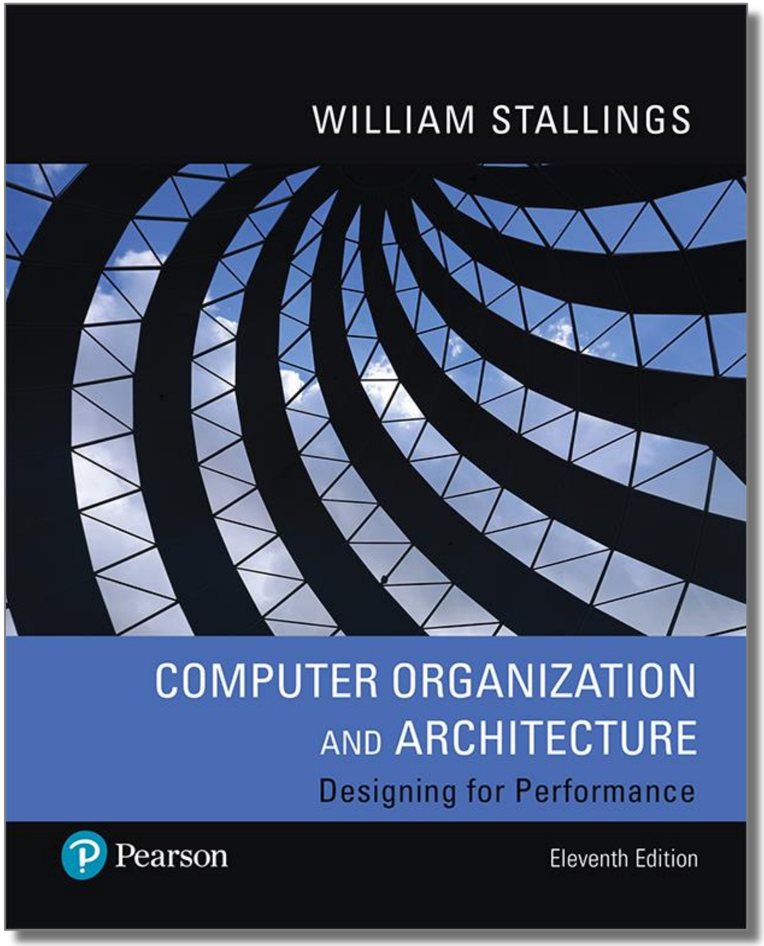
Computer Organization and Architecture
Designing for Performance

