Computer Mathematics

Fundamental Computer Information for the EE

Number Systems / Mathematical Operations / Examples / Boolean Algebra

by

John A Camara, BS, MS, PE, TF

Course 320

3 PDH (3 Hours)

PO Box 449
Pewaukee, WI 53072
(888) 564 - 9098
support@pdhacademy.com

Nomenclature[1]

ALU	Arithmetic Logic Unit
BCD	Binary Coded Decimal
bit	Smallest Unit of Data a Computer can Process or Store
Boolean	Algebraic Notation representing Logical Propositions
CPU	Central Processing Unit
hex	Hexadecimal (Base-16)
LSB	Lease Significant Bit
LSD	Least Significant Digit
MSB	Most Significant Bit
MSD	Most Significant Digit
octal	Base-8
radix	The Power to which a given Base is Raised

Symbols

b base
N number

Subscripts/Superscripts

* complement
0 initial (zero value)
b base
n number

[1] Not all the nomenclature, symbols, or subscripts may be used in this course—but they are related, and may be found when reviewing the references listed for further information.

TABLE OF CONTENTS

NOMENCLATURE 2
SYMBOLS 2
SUBSCRIPTS/SUPERSCRIPTS 2
LIST OF FIGURES 3
LIST OF TABLES 4
LIST OF EQUATIONS 4
LIST OF EXAMPLES 4
INTRODUCTION 4
POSITIONAL NUMBERING SYSTEMS 5
CONVERTING BASE-B NUMBERS TO BASE-10 5
CONVERTING BASE-10 NUMBERS TO BASE-B 6
BINARY NUMBER SYSTEM 8
OCTAL NUMBER SYSTEM 9
HEXADECIMAL NUMBER SYSTEM 10
CONVERSIONS AMOUNG BINARY, OCTAL, AND HEXADECIMAL NUMBERS 11
COMPLEMENT OF A NUMBER 13
APPLICATION OF COMPLEMENTS TO COMPUTER ARITHMETIC 15
MULTIPLICATION AND DIVISION 17
COMPUTER REPRESENTATION OF NEGATIVE NUMBERS 19
BOOLEAN ALGEBRA FUNDAMENTAL POSTULATES 20
BOOLEAN ALGEBRA LAWS 21
BOOLEAN ALGEGRA THEOREMS 22
CONCLUSION 23
REFERENCES 25

List of Figures

FIGURE 1: SIMPLIFIED COMPUTER ARCHITECTURE 24

FIGURE 2: MICROPROCESSOR ARCHITECTURE 24

List of Tables

TABLE 1: BINARY, OCTAL, AND HEXADECIMAL EQUIVALENTS 10

List of Equations

EQUATION 1: CONVERT TO BASE 2 5
EQUATION 2: BASE-10 FRACTIONS 5
EQUATION 3: THE B'S COMPLEMENT 14
EQUATION 4: THE B−1'S COMPLEMENT 14
EQUATION 5: THE 10'S COMPLEMENT 14
EQUATION 6: THE NINE'S COMPLEMENT 14
EQUATION 7: THE TWO'S COMPLEMENT 14
EQUATION 8: THE ONE'S COMPLEMENT 15
EQUATION 9: COMPLEMENT OF A COMPLEMENT 15
EQUATION 10: SUBTRACTION USING COMPLEMENTS 15

List of Examples

EXAMPLE 1: CONVERT TO BASE 10 6
EXAMPLE 2: CONVERT TO BASE 2 7
EXAMPLE 3: CONVERTING FRACTIONS 7
EXAMPLE 4: ADDING OCTAL NUMBERS 9
EXAMPLE 5: CONVERT HEX TO BASE-10 11
EXAMPLE 6: OCTAL TO BINARY CONVERSION 12
EXAMPLE 7: BASE-2 TO OCTAL CONVERSION 12
EXAMPLE 8: BASE-2 TO BASE-16 13
EXAMPLE 9: APPLICATION OF TEN'S COMPLEMENTS 15
EXAMPLE 10: APPLICATION OF TWO'S COMPLEMENTS 16
EXAMPLE 11: END-AROUND CARRY 17
EXAMPLE 12: BINARY MULTIPLICATION 18
EXAMPLE 13: COMPUTER NEGATIVE NUMBERS 19

INTRODUCTION

Computers are ubiquitous in modern society. An understanding of the manner in which computers "think," or more correctly, process information is both enlightening and requisite knowledge for both computer hardware engineers and software programing engineers. The information contained

within this course comes primarily from the author's text on Computer Engineering [A], which contains additional information to understand the underlying theory and practical applications.

