

MODULE 1: DATA STORAGE AND DATA MANIPULATION

PART I: DATA STORAGE (Chapter 1.1 - 1.7)

1.1 BITS AND THEIR STORAGE

Fundamental Concept

Bit - Short for **binary digit** - The smallest unit of data in a computer

- Can only be 0 or 1
- These are symbols whose meaning depends on the application
- Sometimes represent numeric values
- Sometimes represent characters, images, sounds

Boolean Operations

George Boole (1815-1864) - Pioneer in the field of mathematics called logic

- Operations that manipulate true/false values
- Represent 0 as false, 1 as true

Three Basic Boolean Operations:

BOOLEAN OPERATIONS					
Inputs	AND	OR	XOR	NOT	
A B	A AND B	A OR B	A XOR B	NOT A	
0 0	0	0	0	1	
0 1	0	1	1	1	
1 0	0	1	1	0	
1 1	1	1	0	0	

AND Operation:

- Output is 1 ONLY when BOTH inputs are 1
- Based on compound statements: "P AND Q"

- Example: "Kermit is a frog AND Miss Piggy is an actress" is true only if both are true

OR Operation:

- Output is 1 when AT LEAST ONE input is 1
- Based on compound statements: "P OR Q"
- True when at least one component is true

XOR (Exclusive OR) Operation:

- Output is 1 when inputs are DIFFERENT
- Means "either P or Q but not both"

NOT Operation:

- Only ONE input
- Output is the opposite of the input
- Changes true to false, false to true

Gates and Flip-Flops

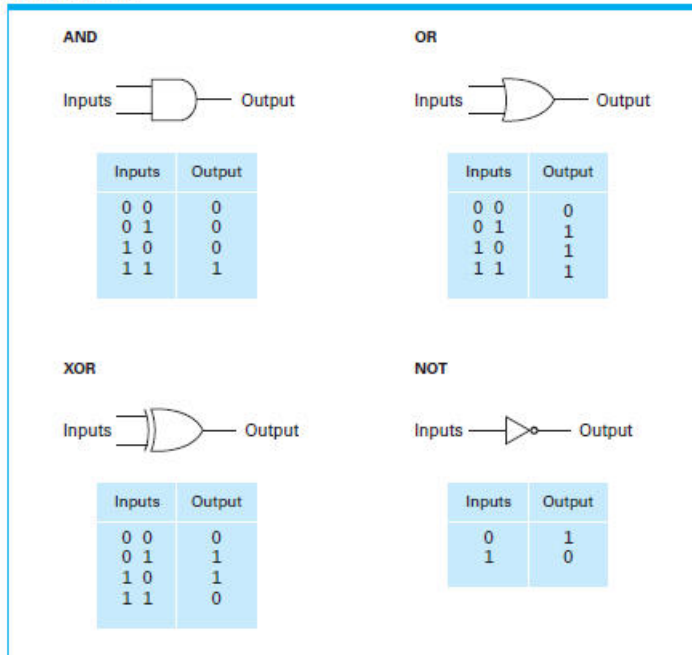
Gate: A device that produces the output of a Boolean operation when given the operation's input values

- Implemented as small electronic circuits
- Digits 0 and 1 represented as voltage levels
- Different shapes represent different operations

text

FIGURE 1.2

Figure 1.2 A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values



Flip-Flop:

- A fundamental unit of computer memory
- Circuit that produces output value of 0 or 1
- Output remains CONSTANT until a pulse causes it to shift to the other value
- Can be set to "remember" a zero or a one

How Flip-Flop Works:

1. Both inputs remain 0 → output (0 or 1) will not change
2. Temporarily place 1 on upper input → forces output to be 1
3. Temporarily place 1 on lower input → forces output to be 0
4. Output persists after input returns to 0

Significance of Flip-Flop:

1. Demonstrates how devices are constructed from gates (digital circuit design)
2. Provides example of **abstraction** and use of **abstract tools**
 - Different internal structures can have same external properties
 - Engineer needs only understanding of external properties
 - Hierarchical structure - each level uses lower level components as abstract tools
3. One means of storing a bit within a modern computer

VLSI (Very Large-Scale Integration):

- Technology allowing millions of electrical components on a wafer (chip)
- Creates miniature devices containing millions of flip-flops with controlling circuitry
- Can create entire computer system on a single chip

Hexadecimal Notation

Purpose: Simplify representation of long bit patterns (streams)

- Bit patterns in machines tend to have lengths in multiples of four
- Uses single symbol to represent pattern of four bits

FIGURE 1.6

Figure 1.6 The hexadecimal encoding system

Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Examples:

- Bit pattern 10110101 → Divide into substrings: 1011 0101 → B5
- Bit pattern 1010010011001000 → A4C8

1.2 MAIN MEMORY

Memory Organization

Main Memory: Large collection of circuits (flip-flops) capable of storing single bits

Memory Cell: Manageable unit of storage

- Typical cell size = **8 bits**
- **Byte** = String of eight bits
- Small computers (embedded devices): few hundred cells
- Large computers: billions of cells

Terminology:

- **High-order end** - Left end of the row
- **Low-order end** - Right end of the row
- **High-order bit / Most significant bit** - Leftmost bit
- **Low-order bit / Least significant bit** - Rightmost bit

Memory Addressing:

- Each cell assigned unique "name" called its **address**
- Addresses are entirely numeric
- Cells placed in single row and numbered starting with 0
- Provides unique identification AND order
- Enables phrases like "next cell" or "previous cell"

Consequences of Ordering:

- Can store bit patterns longer than single cell by using consecutive cells
- Example: String of 16 bits uses two consecutive memory cells

Read Operation: Other circuits get data from memory by electronically asking for contents of a certain address

Write Operation: Other circuits record information by requesting that a certain bit pattern be placed in cell at particular address

Random Access Memory (RAM):

- Cells can be accessed independently as required
- Reflects ability to access cells in ANY order
- Contrast with mass storage systems (manipulate long strings as amalgamated blocks)

DRAM (Dynamic RAM):

- Most modern RAM uses complex technologies (not flip-flops)
- Stores bits as tiny electric charges that dissipate quickly
- Requires **refresh circuit** - repeatedly replenishes charges many times per second
- Called "dynamic" memory due to this volatility

SDRAM (Synchronous DRAM):

- DRAM with additional techniques to decrease retrieval time

Measuring Memory Capacity

Convenient Design: Total number of cells is a power of two

Traditional Terminology (Powers of Two):

Term	Abbreviation	Size	Exact Value
Kilobyte	KB	1024 bytes	2^{10} bytes
Megabyte	MB	1024 KB	2^{20} bytes
Gigabyte	GB	1024 MB	2^{30} bytes
Terabyte	TB	1024 GB	2^{40} bytes

Note: This is a MISUSE of terminology!