11th Edition



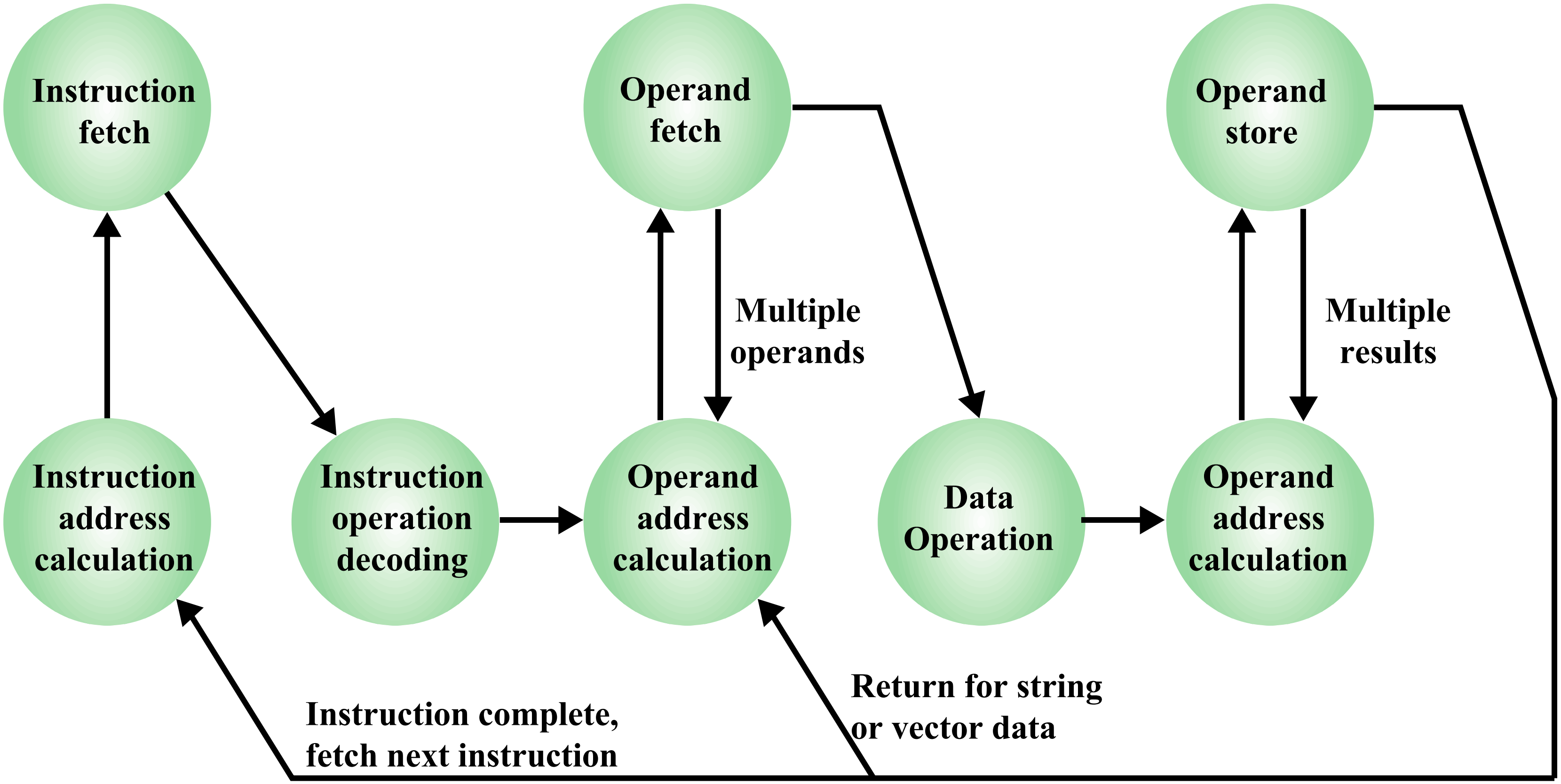
Chapter 13

Instruction Sets: Characteristics and Functions

Machine Instruction Characteristics

- The operation of the processor is determined by the instructions it executes, referred to as *machine instructions* or *computer instructions*
- The collection of different instructions that the processor can execute is referred to as the processor's *instruction set*
- Each instruction must contain the information required by the processor for execution

Figure 13.1
Instruction Cycle State Diagram



Elements of a Machine Instruction

- **Operation code:** Specifies the operation to be performed (e.g., ADD, I/O). The operation is specified by a binary code, known as the operation code, or opcode.
- **Source operand reference:** The operation may involve one or more source operands, that is, operands that are inputs for the operation.
- **Result operand reference:** The operation may produce a result.
- **Next instruction reference:** This tells the processor where to fetch the next instruction after the execution of this instruction is complete.

Source and result operands can be in one of four areas:

1) Main or virtual memory
   - As with next instruction references, the main or virtual memory address must be supplied

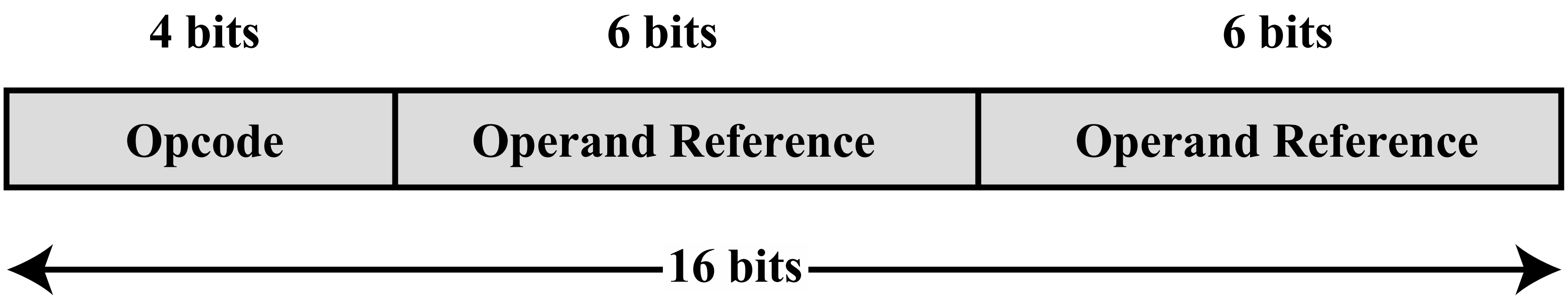
2) I/O device
   - The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address