POSITIONAL NUMBERING SYSTEMS

A base-b number, N_b, is made up of individual digits. In a positional numbering system, the position of a digit in the number determines that digit's contribution to the total value of the number. Specifically, the position of the digit determines the power to which the base (also known as the radix), b, is raised. For decimal numbers, the radix is 10, hence the description base-10 numbers.

Equation 1: Convert to Base 2

$$(a_n a_{n-1} \cdots a_2 a_1 a_0)_b = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_2 b^2 + a_1 b + a_0$$

The leftmost digit, an, contributes the greatest to the number's magnitude and is known as the most significant digit (MSD). The rightmost digit, a_0, contributes the least and is known as the least significant digit (LSD).

CONVERTING BASE-b NUMBERS TO BASE-10

Equation 1 converts base-*b* numbers to base-10 numbers. The calculation on the right-hand side of Equation 1 is performed in the base-10 arithmetic and is known as the *expansion method.*[2] See Example 1.
Converting base-b fractions to base-10 is similar to converting whole numbers and is accomplished by Equation 2. See Example 1.

Equation 2: Base-10 Fractions

[2] Equation 1 works with any base number. The *double-dabble* (*double and add*) *method* is a specialized method of converting from base-2 to base-10 numbers in binary-coded decimal (BCD) formation. This is also known as a *shift-and-add-3* algorithm, which can be implemented with a small number of logic gates. [B]

$$(0.a_1a_2\cdots a_m)_b = a_1b^{-1} + a_2b^{-2} + \cdots a_mb^{-m}$$

Some conversion practice follows.

Example 1: Convert to Base 10

What number N is $(1011)_2$ in base-10?

Solution

Using Equation with $b = 2$ gives and noting the most significant digit (MSD) is at $n = 3$ gives the following.

$$N = (a_n a_{n-1}\cdots a_2a_1a_0)_b = a_nb^n + a_{n-1}b^{n-1} + \cdots + a_2b^2 + a_1b + a_0$$

$$= (1)(2)^3 + (0)(2)^2 + (1)(2)^1 + 1$$

$$= 8 + 0 + 2 + 1$$

$$= 11$$

CONVERTING BASE-10 NUMBERS TO BASE-b

The *remainder method* is used to convert base-10 numbers to base-*b* numbers. This method consists of success divisions by the base, *b*, until the quotient is zero. The base-*b* number is found by taking the remainders in the reverse order from which they were found. This method is illustrated in Example 2.

Example 2: Convert to Base 2

What number N is $(75)_{10}$ in base-2?

Solution

Using the remainder method gives the following.

$$\frac{75}{2} = 37 \text{ remainder } 1$$
$$\frac{37}{2} = 18 \text{ remainder } 1$$
$$\frac{18}{2} = 9 \text{ remainder } 0$$
$$\frac{9}{2} = 4 \text{ remainder } 1$$
$$\frac{4}{2} = 2 \text{ remainder } 0$$
$$\frac{2}{2} = 1 \text{ remainder } 0$$
$$\frac{1}{2} = 0 \text{ remainder } 1$$

Now, taking the remainders in reverse order (last-to-first) gives the binary representation for $(75)_{10}$ as $(1001011)_2$.

Converting a base-10 fraction to base-*b* requires multiplication of the base-10 fraction and subsequent fractional parts by the base. The base-*b* fraction is formed from the integer parts of the products taken in the same order in which they were determined. See Example 3.

Example 3: Converting Fractions

What is $(0.14)_{10}$ in base-8 (the octal numbering system)?