- Kilo- in other fields = 1000 (kilometer = 1000 meters)
- Mega- = 1,000,000 (megahertz = 1,000,000 hertz)

International Standards (Powers of Two - Correct Terminology):

Term	Abbreviation	Size
Kibibyte	KiB	1024 bytes
Mebibyte	MiB	1024 KiB
Gibibyte	GiB	1024 MiB
Tebibyte	TiB	1024 GiB

General Rule:

- In computer measurements: kilo-, mega-, etc. refer to powers of two
- In other contexts: refer to powers of a thousand
- Both general public and many computer scientists still use ambiguous "megabyte"

1.3 MASS STORAGE

Definition: Additional memory devices due to volatility and limited size of main memory

Advantages over Main Memory:

- Less volatility
- Large storage capacities
- Low cost
- Removable storage medium for archival purposes

Disadvantages:

- Require mechanical motion → significantly more time to store/retrieve data
- More prone to mechanical failures than solid state systems
- All activities in main memory are electronic

Magnetic Systems

Magnetic Disk / Hard Disk Drive (HDD):

- Thin spinning disk with magnetic coating
- **Read/write heads** placed above/below disk
- As disk spins, each head traverses a circle called a **track**
- By repositioning heads, different concentric tracks can be accessed

Multiple Platters:

- Several disks mounted on common spindle
- Read/write heads move in unison
- Each repositioning makes a new set of tracks (**cylinder**) accessible

Sectors:

- Each track divided into small arcs called **sectors**
- Information recorded as continuous string of bits
- All sectors contain same number of bits (512 bytes to few KB)
- Simple systems: each track contains same number of sectors
 - Bits less compactly stored on outer tracks (longer than inner tracks)

Zoned-Bit Recording:

- High-capacity disk systems
- Several adjacent tracks collectively known as **zones**
- Typical disk: approximately 10 zones
- All tracks within zone have same number of sectors
- Each zone has MORE sectors per track than zone inside it

- Achieves efficient use of entire disk surface

Performance Measurements:

Measurement	Definition
Seek Time	Time required to move read/write heads from one track to another
Rotation Delay / Latency Time	Half the time for disk to make complete rotation (average time for desired data to rotate to read/write head)
Access Time	Sum of seek time and rotation delay
Transfer Rate	Rate at which data can be transferred to/from disk

Note on Zoned-Bit Recording:

- Data passing read/write head in single rotation = greater for outer zones
- Data transfer rate varies depending on portion of disk being used

Head Crash:

- Read/write heads "float" just off surface (do not touch disk)
- Spacing so close that single particle of dust can jam between head and disk surface
- Destroys both head and disk surface
- Disk systems sealed at factory
- Rotation speeds: several hundred times per second
- Transfer rates: MB per second

Comparison with Electronic Circuits:

- Electronic circuits: nanoseconds (billionths of a second) or less
- Disk systems: milliseconds (thousandths of a second)
- Time to retrieve from disk = "eternity" to electronic circuit awaiting result

Other Magnetic Technologies (Less Widely Used):

Technology	Description	Characteristics
Magnetic Tape	Information recorded on magnetic coating of thin plastic tape wound on reels	Extremely long seek times; low cost; high capacity; suitable for archival backups

Technology	Description	Characteristics
Floppy Disk Drives	Single platters with magnetic coating in portable cartridge	Removable; much lower data densities and access speeds than hard disks

Optical Systems

Compact Disk (CD):

- 12 centimeters (approximately 5 inches) diameter
- Reflective material covered with clear protective coating
- Information recorded by creating variations in reflective surface
- Retrieved by laser detecting irregularities on reflective surface as CD spins

CD-DA (Compact Disk-Digital Audio):

- Original format for audio recordings
- Computer data storage uses essentially same format

CD Storage Format:

- Single track spirals around CD like groove in phonograph record
- UNLIKE phonograph records: spirals from INSIDE OUT
- Track divided into units called **sectors**
- Each sector has own identifying markings
- Capacity: 2KB of data (1/75 second of music for audio)

Uniform Linear Density:

- Information stored at uniform linear density over entire spiraled track
- More information in loop around outer portion than inner portion
- More sectors read in single revolution when scanning outer portion
- CD-DA players: vary rotation speed depending on laser location
- Computer data storage CDs: faster, constant speed → accommodate variations in data transfer rates

Performance Characteristics:

- Best for long, continuous strings of data (reproducing music)
- Random access: magnetic disk approach outperforms spiral approach

Capacities:

Technology	Capacity
Traditional CD	600-700 MB
DVD (Digital Versatile Disk)	Several GB (multiple semitransparent layers)
Blu-ray Disk (BD)	Over 5× DVD capacity (blue-violet laser, finer precision)

Flash Drives

Flash Memory Technology:

- Bits stored by sending electronic signals directly to storage medium
- Electrons trapped in tiny chambers of silicon dioxide
- Alters characteristics of small electronic circuits
- Chambers hold captive electrons for many years without external power
- Excellent for portable, nonvolatile data storage

Limitations:

- Data can be accessed in small byte-size units (like RAM)
- Current technology: stored data must be erased in LARGE BLOCKS
- Repeated erasing slowly damages silicon dioxide chambers
- NOT suitable for general main memory (contents altered many times per second)

Applications:

- Digital cameras
- Smartphones
- Replacing other mass storage technologies in portable applications (laptop computers)
- Not sensitive to physical shock (unlike magnetic and optic systems)

Flash Drive:

- Capacities: hundreds of GB
- Small plastic cases with removable cap
- High capacity + easily connected/disconnected → ideal for portable data storage
- Vulnerability of tiny storage chambers → not as reliable as optical disks for long-term applications

SSD (Solid-State Disk):

- Larger flash memory devices designed to replace magnetic hard disks
- Advantages over hard disks:
 - Resilience to vibrations and physical shock
 - Quiet operation (no moving parts)
 - Lower access times
- Disadvantages:
 - More expensive than hard disks of comparable size
 - Still considered high-end option
 - Limited lifetime (all flash memory technologies)
- **Wear-leveling techniques:** Relocate frequently altered data blocks to fresh locations on drive

SD (Secure Digital) Memory Card:

- Up to 2GB storage
- Plastic rigged wafer about size of postage stamp
- Mini and micro sizes available

SDHC (High Capacity):

- Up to 32GB storage

SDXC (Extended Capacity):

- May exceed 1TB storage

Applications:

- Digital cameras
- Smartphones
- Music players
- Car navigation systems
- Host of other electronic appliances

1.4 REPRESENTING INFORMATION AS BIT PATTERNS**Representing Text**

Approach: Code in which each different symbol is assigned a unique bit pattern

- Text represented as long string of bits
- Successive patterns represent successive symbols

ASCII (American Standard Code for Information Interchange):

- Adopted by ANSI (American National Standards Institute)
- Uses bit patterns of length SEVEN
- Represents:
 - Upper and lowercase English alphabet letters
 - Punctuation symbols
 - Digits 0 through 9
 - Control information (line feeds, carriage returns, tabs)

8-bit ASCII:

- Extended by adding 0 at most significant end of each 7-bit pattern
- Each pattern fits conveniently into typical byte-size memory cell
- Provides 128 additional bit patterns (extra bit assigned value 1)
- Can represent symbols beyond English alphabet and punctuation

ISO Extensions to ASCII:

- Developed for major language groups
- Example: Western European languages
 - British pound symbol (£)
 - German vowels (ä, ö, ü)

Problems with Extended ASCII:

1. **Insufficient patterns:** Cannot accommodate alphabet of many Asian and Eastern European languages
2. **Single standard constraint:** Document constrained to one selected standard; cannot support multiple language groups

Unicode:

- Developed through cooperation of leading hardware/software manufacturers
- Uses unique pattern of up to **21 bits** per symbol
- Gained rapid support of computing community

UTF-8 (Unicode Transformation Format 8-bit):

- Original ASCII characters: 8 bits

- Thousands of additional characters (Chinese, Japanese, Hebrew): 16 bits
- More obscure Unicode symbols: 24 or 32 bits
- Ample room for future expansion

Text File:

- File consisting of long sequence of symbols encoded using ASCII or Unicode
- Manipulated by utility programs called **text editors** (or simply **editors**)

Text File vs. Word Processor File:

- **Text file:** Only character-by-character encoding of text
- **Word processor file:** Contains text + numerous proprietary codes (font changes, alignment information, etc.)