3) Processor register
   - A processor contains one or more registers that may be referenced by machine instructions.
   - If more than one register exists each register is assigned a unique name or number and the instruction must contain the number of the desired register

4) Immediate
   - The value of the operand is contained in a field in the instruction being executed

Figure 13.2
A Simple Instruction Format

- Within the computer each instruction is represented by a sequence of bits
- The instruction is divided into fields, corresponding to the constituent elements of the instruction



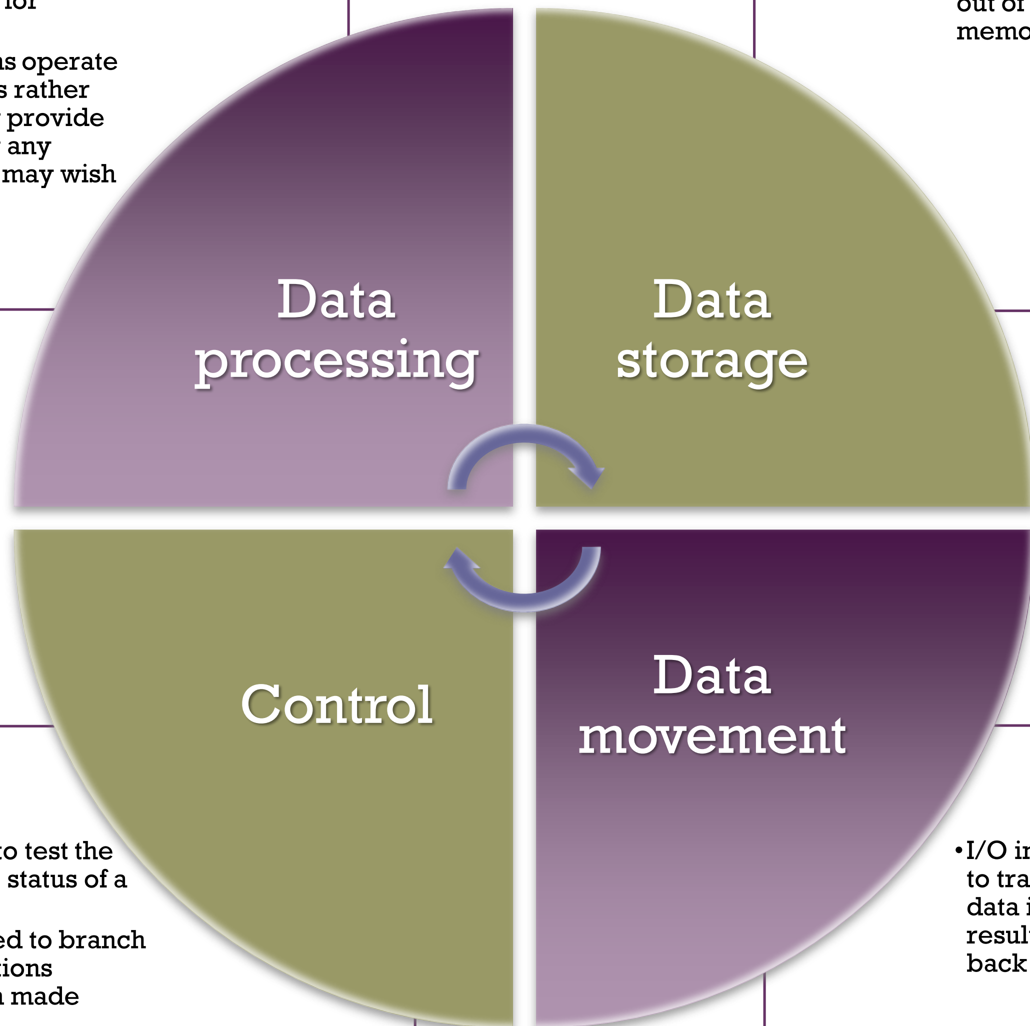
Instruction Representation

- Opcodes are represented by abbreviations called *mnemonics*
- Examples include:
  - ADD Add
  - SUB Subtract
  - MUL Multiply
  - DIV Divide
  - LOAD Load data from memory
  - STOR Store data to memory
- Operands are also represented symbolically
- Each symbolic opcode has a fixed binary representation
  - The programmer specifies the location of each symbolic operand