Solution

First, multiply the base-10 fractional parts by the base, in this case 8.

$$0.14 \times 8 = 1.12$$

$$0.12 \times 8 = 0.96$$

$$0.96 \times 8 = 7.68$$

$$0.68 \times 8 = 5.44$$

$$0.44 \times 8 = \cdots$$

Now, take the integer parts of the products in the same order in which they were determined, in this case 1075… giving the answer as

$$(0.1075\cdots)_8$$

BINARY NUMBER SYSTEM

There are only two *binary digits (bits)* in the *binary number system*: zero and one.[3] Thus, all binary numbers consist of strings of bits (i.e., zeros and ones). The left most bit is known as the *most significant bit* (*MSB*), and the rightmost bit is the *least significant bit* (*LSB*).

As with digits from other numbering systems, bits can be added, subtracted, multiplied, and divided, although only digits 0 and 1 are allowed in the results. the rules of bit addition are

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0 \text{ carry } 1$$

[3] Alternatively, the binary states may be called *true* and *false*, *on* and *off*, *high* and *low*, or *positive* and *negative*.

OCTAL NUMBER SYSTEM

The *octal* (*base-8*) *system* is one of the alternatives to working with long binary numbers. Only the digits 0 through 7 are used. The rules for addition in the octal system are the same as for the decimal system except that the digits 8 and 9 do not exist. The rules, for example, give the following.

$$7+1=6+2=5+3=(10)_8$$
$$7+2=6+3=5+4=(11)_8$$
$$7+3=6+4=5+5=(12)_8$$

Example 4: Adding Octal Numbers

Add the following base-8 numbers.

(a) $(2)_8+(5)_8$
(b) $(7)_8+(6)_8$

Solution

(a) The sum of 2 and 5 in base-10 is 7, which is less than 8 and; therefore, is a valid number in the octal system. The answer is $(7)_8$.

(b) The sum of 7 and 6 in base-10 is 13, which is greater than 8 (and therefore, needs to be converted). Using the remainder method gives the following.

$$\frac{13}{8} = 1 \text{ remainder } 5$$

$$\frac{1}{8} = 0 \text{ remainder } 1$$

Now, taking the remainders in the reverse order gives a solution of $(15)_8$.

HEXADECIMAL NUMBER SYSTEM

The *hexadecimal* (*base-16*) *system* is a shorthand method of representing the value of four binary digits at a time.[4] Because 16 distinctly different characters are need, the capital letters A through F are used to represent the decimal numbers 10 through 15. The progression of hexadecimal number is illustrated in Table 1.

Table 1: Binary, Octal, and Hexadecimal Equivalents[5]

binary	octal	decimal	hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A

[4] The term *hex* is often heard.

[5] A more complete explanation with examples of usage can be found in Schaum's Outlines. [C] And, also in a very useful text for an excellent overview of computer engineering. [D]

1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10

Example 5: Convert HEX to Base-10

What number N is $(4D3)_{16}$ in base-10?

Solution

The hexadecimal number D is 13 in base-10. Using Equation 1 gives the following.

$$\begin{aligned} N &= (4)(16)^2 + (13)(16)^1 + 3 \\ &= (4)(256) + 208 + 3 \\ &= 1024 + 208 + 3 \\ &= (1235)_{10} \end{aligned}$$

CONVERSIONS AMOUNG BINARY, OCTAL, AND HEXADECIMAL NUMBERS

The octal system is closely related to the binary system because $(2)^3 = 8$. Conversion from a binary to an octal number is accomplished directly by starting at the LSB (right-hand bit) and grouping the bits in threes. Each group of three bits corresponds to an octal digit. Similarly, each digit in an octal number generates three bits in the equivalent binary number.

Conversion from a binary to a hexadecimal number starts by grouping the bits (starting at the LSB) into fours. Each group of four bits corresponds to a hexadecimal digit. Similarly, each digit in a hexadecimal number generates four bits in the equivalent binary number.

Conversion between octal and hexadecimal numbers is easiest when the number is first converted to a binary number.

Example 6: Octal to Binary Conversion

What number is $(5431)_8$ in base 2?

Solution

Group the bits in threes.

$$(5)_8 = (101)_2$$
$$(4)_8 = (100)_2$$
$$(3)_8 = (011)_2$$
$$(1)_8 = (001)_2$$

Group the results from MSB to LSB.

