Multiplication

- Binary multiplication:

Multiplicand	1011
Multiplier	x 101
Partial Products	1011
	0000 -
	1011
Sum	110111

- Octal multiplication:

Multiplicand	762
Multiplier	x 45
Partial Products	4672
	3710 -
Sum	43772

BCD Arithmetic

- Given a BCD code, first use binary arithmetic to add the digits.
- If the digit sum > 9, then **add 6** (110) to the sum.

8	1000	Eight
+5	+0101	Plus 5
13	1101	13 (> 9)
	+0110	so add 6
	($C = 1$) 0011	leaving 3 and carry C
	00010011	Final answer

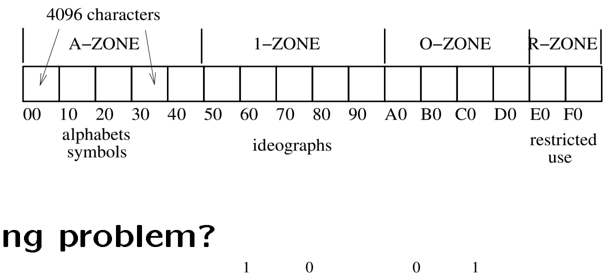
Error-Detection Codes

- **Redundancy** (e.g., extra information) can be added to a binary code word for the purpose of **detecting** and **correcting** errors.
- **Parity:** An extra bit is appended onto the code word to make the sum of bits odd or even.
  - **Even parity:** the sum of bits is even.
  - **Odd parity:** the sum of bits is odd.
- Parity can be used to **detect** all single-bit errors and some multiple-bit errors.
- Exp: Parity codes for 2 data bits

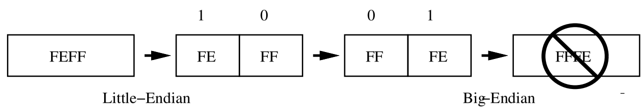
Even Parity	Odd Parity
Message - Parity	Message - Parity
00 - 0	00 - 1
01 - 1	01 - 0
10 - 1	10 - 0
11 - 0	11 - 1

ASCII: American Standard Code for Information Interchange

- Uses 7 bits to represent 94 graphic printing and 34 non-printing characters. (Refer to Table 1-4 in the text.)
  - Digits 0 to 9 are consecutive and span hexadecimal values 30_{16} to 39_{16}.
  - Upper case letters A–Z are consecutive and span 41_{16} to $5A_{16}$.
  - Lower case letters a–z are consecutive and span 61_{16} to $7A_{16}$.
  - Lower to upper case translation (and vice versa) occurs by complementing bit 6.
- **UNICODE** extends ASCII to 65536 (16-bit) universal characters codes.

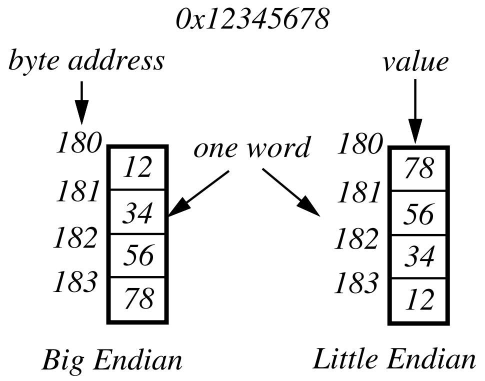


- **Byte-ordering problem?**



Byte Ordering

- **Big Endian:** Byte 0 is placed on the most significant end ($xx \ldots 00$ = Big end of word)
  - IBM 360/370, Motorola 68K, MIPS, SPARC, HP PA
  - Advantages: Consistent integers and character strings ordering, faster in character-string sorting, decimal/ASCII dumps.
- **Little Endian:** Byte 0 is placed on the least significant end ($00 \ldots xx$ = Little end of word)
  - Intel 80x86, Pentium, DEC VAX
  - Advantages: Integer address conversion, arithmetic.



Binary Logic

- **Binary variables:** one of two states.
  - True/False, On/Off, 1/0; variables are given names: $A, B, x, y, \ldots$
- **Logical operators:** operate on binary variables.
- **Logical operations:** AND, OR, NOT, etc.
  - **AND** ($\cdot$): $A \cdot B = Y$: "A AND B is equal to Y" (often omit $\cdot$)
  - **OR** ($+$): $x + y = z$: "x OR y is equal to z"
  - **NOT** ($'$ or overline): $A' = X$: "A NOT is equal to X"
  - Note: $1+1=2$: "one <u>**plus**</u> one equals two"; $1+1=1$: "one <u>**or**</u> one equals one";
- **Truth table:** Truth tables list the **output** of a function for **all possible inputs**.

AND

X	Y	Z=X*Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

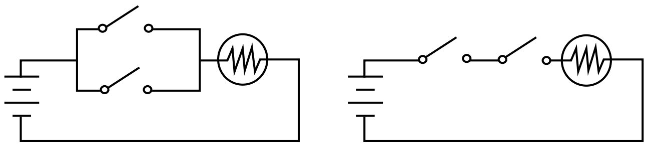
X	Y	Z=X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	$Z=\overline{X}$
0	1
1	0

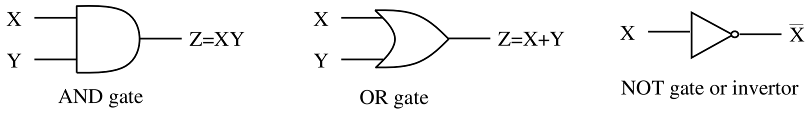
Logic Gates

- **Logic gates:** physical devices that implement binary logic functions.

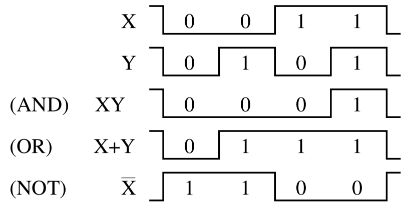


switches in parallel => OR switches in series => AND

- **Logic gate symbols:** physical devices that implement binary logic functions.



(a) Graphic symbols

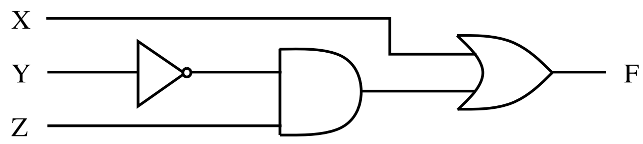


(b) Timing diagram

Logic Gate Diagram

- Boolean equations, truth tables, and logic diagrams express the same function.

Expression: $F = X + \overline{Y}Z$



Logic Circuit Diagram for $F = X + \overline{Y}Z$

Truth table for the function $F = X + \overline{Y}Z$

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1