Representing Numeric Values

Problem with Encoded Characters:

- Storing value 25 as ASCII: 2 bytes (16 bits)
- Largest number storable in 16 bits using ASCII: 99
- Using binary notation: store any integer from 0 to 65535 in 16 bits

Binary Notation:

- Way of representing numeric values using only digits 0 and 1
- Contrast with decimal (base 10) system using digits 0-9

Analogy - Car Odometer with only digits 0 and 1:

- Sequence of readings: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, ...
- Represents binary representations of integers 0 through 8
- Bit pattern of 16 1s = value 65535

Numeric Storage Techniques:

- **Two's complement notation:** Common for whole numbers (positive and negative)
- **Floating-point notation:** For numbers with fractional parts ($4\frac{1}{2}$, $\frac{3}{4}$)

Representing Images

Pixel (Picture Element):

- Image interpreted as collection of dots
- Each dot = pixel

- Appearance of each pixel encoded
- Entire image = collection of encoded pixels

Bit Map:

- Collection of encoded pixels
- Popular because many display devices operate on pixel concept
- Images in bit map form easily formatted for display

Encoding Methods:

Image Type	Encoding Method
Simple black-and-white	Single bit per pixel (0=black, 1=white)
Black-and-white photographs	8 bits per pixel (256 shades of gray)
Color images	More complex systems

RGB Encoding:

- Each pixel = three color components: RED, GREEN, BLUE (primary colors of light)
- One byte for intensity of each color component
- Three bytes of storage per pixel

Luminance/Chrominance Encoding:

- **Luminance** (brightness): Sum of red, green, blue components (amount of white light)
- **Blue chrominance**: Difference between luminance and amount of blue light
- **Red chrominance**: Difference between luminance and amount of red light
- Origin: Color television broadcast
- Compatibility with older black-and-white receivers
- Gray-scale version using only luminance components

Disadvantage of Bit Maps:

- Image cannot be rescaled easily to arbitrary size
- Enlarging = making pixels bigger → grainy appearance
- **Digital zoom** vs. **Optical zoom**

Alternate Representation - Geometric Structures:

- Describe image as collection of geometric structures (lines, curves)

- Encoded using analytic geometry techniques
- Display device decides how to display geometric structures
- NOT forced to reproduce particular pixel pattern

Applications:

- **Scalable fonts:** TrueType (Microsoft/Apple), PostScript (Adobe Systems)
- **Computer-aided design (CAD):** Drawings of 3D objects displayed and manipulated

Drawing Software Systems (e.g., Microsoft Paint):

- User selects geometric shape from menu
- Directs drawing via mouse
- Software maintains geometric description during drawing
- As directions given: internal geometric representation modified → reconverted to bit map → displayed
- Allows easy scaling and shaping
- After drawing complete: geometric description discarded, only bit map preserved
- Some systems preserve geometric shapes for later modification

Representing Sound

Sampling Method:

- Sample amplitude of sound wave at regular intervals
- Record series of values obtained

Example: Series 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0

- Represents sound wave rising in amplitude, falling briefly, rising higher, dropping back to 0

Applications:

- Long-distance voice telephone communication: 8,000 samples per second
- Voice encoded as numeric values every eight-thousandth of a second
- Transmitted over communication line
- Receiving end reproduces sound

High-Fidelity Music:

- Musical CDs: 44,100 samples per second
- Each sample: 16 bits (32 bits for stereo)
- One second of stereo music: more than 1 million bits

MIDI (Musical Instrument Digital Interface):

- Alternative encoding system
- Widely used in:
 - Music synthesizers (electronic keyboards)
 - Video game sound
 - Sound effects for websites
- Encodes DIRECTIONS for producing music, NOT the sound itself
- Avoids large storage requirements of sampling technique

MIDI Encoding Example:

- Clarinet playing note D for two seconds = 3 bytes
- Same sound sampled at 44,100 samples/second = more than 2 million bits

Analogy:

- MIDI = encoding sheet music read by performer
- NOT the performance itself
- MIDI "recording" can sound significantly different on different synthesizers

1.5 THE BINARY SYSTEM**Binary Notation****Base 10 System:**

- Each position associated with a quantity
- $375 = 3 \times 100 + 7 \times 10 + 5 \times 1$
- Technical notation: $(3 \times 10^2) + (7 \times 10^1) + (5 \times 10^0)$

Base 2 System:

- Each position associated with quantity TWICE the quantity to its right
- Rightmost digit: quantity 1 (2^0)
- Next left: quantity 2 (2^1)
- Next: quantity 4 (2^2)
- Next: quantity 8 (2^3)
- And so on...

Decoding Binary Representation:

- Multiply each digit value by quantity associated with its position

- Add results
- Since binary uses only digits 0 and 1 → reduces to adding quantities associated with positions occupied by 1s

Example: $100101 = 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 32 + 4 + 1 = 37$

text

Algorithm for Finding Binary Representation of Positive Integer:

FIGURE 1.15

Figure 1.15 An algorithm for finding the binary representation of a positive integer

- Step 1.** Divide the value by two and record the remainder.

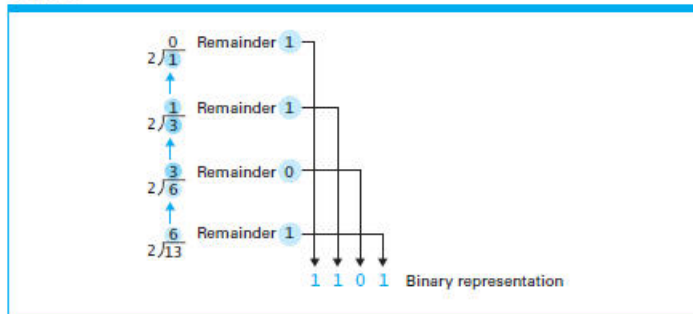
Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

Example - Converting 13 to Binary:

FIGURE 1.16

Figure 1.16 Applying the algorithm in Figure 1.15 to obtain the binary representation of thirteen



Binary Addition

Procedure (Same as Base 10):

1. Progress from RIGHT to LEFT
2. Add digits in each column
3. Write least significant digit of sum under column
4. Carry more significant digit to next column

Binary Addition Facts:

Example - $111010 + 11011$:

$$\begin{array}{r}
 111010 \\
 + 11011 \\
 \hline
 1010101
 \end{array}$$

Fractions in Binary

Radix Point: Same role as decimal point in decimal notation

- Digits to LEFT: integer part (interpreted as in binary system)
- Digits to RIGHT: fractional part

Positional Quantities to Right of Radix:

Position	Quantity
1st right	$\frac{1}{2} (2^{-1})$
2nd right	$\frac{1}{4} (2^{-2})$
3rd right	$\frac{1}{8} (2^{-3})$
nth right	$\frac{1}{2^n} (2^{-n})$

Decoding Binary Fraction:

- Multiply each bit value by quantity assigned to that bit's position
- Add results

Example - 101.101:

Binary Addition with Fractions:

- Align radix points
- Apply same addition process

Example - 10.011 + 100.11:

text

$$\begin{array}{r}
 10.011 \\
 + 100.110
 \end{array}$$

1 1 1 . 0 0 1

Analog vs. Digital

Digital System:

- Value encoded as series of digits
- Stored using several devices, each representing one digit
- Example: Empty bucket = digit 0, full bucket = digit 1
- Store numeric value in row of buckets using floating-point notation

Analog System:

- Each value stored in single device
- Device can represent any value within continuous range
- Example: Partially fill single bucket; water level = numeric value

Advantage of Digital:

- Less sensitive to error
- Significant sloshing required before distinction between full and empty is blurred
- Reason many analog technologies shifting to digital:
 - Telephone communication
 - Audio recordings
 - Television

1.6 STORING INTEGERS

Two's Complement Notation

Definition: Most popular system for representing integers within today's computers

- Uses fixed number of bits for each value in the system
- Common: 32-bit patterns
- Study smaller systems for demonstration

Two's Complement Systems (3-bit and 4-bit):

Construction Method:

1. Start with string of 0s of appropriate length
2. Count in binary until pattern consisting of single 0 followed by 1s is reached

- These patterns represent values 0, 1, 2, 3, ...
- 3. Patterns representing negative values:
 - Start with string of 1s of appropriate length
 - Count BACKWARD in binary until pattern consisting of single 1 followed by 0s is reached
 - These patterns represent values -1, -2, -3, ...