Instruction Types

- Arithmetic instructions provide computational capabilities for processing numeric data
- Logic (Boolean) instructions operate on the bits of a word as bits rather than as numbers, thus they provide capabilities for processing any other type of data the user may wish to employ

- Movement of data into or out of register and or memory locations



- Test instructions are used to test the value of a data word or the status of a computation
- Branch instructions are used to branch to a different set of instructions depending on the decision made

- I/O instructions are needed to transfer programs and data into memory and the results of computations back out to the user

Figure 13.3

Programs to Execute $Y = \dfrac{A - B}{C + (D \times E)}$

Instruction		Comment
SUB	Y, A, B	Y ¬ A – B
MPY	T, D, E	T ¬ D ´ E
ADD	T, T, C	T ¬ T + C
DIV	Y, Y, T	Y ¬ Y ÷ T

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ¬ A
SUB	Y, B	Y ¬ Y – B
MOVE	T, D	T ¬ D
MPY	T, E	T ¬ T ´ E
ADD	T, C	T ¬ T + C
DIV	Y, T	Y ¬ Y ÷ T

(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ¬ D
MPY	E	AC ¬ AC ´ E
ADD	C	AC ¬ AC + C
STOR	Y	Y ¬ AC
LOAD	A	AC ¬ A
SUB	B	AC ¬ AC – B
DIV	Y	AC ¬ AC ÷ Y
STOR	Y	Y ¬ AC

(c) One-address instructions

Table 13.1
Utilization of Instruction Addresses (Nonbranching Instructions)

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
3	OP	T ← (T – 1) OP T

AC = accumulator
T = top of stack
(T – 1) = second element of stack
A, B, C = memory or register locations

Instruction Set Design

Very complex because it affects so many aspects of the computer system

Defines many of the functions performed by the processor

Programmer's means of controlling the processor

Fundamental design issues:

Operation repertoire
- How many and which operations to provide and how complex operations should be

