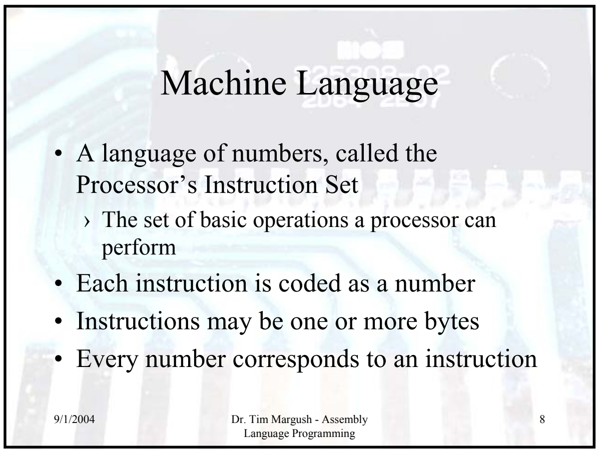
Numbering Systems

- Binary - Base 2
  - › 0, 1
- Octal - Base 8
  - › 0, 1, 2, … 7
- Decimal - Base 10
  - › 0, 1, 2, …, 9
- Hexadecimal (Hex)
  - › 0, 1, …, 9, A, B, …, F
- Raw Binary format
  - › All information is coded for internal storage
  - › Externally, we may choose to express the information in any numeration system, or in a decoded form using other symbols



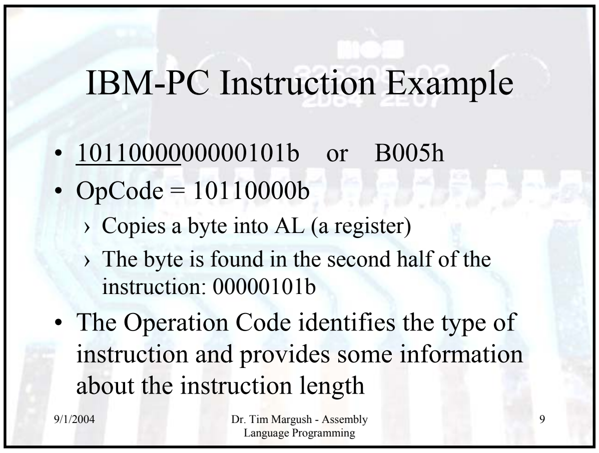
Decoding a Byte

- Raw
  - 01010000b
- Hex
  - 50h
- Octal
  - 120_8
- Decimal
  - 80d
- Machine Instruction
  - Push AX
- ASCII Character code
  - ‘P’
- Integer
  - 80 (eighty)
- BCD
  - 50 (fifty)
- Custom code ???



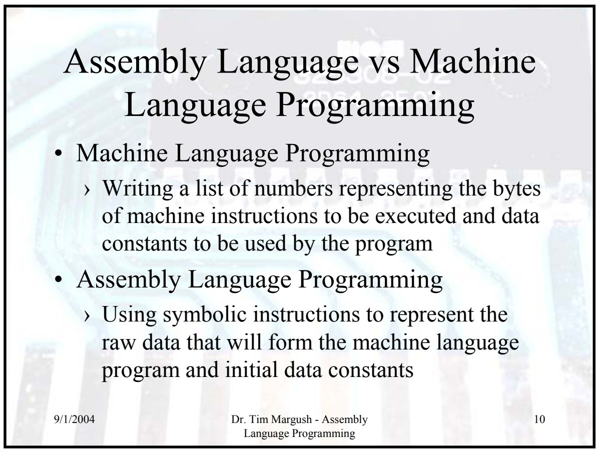
Machine Language

- A language of numbers, called the Processor’s Instruction Set
  - The set of basic operations a processor can perform
- Each instruction is coded as a number
- Instructions may be one or more bytes
- Every number corresponds to an instruction



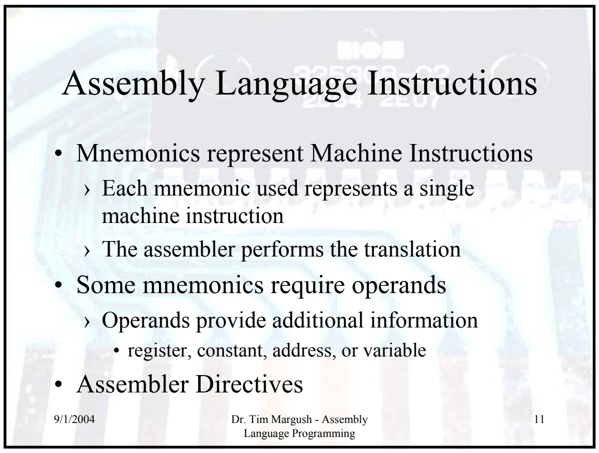
IBM-PC Instruction Example

- <u>10110000</u>00000101b or B005h
- OpCode = 10110000b
  - Copies a byte into AL (a register)
  - The byte is found in the second half of the instruction: 00000101b
- The Operation Code identifies the type of instruction and provides some information about the instruction length



Assembly Language vs Machine Language Programming

- Machine Language Programming
  - › Writing a list of numbers representing the bytes of machine instructions to be executed and data constants to be used by the program
- Assembly Language Programming
  - › Using symbolic instructions to represent the raw data that will form the machine language program and initial data constants



Assembly Language Instructions

- Mnemonics represent Machine Instructions
  - › Each mnemonic used represents a single machine instruction
  - › The assembler performs the translation
- Some mnemonics require operands
  - › Operands provide additional information
    - register, constant, address, or variable
- Assembler Directives