Sign Bit:

- Leftmost bit of bit pattern indicates sign of value represented
- **Sign bit = 1:** Negative values
- **Sign bit = 0:** Nonnegative values

Relationship Between Positive and Negative Patterns:

- Identical when read from RIGHT to LEFT, up to and including the first 1
- From there on, patterns are **complements** of one another
- **Complement:** Pattern obtained by changing all 0s to 1s and all 1s to 0s

Example (4-bit system):

- 2 = 0010
- -2 = 1110
- Both end with "10"
- 2 begins with "00", -2 begins with "11"

Conversion Algorithm (Positive ↔ Negative):

Decoding Two's Complement Representations:

Sign Bit	Procedure
0	Read value as though pattern were binary representation
1	Value is negative; apply "copy and complement" to find magnitude; decode as binary

Example - Decode 1010:

1. Sign bit = 1 → value is negative
2. Apply "copy and complement": 1010 → 0110
3. 0110 is binary for 6
4. Original pattern represents -6

Addition in Two's Complement Notation

Procedure:

- Apply same algorithm as binary addition
- All bit patterns (including answer) are SAME LENGTH
- Any extra bit generated on left of answer by final carry must be TRUNCATED

Examples:

Major Benefit:

- Addition of ANY combination of signed numbers uses SAME algorithm
- SAME circuitry
- Machine only needs to know how to ADD
- Subtraction = addition of negative number

Example - 7 - 5:

1. Change 5 (0101) to -5 (1011)
2. Perform addition: $0111 + 1011 = 10010 \rightarrow$ truncate to $0010 = 2$

Result: Circuit for addition + circuit for negating value = sufficient for both addition and subtraction

Overflow

Definition: Problem that occurs when computation produces value outside range of representable values

4-bit Two's Complement Limits:

- Largest positive integer: 7
- Most negative integer: -8
- Value 9 cannot be represented

Example - $5 + 4 = 9$:

- Cannot be represented
- Result would appear as -7 (incorrect)

When Overflow Occurs:

- Adding two positive values
- Adding two negative values

Detection:

- Check sign bit of answer

- Overflow indicated if:
 - Addition of two positive values → pattern for negative value
 - Addition of two negative values → pattern appears positive

Modern Systems:

- 32-bit patterns
- Positive values up to 2,147,483,647 before overflow
- Larger values: longer bit patterns or change units of measure

Historical Example - September 19, 1989:

- Hospital computer system malfunctioned
- Date was 32,768 days after January 1, 1900
- Program used 16-bit two's complement
- Overflow produced negative value
- Program not designed to handle this

Excess Notation

Definition: Another method of representing integer values

- Each value represented by bit pattern of same length
- First pattern with 1 as most significant bit = zero
- Patterns following = 1, 2, 3, ...
- Patterns preceding = -1, -2, -3, ...

Key Property:

- Binary interpretation of pattern EXCEEDS the excess notation interpretation by the excess value

Excess Eight Example:

- Pattern 1100 = binary 12, excess eight represents 4 ($12 - 8 = 4$)
- Pattern 0000 = binary 0, excess eight represents -8 ($0 - 8 = -8$)

Excess System Naming:

- Excess eight (4-bit) = $2^3 = 8$
- Excess sixteen (5-bit) = $2^4 = 16$
- Excess four (3-bit) = $2^2 = 4$

Difference from Two's Complement:

- Sign bits are REVERSED

1.7 STORING FRACTIONS

Floating-Point Notation

Definition: Popular way of storing values with fractional parts

- Based on scientific notation
- Stores both binary representation AND position of radix point

8-bit Floating-Point Format (Example):

- 1 byte of storage
- High-order bit = **sign bit** (0 = nonnegative, 1 = negative)
- Remaining 7 bits divided into two groups:
 - **Exponent field** (3 bits following sign bit)
 - **Mantissa field** (remaining 4 bits)

Decoding Process:

1. Extract mantissa, place radix point on its left → .1011
2. Extract exponent field (110), interpret as integer using 3-bit excess notation → +2
3. Move radix right by 2 bits → 10.11
4. Convert to decimal → $2\frac{3}{4}$
5. Apply sign bit (0) → $+2\frac{3}{4}$

Example - 00111100:

1. Mantissa: .1100
2. Exponent: 011 (excess four = -1)
3. Move radix left 1 bit → .01100
4. Value = $\frac{3}{8}$
5. Sign bit 0 → $+\frac{3}{8}$

Encoding Process:

1. Express value in binary
2. Copy bit pattern into mantissa field from LEFT to RIGHT, starting with leftmost 1
3. Determine how many bits and direction radix must be moved to obtain original number
4. Fill exponent field with appropriate excess notation value
5. Fill sign bit (0 for nonnegative, 1 for negative)

Example - Encode $1\frac{1}{2}$:

1. 1% in binary = 1.001
2. Mantissa: copy from leftmost 1 $\rightarrow$ 1001 (fills 4 bits)
3. Radix in 1.001 must move 1 bit RIGHT to get 1.001 $\rightarrow$ exponent = +1
4. Excess four: +1 = 101
5. Sign bit = 0
6. Result: 0 101 1001 = 01011001

Normalized Form:

- Mantissa field filled starting with leftmost 1 in binary representation
- Eliminates possibility of multiple representations for same value
- All nonzero values have mantissa starting with 1
- Value zero = special case (all 0s)

Truncation Errors

Definition: Part of value being stored is lost because mantissa field is not large enough

Example - Encoding 2% :

- 2% in binary = 10.101
- Copy to mantissa (starting with leftmost 1) $\rightarrow$ 1010 (only 4 bits)
- Rightmost 1 (representing last $\frac{1}{8}$) is LOST
- Result: 01101010 represents $2\frac{1}{2}\%$, NOT 2%

Solution: Use longer mantissa field

- Modern computers: at least 32 bits for floating-point
- Even with longer formats, truncation errors still occur

Nonterminating Expansions:

- Some values cannot be accurately expressed regardless of how many digits used
- More values have nonterminating representations in binary than in decimal
- Example: $1/10$ is nonterminating in binary

Implication:

- Using floating-point to store dollars and cents $\rightarrow$ value of a dime cannot be stored accurately
- Solution: Manipulate data in units of pennies (integers) $\rightarrow$ accurately stored using two's complement

Numerical Analysis:

- Branch of mathematics dealing with problems in actual computations
- Often massive and require significant accuracy

Example - Adding $2\frac{1}{2} + \frac{1}{8} + \frac{1}{8}$:

Order 1 (Large + Small):

1. $2\frac{1}{2} + \frac{1}{8} = 2\frac{5}{8}$ (cannot store accurately) $\rightarrow$ stored as $2\frac{1}{2}$
2. Add last $\frac{1}{8} \rightarrow$ truncation error again $\rightarrow$ final result $2\frac{1}{2}$ (INCORRECT)

Order 2 (Small + Small then Large):