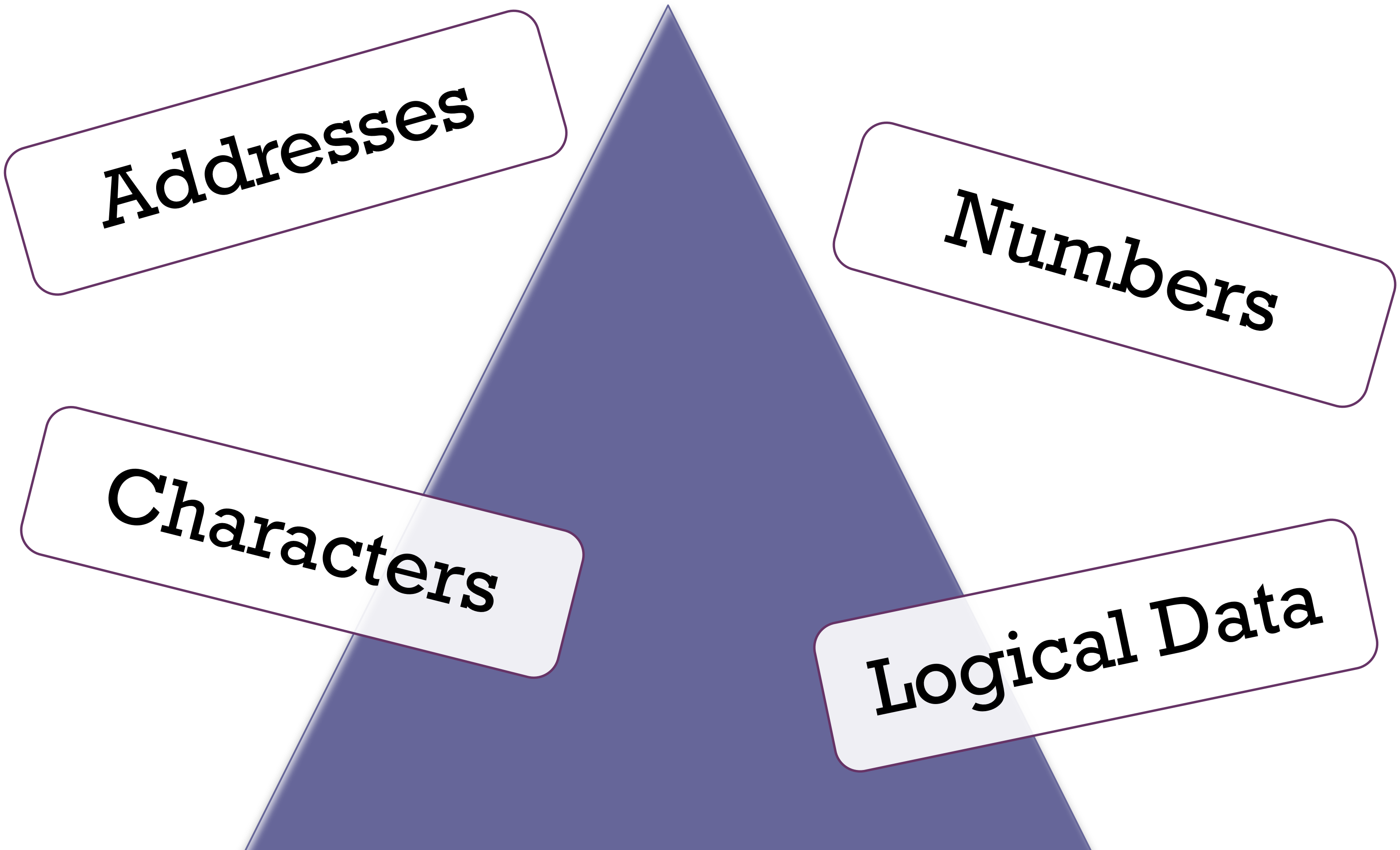
Data types
- The various types of data upon which operations are performed

Instruction format
- Instruction length in bits, number of addresses, size of various fields, etc.

Registers
- Number of processor registers that can be referenced by instructions and their use

Addressing
- The mode or modes by which the address of an operand is specified

Types of Operands



Numbers

- All machine languages include numeric data types
- Numbers stored in a computer are limited:
  - Limit to the magnitude of numbers representable on a machine
  - In the case of floating-point numbers, a limit to their precision
- Three types of numerical data are common in computers:
  - Binary integer or binary fixed point
  - Binary floating point
  - Decimal
- Packed decimal
  - Each decimal digit is represented by a 4-bit code with two digits stored per byte
  - To form numbers 4-bit codes are strung together, usually in multiples of 8 bits

Characters

- A common form of data is text or character strings
- Textual data in character form cannot be easily stored or transmitted by data processing and communications systems because they are designed for binary data
- Most commonly used character code is the International Reference Alphabet (IRA)
  - Referred to in the United States as the American Standard Code for Information Interchange (ASCII)
- Another code used to encode characters is the Extended Binary Coded Decimal Interchange Code (EBCDIC)
  - EBCDIC is used on IBM mainframes