The answer is the following.

$$(101100011001)_2$$

Example 7: Base-2 to Octal Conversion

What number is $(1001011)_2$ in base 8?

Solution

Group the bits in threes starting at the LSB.

1 001 011

Convert the group of threes into octal.

$$(1)_2 = (1)_8$$
$$(001)_2 = (1)_8$$
$$(011)_2 = (3)_8$$

The answer is the following.

$$(113)_8$$

Example 8: Base-2 to Base-16

What is $(1011111101111001)_2$ in the octal numbering system?

Solution

Group the bits into fours starting from the LSB.

1011 1111 0111 1001

Convert these groups into their hexadecimal equivalents.

$$(1011)_2 = (B)_{16}$$
$$(1111)_2 = (F)_{16}$$
$$(0111)_2 = (7)_{16}$$
$$(1001)_2 = (9)_{16}$$

The answer is the following.

$(BF79)_{16}$

COMPLEMENT OF A NUMBER

The *complement* of a given number will be useful in the method for performing computer arithmetic.

The *complement*, *N**, of a number, *N*, depends on the machine (computer, calculator, etc.) being used. Assuming that the machine has a maximum number, *n*, of digits per integer number stored, the *b*'s and ($b-1$)'s complements are as follows.

Equation 3: The b's Complement

$$N_b^* = b^n - N$$

Equation 4: The b−1's Complement

$$N_{b-1}^* = N_b^* - 1$$

For a machine that works in base-10 arithmetic, the *ten's* and *nine's complements* are as follows.

Equation 5: The 10's Complement

$$N_{10}^* = 10^n - N$$

Equation 6: The Nine's Complement

$$N_9^* = N_{10}^* - 1$$

For a machine that works in base-2 arithmetic, the *two's* and *one's complements* are as follows.

Equation 7: The Two's Complement

$$N_2^* = 2^n - N$$

Equation 8: The One's Complement

$$N_1^* = N_2^* - 1$$

APPLICATION OF COMPLEMENTS TO COMPUTER ARITHMETIC

The practical applications of complements to computer arithmetic are represented by Equations 9 and 10.

Equation 9: Complement of a Complement

$$(N^*)^* = N$$

Equation 10: Subtraction Using Complements

$$M - N = M + N^*$$

The binary one's complement is easily found by switching all of the ones and zeros to zeros and ones, respectively. [Equation 9]. It can be combined with a technique known as *end-around carry* to perform subtraction. [Equation 10] End-around carry is the addition of an *overflow bit* to the sum of *N* and it's one's complement.

Example 9: Application of Ten's Complements

Simulate the operation of a base-10 machine with a capacity of four digits per number.

What is the difference of $(18)_{10} - (6)_{10}$ in ten's complements?

Solution

The ten's complement of 6, using Equation 5, is as follows. Recall, the *n* in Equation 5 is the number of available digits, 4 in this case.

$$(6)_{10}^{*} = (10)^4 - 6 = 10{,}000 - 6 = 9994$$

Now, using Equation 10, perform the subtraction.

$$18 - 6 = 18 + (6)_{10}^{*} = 18 + 9994 = 10{,}012$$

However, the machine has a maximum capacity of four digits. Therefore, the leading 1 is dropped leaving 0012 as the answer.

Example 10: Application of Two's Complements

Simulate the operation of base-2 machine with a capacity of five digits per number.

What is the difference of $(01101)_2 - (01010)_2$ in two's complements?

Solution

The two's complement of $(01010)_2$, using Equation 7, is as follows. Recall, the *n* in Equation 7 is the number of available digits, 5 in this case.

$$\begin{aligned} N_2^{*} &= (2)^5 - N = (32)_{10} - N \\ &= (100000)_2 - (01010)_2 \\ &= (32)_{10} - (10)_{10} = (22)_{10} \\ &= (10110)_2 \end{aligned}$$

Now, using Equation 10, perform the subtraction.

$$\begin{aligned} (01101)_2 - (01010)_2 &= (01101)_2 + (10110)_2 \\ &= (100011)_2 \end{aligned}$$

Because the machine has a capacity of only five bits, the leftmost bit is dropped leaving $(00011)_2$ as the answer.