1. $\frac{1}{8} + \frac{1}{8} = \frac{1}{4}$ (stored accurately)
2. $2\frac{1}{2} + \frac{1}{4} = 2\frac{3}{4}$ (stored accurately) $\rightarrow$ final result $2\frac{3}{4}$ (CORRECT)

Conclusion: Order of addition matters in floating-point

- If very large number added to very small number $\rightarrow$ small number may be truncated
- General rule: Add smaller values first

Commercial Software:

- Good job shielding uneducated user from such problems
- Typical spreadsheet: correct answers unless values differ by factor of 10^{16} or more
- Adding 1 to 10,000,000,000,000,000 might get 10,000,000,000,000,000 (not 10,000,000,000,000,001)

Significant Applications:

- Navigational systems
- Minor errors compounded in additional computations
- Can produce significant consequences

Single Precision vs. Double Precision

Single Precision Floating Point (32-bit):

- 1 bit for sign
- 8 bits for exponent (excess notation)
- 23 bits for mantissa
- Capable of expressing:
 - Very large numbers: order of 10^{38}
 - Very small numbers: order of 10^{-37}
- Precision: 7 decimal digits

- First 7 digits stored with very good accuracy
- Any digits past first 7 are lost by truncation error

Double Precision Floating Point (64-bit):

- Provides precision of 15 decimal digits

PART II: DATA MANIPULATION (Chapter 2.1-2.5)

2.1 COMPUTER ARCHITECTURE

CPU (Central Processing Unit):

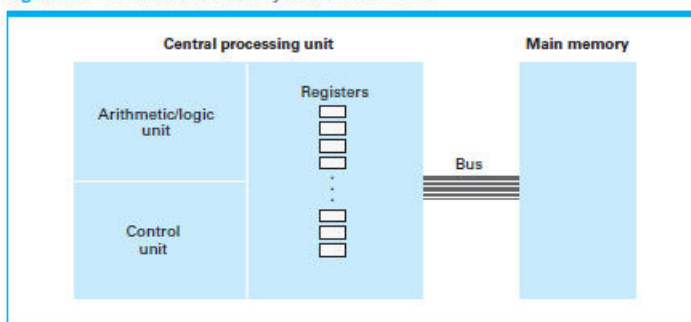
- Circuitry that controls manipulation of data
- Also called **processor**
- Mid-20th century: large units (several racks of electronic circuitry)
- Today: small flat squares (~2 inches × 2 inches)
- Connecting pins plug into socket on **motherboard**
- Smartphones, mini-notebooks, Mobile Internet Devices (MID): ~half size of postage stamp
- Called **microprocessors** due to small size

CPU Basics

Three Parts of CPU:

FIGURE 2.1

Figure 2.1 CPU and main memory connected via a bus



Component	Function
Arithmetic/Logic Unit (ALU)	Circuitry that performs operations on data (addition, subtraction)
Control Unit	Circuitry for coordinating machine's activities
Register Unit	Data storage cells within CPU (called registers) for temporary storage

Registers:

- **General-purpose registers:** Temporary holding places for data being manipulated by CPU
- **Special-purpose registers:** Discussed in Section 2.3

ALU Operation Process:

1. Control unit transfers data from main memory into general-purpose registers
2. Informs ALU which registers hold the data
3. Activates appropriate circuitry within ALU
4. Tells ALU which register should receive the result

Bus:

- Collection of wires connecting CPU and main memory
- CPU **reads** data from memory:
 - Supplies address of pertinent memory cell
 - Sends electronic signal telling memory circuitry to retrieve data in indicated cell
- CPU **writes** data to memory:
 - Provides address of destination cell
 - Provides data to be stored
 - Sends appropriate electronic signal

Task of Adding Two Values Stored in Memory:

The Stored-Program Concept

Early Computers:

- Steps built into control unit as part of machine
- Flexibility achieved by rewiring CPU

- Pegboard arrangement (old telephone switchboards)

Breakthrough:

- Program, like data, can be encoded and stored in main memory
- Control unit extracts program from memory, decodes instructions, executes them
- Program changed by changing contents of memory, NOT rewiring CPU

Stored-Program Concept:

- Standard approach used today
- Seems obvious now
- Originally difficult because programs and data were thought of as different entities
- Data in memory; programs part of CPU

Credit Dispute:

- Often credited to John von Neumann (incorrectly)
- Actually developed by researchers led by J.P. Eckert at Moore School of Electrical Engineering, University of Pennsylvania
- von Neumann was first to publish the idea

Cache Memory

Memory Hierarchy by Functionality:

Memory Level	Purpose
Registers	Hold data immediately applicable to operation at hand
Cache Memory	High-speed memory within CPU; copy of portion of main memory of current interest
Main Memory	Hold data that will be needed in near future
Mass Storage	Hold data not needed in immediate future

Cache Operation:

- Data transfers normally between registers and main memory → made between registers and cache memory
- Changes to cache memory transferred collectively to main memory at more opportune time
- Result: CPU executes machine cycle more rapidly (not delayed by main memory communication)

2.2 MACHINE LANGUAGE

Definition: Collection of instructions along with encoding system that CPUs recognize

- **Machine-level instruction / Machine instruction:** Instruction expressed in machine language

The Instruction Repertoire

Key Insight:

- Once machine can perform certain elementary but well-chosen tasks
- Adding more features does NOT increase theoretical capabilities
- Additional features may increase convenience but add nothing to fundamental capabilities

Two Philosophies:

RISC (Reduced Instruction Set Computer)

CISC (Complex Instruction Set Computer)

CPU designed to execute minimal set of machine instructions

CPU with ability to execute large number of complex instructions

Efficient, fast, less expensive to manufacture

Technically redundant instructions

ARM-based processors (Qualcomm, Texas Instruments)

Intel processors

Low power consumption

Insatiable thirst for electrical power

Found in game controllers, digital TVs, navigation systems, smartphones

Found in desktop and laptop computers

Three Categorizations of Machine Instructions:

Group

Function

Data Transfer Group

Request movement of data from one location to another

Arithmetic/Logic Group

Tell control unit to request activity within ALU

Control Group

Direct execution of program rather than data manipulation

1. Data Transfer Group:

Terminology Note:

- "Transfer" or "move" is misnomer
- Data is rarely erased from original location
- More like COPYING or CLONING

LOAD Instruction:

- Request to fill general-purpose register with contents of memory cell

STORE Instruction:

- Request to transfer contents of register to memory cell

I/O Instructions:

- Commands for communicating with devices outside CPU-main memory context
- Printers, keyboards, display screens, disk drives
- Sometimes considered separate category
- Can be handled by same instructions that request data transfers between CPU and main memory

2. Arithmetic/Logic Group:

- Instructions that tell control unit to request activity within ALU
- Basic arithmetic operations (ADD, SUBTRACT, MULTIPLY, DIVIDE)
- Boolean operations (AND, OR, XOR)
- SHIFT operations (move bits right/left, discard bits that "fall off")
- ROTATE operations (move bits right/left, bits that "fall off" fill holes at other end)

3. Control Group:

- Instructions that direct program execution
- **JUMP (BRANCH) instructions:** Direct CPU to execute instruction other than next one in list

Two Varieties:

Type

Description

Unconditional Jump

Always execute jump (e.g., "Skip to Step 5")

Type	Description
Conditional Jump	Jump only if certain condition is satisfied (e.g., "If value obtained is 0, skip to Step 5")

Illustrative Machine Language (Appendix C)

Machine Architecture:

- 16 general-purpose registers (numbered 0 through F in hexadecimal)
- Each register: 1 byte (8 bits)
- 256 main memory cells (addresses 00 through FF in hexadecimal)
- Each memory cell: 8 bits capacity