Logical Data

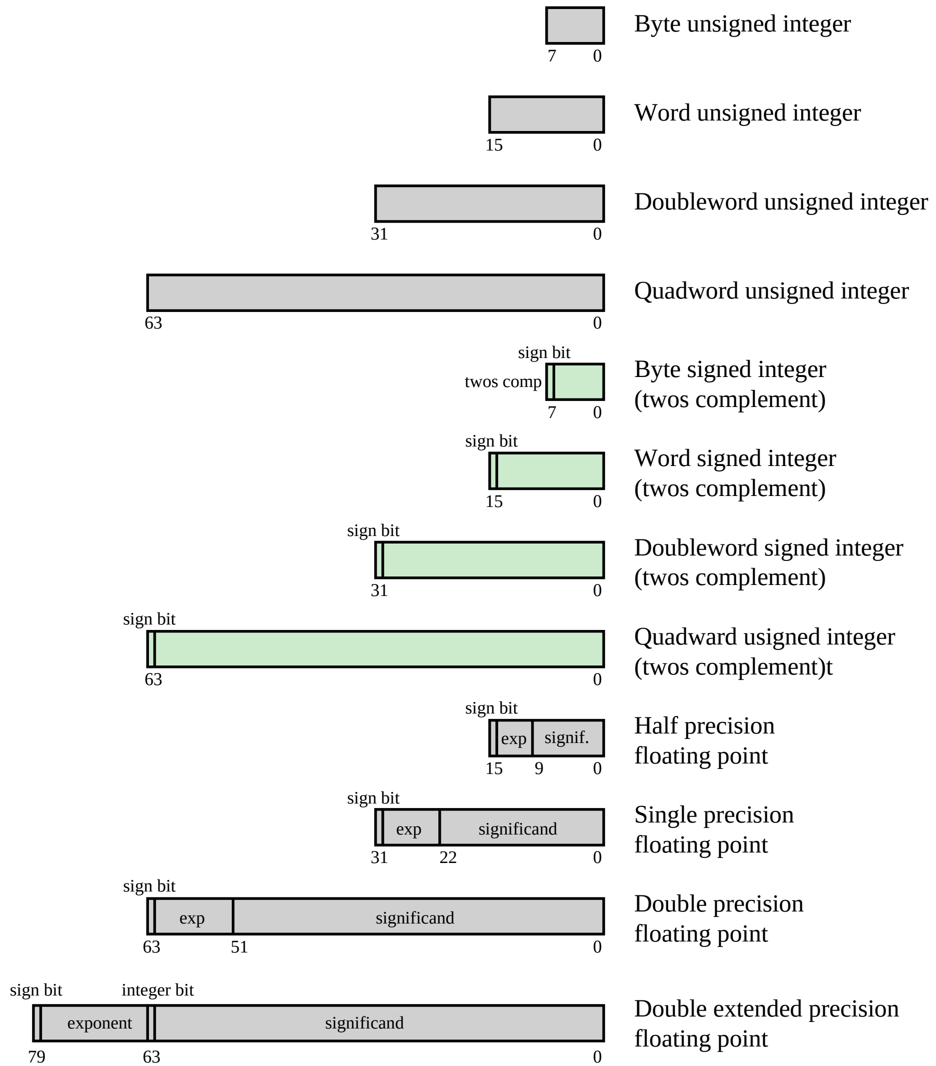
- An *n*-bit unit consisting of *n* 1-bit items of data, each item having the value 0 or 1
- Two advantages to bit-oriented view:
  - Memory can be used most efficiently for storing an array of Boolean or binary data items in which each item can take on only the values 1 (true) and 0 (false)
  - To manipulate the bits of a data item
    - If floating-point operations are implemented in software, we need to be able to shift significant bits in some operations
    - To convert from IRA to packed decimal, we need to extract the rightmost 4 bits of each byte

Table 13.2 x86 Data Types

Data Type	Description
General	Byte, word (16 bits), doubleword (32 bits), quadword (64 bits), and double quadword (128 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 16-bit, 32-bit, or 64-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Far pointer	A logical address consisting of a 16-bit segment selector and an offset of 16, 32, or 64 bits. Far pointers are used for memory references in a segmented memory model where the identity of a segment being accessed must be specified explicitly.
Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to 32 bits.
Bit string	A contiguous sequence of bits, containing from zero to $2^{23} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing from zero to $2^{23} - 1$ bytes.
Floating point	See Figure 13.4.
Packed SIMD (single instruction, multiple data)	Packed 64-bit and 128-bit data types.

(Table can be found on page 443
In the textbook)

Figure 13.4
x86 Numeric Data Formats



Single-Instruction-Multiple-Data (SIMD) Data Types

- Introduced to the x86 architecture as part of the extensions of the instruction set to optimize performance of multimedia applications
- These extensions include MMX (multimedia extensions) and SSE (streaming SIMD extensions)
- Data types:
  - Packed byte and packed byte integer
  - Packed word and packed word integer
  - Packed doubleword and packed doubleword integer
  - Packed quadword and packed quadword integer
  - Packed single-precision floating-point and packed double-precision floating-point

Copyright