Instruction Format:

- Each instruction: 16 bits (4 hexadecimal digits)
- **Op-code field:** First 4 bits (first hexadecimal digit)
- **Operand field:** Last 12 bits (3 hexadecimal digits)

Op-codes: Hexadecimal digits 1 through C

Why Main Memory Capacities are Powers of Two:

- 8 bits in instruction reserved for specifying memory cell
- Can reference exactly $2^8 = 256$ different memory cells
- Build main memory with this many cells (addressed 0-255)
- More cells → cannot distinguish between them
- Fewer cells → instructions reference nonexisting cells

Example - Instruction 70C5:

- Op-code 7 = OR operation
- Next hex digit (0) = destination register
- Last two hex digits (C5) = registers to OR (C and 5)
- Translation: "OR the contents of register C with the contents of register 5 and leave the result in register 0"

Two LOAD Instructions:

Op-code	Type	Operand Contains
1	Load register with contents of memory cell	Address

Op-code	Type	Operand Contains
2	Load register with particular value	Actual bit pattern to be loaded

Two ADD Instructions:

Op-code	Type
5	Add two's complement representations
6	Add floating-point representations

Reason: Adding bit patterns representing two's complement vs. floating-point requires different activities within ALU

Encoded Version of Addition Program:

FIGURE 2.7

Figure 2.7 An encoded version of the instructions in Figure 2.2

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

2.3 PROGRAM EXECUTION

Process:

1. Copy instructions from memory into CPU as needed
2. Once in CPU, each instruction is decoded and obeyed
3. Order of fetching = order of storage in memory (unless altered by JUMP)

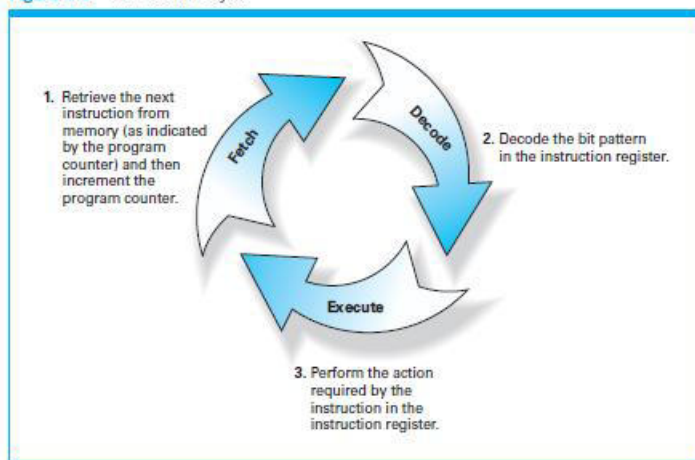
Special Purpose Registers within CPU:

Register	Purpose
Instruction Register	Holds the instruction being executed
Program Counter	Contains address of NEXT instruction to be executed

The Machine Cycle

FIGURE 2.8

Figure 2.8 The machine cycle



Step 1 - FETCH:

1. CPU requests main memory to provide instruction stored at address indicated by program counter
2. For our machine: each instruction is 2 bytes → retrieve contents of 2 consecutive memory cells
3. CPU places instruction received in **instruction register** (16 bits long)
4. CPU increments program counter by 2 → contains address of next instruction in memory

Step 2 - DECODE:

- CPU analyzes instruction in instruction register
- Breaks operand field into proper components based on op-code

Step 3 - EXECUTE:

- CPU activates appropriate circuitry to perform requested task
- **LOAD from memory:** Send signals to main memory, wait for data, place data in requested register

- **Arithmetic operation:** Activate appropriate ALU circuitry with correct registers as inputs, wait for result

JUMP Instruction Execution:

- Example: B258 = "JUMP to instruction at address 58 if contents of register 2 equals that of register 0"
- Execute step begins with comparison of registers 2 and 0
- If different → execute step terminates, next machine cycle begins
- If equal → machine places value 58 in program counter during execute step
- Next fetch step finds 58 in program counter → instruction at that address is fetched next

Special Case - B058:

- Decision depends on whether contents of register 0 equals contents of register 0
- Same registers → must have equal content
- Any instruction of form B0XY will cause jump to memory location XY regardless of contents of register 0

Comparing Computer Power

Clock:

- Circuit (oscillator) that generates pulses to coordinate machine's activities
- Faster oscillating circuit → faster machine cycle

Clock Speed Measurement:

- **Hertz (Hz):** One cycle (pulse) per second
- **MHz (megahertz):** Million Hz
- **GHz (gigahertz):** 1000 MHz
- Typical desktop computers: few hundred MHz (older) to several GHz

Problem:

- Different CPU designs may perform different amounts of work in one clock cycle
- Clock speed alone fails to compare machines with different CPUs
- Intel processor vs. ARM processor → clock speed not relevant

Solution - Benchmarking:

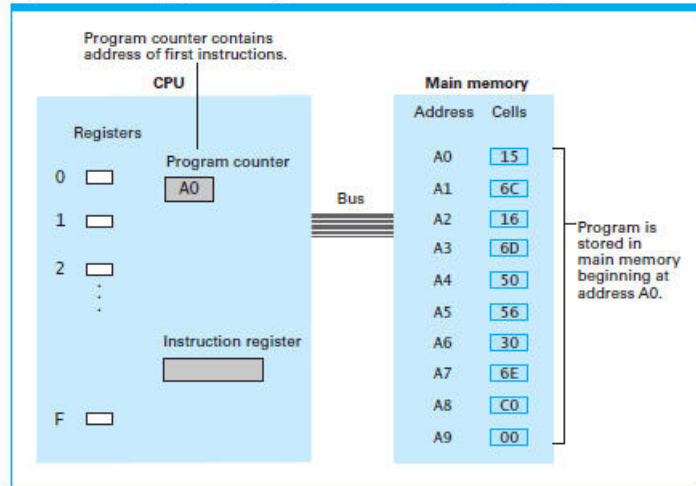
- Process of comparing performance of different machines when executing same program
- **Benchmark:** Program used for this comparison

- Selecting benchmarks representing different applications → meaningful comparisons for various market segments

Example of Program Execution

Program from Figure 2.7 stored in memory starting at address A0:

Figure 2.10 The program from Figure 2.7 stored in main memory ready for execution



Execution Trace:

Cycle 1 - Fetch:

- Program counter = A0
- CPU extracts instruction from memory cells A0-A1: 156C
- Places 156C in instruction register
- Program counter incremented to A2

Cycle 1 - Decode & Execute:

- Decode: LOAD register 5 with contents of memory cell at address 6C
- Execute: Register 5 loaded

Cycle 2 - Fetch:

- Program counter = A2
- Fetch instruction 166D from A2-A3
- Place in instruction register
- Program counter incremented to A4

Cycle 2 - Decode & Execute:

- Decode: LOAD register 6 with contents of memory cell at address 6D

- Execute: Register 6 loaded

Cycle 3 - Fetch:

- Program counter = A4
- Fetch instruction 5056 from A4-A5
- Place in instruction register
- Program counter incremented to A6

Cycle 3 - Decode & Execute:

- Decode: ADD registers 5 and 6 (two's complement), result in register 0
- Execute: ALU performs addition, leaves result in register 0

Cycle 4 - Fetch:

- Program counter = A6
- Fetch instruction 306E from A6-A7
- Place in instruction register
- Program counter incremented to A8

Cycle 4 - Decode & Execute:

- Decode: STORE register 0 in memory cell at address 6E
- Execute: Sum placed in memory location 6E

Cycle 5 - Fetch:

- Program counter = A8
- Fetch instruction C000 from A8-A9
- Place in instruction register
- Program counter incremented to AA

Cycle 5 - Decode & Execute:

- Decode: HALT instruction
- Execute: Machine stops

Programs Versus Data

Multiple Programs:

- Many programs can be stored simultaneously in main memory
- As long as they occupy different locations

- Which program runs when machine is started = determined by setting program counter appropriately

Problem:

- Data also contained in main memory
- Encoded in terms of 0s and 1s
- Machine alone has no way of knowing what is data and what is program
- If program counter assigned address of data → CPU extracts data bit patterns as instructions and executes them
- Result depends on data involved

Advantage:

- Common appearance of programs and data in memory is USEFUL
- Allows one program to manipulate other programs (or itself) the same as data
- Example: Program that modifies itself in response to environment → exhibits ability to learn
- Program that writes and executes other programs to solve problems

2.4 ARITHMETIC/LOGIC INSTRUCTIONS

Logic Operations

Bitwise Operations:

- Extend basic Boolean operations to combine two strings of bits
- Apply basic operation to individual columns
- Produce single output string

Example - AND Operation:

text

10011010

AND 11001001

10001000

Example - OR Operation:

text

10011010

OR 11001001

11011011

Example - XOR Operation:

text

10011010

XOR 11001001

01010011

Major Use of AND - Masking:

- Placing 0s in one part of bit pattern while not disturbing the other part
- **Mask:** One operand that determines which part of other operand will affect result
- Result = partial replica of one operand, with 0s occupying nonduplicated positions

Example:

text

00001111 (mask)

AND 10101010

00001010

- Four most significant bits = 0s
- Four least significant bits = copy of second operand

Applications:

- Filtering certain colors out of digital image in RGB format
- Isolating specific bits in bit map

Bit Map Example - 52 bits representing playing cards:

- Each bit associated with particular playing card
- Poker hand: 1s for five cards in hand, 0s for others
- Bridge hand: 13 ones
- Ice cream flavors: 32 bits

Finding if specific object is present:

- AND byte with mask (e.g., 00100000 for third bit from high-order end)

- Result = all 0s if and only if that bit is 0
- Follow AND operation with conditional branch instruction

Changing a 1 to 0 without disturbing other bits:

- AND bit map with mask (e.g., 11011111 for third bit from high-order end)
- Store result in place of original

OR Operation:

- Duplicate part of string while putting 1s in nonduplicated part
- Mask: 0s indicate positions to duplicate, 1s indicate nonduplicated positions

Example:

text

11110000 (mask)

OR 10101010

11111010

- Most significant four bits = 1s
- Remaining bits = copy of least significant four bits of other operand

XOR Operation:

- Can invert all bits of an RGB bitmap image
- Results in "inverted" color image (light colors replaced by dark, vice versa)

Example:

text

11111111 (mask)

XOR 10101010

01010101

Machine Language (Appendix C):

Op-code	Operation
7	OR

Op-code	Operation
8	AND
9	XOR

Format: 7ABC = OR contents of registers B and C, result in register A

Rotation and Shift Operations

Purpose: Moving bits within a register

- Often used in solving alignment problems

Classification:

- Direction of motion (RIGHT or LEFT)
- Whether process is CIRCULAR

Circular Shift (Rotation):

- Bit that falls off one end is placed in hole at other end
- Right circular shift on byte-size pattern 8 times → original pattern

Logical Shift:

- Discard bit that falls off edge
- Always fill hole with 0
- Left shift = multiplication by two (binary equivalent of decimal shift)
- Right shift = division by two

Arithmetic Shift:

- Right shifts that always fill hole (sign bit position) with its original value
- Leave sign bit unchanged

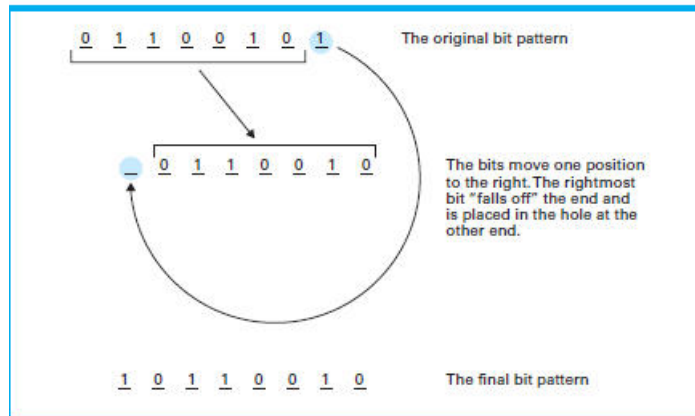
Machine Language (Appendix C):

- Only right circular shift (op-code A)
- Format: A501 = "Rotate contents of register 5 to the right by 1 bit"

Example - Rotating 65 (hexadecimal) one bit right:

FIGURE 2.12

Figure 2.12 Rotating the bit pattern 65 (hexadecimal) one bit to the right



Note: Right circular shift of 3 bits = left circular shift of 5 bits (since register is 8 bits long)

Arithmetic Operations

Simulation:

- Subtraction = addition + negation
- Multiplication = repeated addition
- Division = repeated subtraction
- Some small CPUs designed with only ADD (or ADD and SUBTRACT) instructions

Multiple ADD Instructions:

- Two's complement addition: straightforward column-by-column addition
- Floating-point addition: extract mantissa, shift right/left according to exponent fields, check sign bits, perform addition, translate result to floating-point notation
- Both called "addition" but machine action is NOT the same

2.5 COMMUNICATING WITH OTHER DEVICES

Computer Core: Main memory + CPU

Peripheral Devices:

- Mass storage systems
- Printers
- Keyboards
- Mice
- Display screens
- Digital cameras

- Other computers

The Role of Controllers

Controller:

- Intermediary apparatus between computer and other devices
- Circuitry permanently mounted on motherboard OR circuit board that plugs into slot
- Connects via cables to peripheral devices or to **port** (connector on back of computer)
- Some controllers are small computers themselves (own memory circuitry and simple CPU)

Function:

- Translates messages and data back and forth between:
 - Forms compatible with internal characteristics of computer
 - Forms compatible with peripheral device

Historical:

- Each controller designed for particular type of device
- Purchasing new peripheral device often required purchasing new controller

Modern Standards:

Standard	Description
USB (Universal Serial Bus)	Single controller handles variety of devices; lower cost; USB 2.0, USB 3.0
FireWire	Faster transfer rate; video recorders, online mass storage

USB-Compatible Devices:

- Mice, printers, scanners, mass storage devices, digital cameras, smartphones

Controller Communication:

- Connects to same bus that connects CPU and main memory
- Monitors signals between CPU and main memory
- Injects its own signals onto bus

Memory-Mapped I/O:

- CPU communicates with controllers same way it communicates with main memory

- To send bit pattern to controller:
 1. Bit pattern constructed in one of CPU's general-purpose registers
 2. Instruction similar to STORE executed to "store" bit pattern in controller
- To receive bit pattern from controller:
 - Instruction similar to LOAD used
- Each controller designed to respond to unique set of addresses
- Main memory designed to ignore references to these locations
- When CPU sends message to store bit pattern at memory location assigned to controller → bit pattern actually "stored" in controller, NOT main memory
- When CPU tries to read data from such memory location → receives bit pattern from controller, NOT from memory

Alternative - I/O Instructions:

- Provide special op-codes in machine language to direct transfers to/from controllers
- Example: F5A3 = "STORE contents of register 5 in controller identified by bit pattern A3"

Direct Memory Access (DMA)

Definition: Controller's ability to carry on its own communication with main memory during nanoseconds when CPU is not using the bus

Example - Disk Data Retrieval:

1. CPU sends requests encoded as bit patterns to disk controller
2. CPU asks controller to read sector and place data in specified area of main memory
3. CPU continues with other tasks while controller performs read operation
4. Controller deposits data in main memory via DMA
5. Two activities performed simultaneously:
 - CPU executing program
 - Controller overseeing data transfer between disk and main memory
6. Computing resources of CPU not wasted during relatively slow data transfer

Detrimental Effect:

- Complicates communication over computer's bus
- Bit patterns must move between:
 - CPU and main memory
 - CPU and each controller

- Each controller and main memory
- Coordination of all activity on bus = major design issue

Von Neumann Bottleneck:

- Central bus can become impediment as CPU and controllers compete for bus access
- Consequence of underlying von Neumann architecture
- CPU fetches instructions from memory over central bus

Handshaking

Definition: Constant two-way dialogue between computer and peripheral device

- Exchange information about device's status
- Coordinate activities

Why Needed:

- Computer can produce and send characters to printer much faster than printer can print them
- If computer blindly sent data → printer would quickly fall behind → lost data

Status Word:

- Bit pattern generated by peripheral device and sent to controller
- Bit map where bits reflect conditions of device

Printer Status Word Example:

Bit	Meaning
Least significant bit	Printer out of paper?
Next bit	Printer ready for additional data?
Another bit	Paper jam present?

Controller Action:

- May respond to status information itself
- Or make it available to CPU

Popular Communication Media

Two Types of Paths:

Type	Description	Characteristics
Parallel Communication	Several signals transferred at same time, each on separate "line"	Rapid data transfer; complex communication path; computer's internal bus
Serial Communication	Signals transferred one after another over single line	Simpler data path; popular; USB, FireWire, Ethernet

Short Distance:

- USB and FireWire: relatively high-speed data transfer
- Short distances of only a few meters
- Ethernet (wire or radio broadcast): within home/office building

Greater Distance:

- Traditional voice telephone lines
- Inherently serial systems
- Single wire over which tones are transferred one after another

Modem (Modulator-Demodulator):

- Converts bit patterns into audible tones
- Transfers tones serially over telephone system
- Converts tones back into bits at destination

DSL (Digital Subscriber Line):

- Faster long-distance communication over traditional telephone lines
- Takes advantage of existing telephone lines capable of handling wider frequency range than traditional voice communication
- Uses frequencies ABOVE audible range to transfer digital data
- Leaves lower-frequency spectrum for voice communication
- Transfer rates on order of 54 Mbps

Fiber-Optic Lines:

- Telephone companies rapidly upgrading systems
- Support digital communication more readily than traditional telephone lines

Cable Modems:

- Competing technology
- Modulate and demodulate bit patterns for transmission over cable television systems

- Cable providers use both fiber-optic lines and traditional coaxial cable
- Provide both high definition television signals and computer network access

Satellite Links:

- High-frequency radio broadcast
- Computer network access possible in remote locations
- Far from high-speed telephone and cable television networks

Communication Rates

bps (bits per second):

- Rate at which bits are transferred from one computing component to another

Common Units:

Unit	Abbreviation	Value
Kilo-bps	Kbps	1,000 bps
Mega-bps	Mbps	1,000,000 bps
Giga-bps	Gbps	1,000,000,000 bps

Important Distinction:

- Lowercase b = bit
- Uppercase B = byte
- 8 Kbps = 1 KB per second

Short Distance Communication:

- USB 2.0 and FireWire: several hundred Mbps
- Sufficient for most multimedia applications
- Popular for communication between home computers and local peripherals
- Printers, external disk drives, cameras

Traditional Voice Telephone:

- Multiplexing (encoding/interweaving data so single communication path serves purpose of multiple paths)
- Data compression techniques

- Supported transfer rates of 57.6 Kbps
- Falls short of today's multimedia and Internet applications

Bandwidth Requirements:

- MP3 music recordings: ~64 Kbps
- Low quality video clips: Mbps range
- High definition video streaming (Netflix, YouTube): significantly higher

Alternatives:

- DSL, cable, satellite links
- Transfer rates well into Mbps range
- Replaced traditional audio telephone systems

Bandwidth:

- Maximum rate available in particular setting
- Depends on type of communication path and technology used in implementation
- Often loosely equated to transfer rate
- Also connotations of CAPACITY rather than transfer rate
- High bandwidth = broadband service
- Ability to transfer bits at high rate AND carry large amounts of information simultaneously

SUMMARY - KEY TERMS TO REMEMBER (MODULE 1)

Data Storage

Term	Definition
Bit	Binary digit; smallest unit of data (0 or 1)
Byte	String of eight bits
Boolean Operation	Operation manipulating true/false values
Gate	Device producing Boolean operation output
Flip-Flop	Fundamental unit of computer memory

Term	Definition
Hexadecimal Notation	Shorthand for bit patterns; one symbol per 4 bits
Main Memory	Large collection of circuits storing bits
RAM	Random Access Memory; cells accessed independently
DRAM	Dynamic RAM; requires refresh circuit
Mass Storage	Secondary storage (magnetic, optical, flash)
Sector	Small arc on disk track
Seek Time	Time to move read/write head to track
Latency Time	Half rotation time; average time for data to rotate to head
Access Time	Seek time + latency time
Transfer Rate	Rate of data transfer to/from disk
Flash Memory	Nonvolatile storage using trapped electrons
SSD	Solid-state disk; flash memory replacing hard disks
ASCII	American Standard Code for Information Interchange
Unicode	21-bit code for international symbols
Pixel	Picture element
Bit Map	Collection of encoded pixels
RGB Encoding	Red, Green, Blue color components

Term	Definition
Binary Notation	Base 2 number system
Two's Complement	Notation for signed integers
Floating-Point	Notation for fractions
Sign Bit	Leftmost bit indicating sign (0=positive, 1=negative)
Overflow	Result outside representable range
Excess Notation	Alternative integer representation
Mantissa	Fraction part in floating-point
Exponent	Power in floating-point
Normalized Form	Mantissa starts with 1
Truncation Error	Loss of bits due to limited mantissa

Data Manipulation

Term	Definition
CPU	Central Processing Unit; controls data manipulation
ALU	Arithmetic/Logic Unit; performs operations
Control Unit	Coordinates machine activities
Register	Storage cell within CPU
Bus	Wires connecting CPU and main memory

Term	Definition
Stored-Program Concept	Program encoded and stored in memory
Cache Memory	High-speed memory within CPU
Machine Language	Instruction encoding system
Op-code	Operation code; first 4 bits of instruction
Operand	Detailed information for operation
LOAD	Transfer from memory to register
STORE	Transfer from register to memory
Program Counter	Address of next instruction
Instruction Register	Holds current instruction
Machine Cycle	Fetch-Decode-Execute cycle
JUMP	Execute instruction out of sequence
Conditional Jump	Jump only if condition satisfied
Unconditional Jump	Always jump
Mask	Operand for selecting bits
AND	Bitwise AND operation
OR	Bitwise OR operation
XOR	Bitwise exclusive OR

Term	Definition
Rotation	Circular shift
Logical Shift	Shift with zero fill
Arithmetic Shift	Shift preserving sign bit
Controller	Intermediary between computer and device
Memory-Mapped I/O	Devices appear in memory addresses
DMA	Direct Memory Access; controller accesses memory directly
Handshaking	Two-way dialogue with device
Status Word	Bit pattern indicating device status
Parallel Communication	Multiple signals simultaneously
Serial Communication	Single signal sequentially
Modem	Modulator-Demodulator
DSL	Digital Subscriber Line
bps	Bits per second
Bandwidth	Maximum transfer rate/capacity