Example 11: End-Around Carry

Solve Example 10 using one's complement and end-around carry.

What is the difference of $(01101)_2 - (01010)_2$ using end-around carry?

Solution

First, the one's complement is found by reversing all the digits, giving the following.

$$(01010)_1^* = (10101)_2$$

Now, add the one's complement to the original number using the rules of bit addition.

$$(01101)_2 + (10101)_2 = (100010)_2$$

The leading bit is an overflow bit, which is removed from the one's complement result giving $(00010)_2$ from the above result. Now, the removed bit is added to give the resulting difference desired.

$$(00010)_2 + (1)_2 = (00011)_2$$

This, as expected, matches the result in Example 10.

MULTIPLICATION AND DIVISION

Integer multiplication is similar to the way one is used to the methods in "normal" arithmetic. The first input (multiplicand) is multiplied by each bit of the second input (multiplier), separately. Then the results are added to find the final solution. Note that the result of multiplication in either the original number or zero. That is,

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Example 12: Binary Multiplication

What is the result of the following multiplication?

$$1101 \times 0101$$

Solution

The result, staying in the binary format, is as follows.

$$\begin{array}{r} 1101 \\ \times\ 0101 \\ \hline 1101 \\ 0000 \\ 1101 \\ 0000 \\ \hline 1000001 \end{array}$$

Checking: In decimal terms, this is $(13)_{10}(6)_{10} = (65)_{10}$. This matches the 1000001.

Division can by accomplished by repeated subtracting the divisor from the dividend and counting the number of times divisor can be subtracted from the dividend before said dividend becomes smaller than the divisor. For example 20 can be divided by 5 by subtracting 5 repeatedly (four times) resulting in a quotient of 4. This method, however, is slow (in terms of computer times) and therefore pre-generated tables are used allowing the system to use a faster *lookup method*.

COMPUTER REPRESENTATION OF NEGATIVE NUMBERS

On paper, a minus sign indicates a negative number. This representation is not possible in a machine. Hence, one of the n digits, usually the MSB, is reserved for sign representation. (This reduces the machine's capacity to represent numbers to n − 1 bits per number.) It is arbitrary whether the sign bit is 1 or 0 for negative numbers as long as the MSB is different for positive and negative numbers.

The one's complement is ideal for forming a negative number because it automatically reverses the MSB. For example, $(00011)_2$ is a five-bit representation of decimal 3. The one's complement is $(11100)_2$, which is recognized as a negative number because the MSB is 1.

Example 13: Computer Negative Numbers

For the various parts of this example, simulate the operation of a six-digit binary machine that uses one's complements for negative numbers.

(a) What is the machine representation of $(-27)_{10}$?

Solution
The number $(27)_{10}$ is $(011011)_2$. The negative of this number is the same as the one's complement. The answer is found by changing all the ones to zeros and vice versa in the original number.

$$(100101)_2$$

(b) What is the decimal equivalent of the two's complement of $(-27)_{10}$?

Solution
Consider Equations 7 and 8. The two's complement is one more than the one's complement. Therefore, the two's complement is

$$(100100)_2 + 1 = (100101)_2$$

Since the 1 is the MSD, this represents $(-26)_{10}$.

(c) What is the decimal equivalent of the one's complement of $(-27)_{10}$?

Solution
From Equation 9, the complement of the complement of a number is the original number. Therefore, the answer is $(27)_{10}$, which is the original number.

BOOLEAN ALGEBRA FUNDAMENTAL POSTULATES[6]

Boolean Algebra is a system of mathematics that deals with the algebraic treatment of logic. When the variables of the system are limited to only one of two values, the system is called two-valued. Switching algebra is the term often used to describe two-valued Boolean algebra. The basic postulates of switching algebra follow.

- Postulate 1: A Boolean variable x has two possible values, 0 and 1. That is,

$$\text{if } x = 0 \text{ then } x \neq 1, \text{ and}$$
$$\text{if } x = 1 \text{ then } x \neq 0$$

- Postulate 2: The NOT operation, $\bar{x}$ or x', is defined as

$$\bar{0} = 1 \quad \bar{1} = 0$$

- Postulate 3: The logical operations AND ($\wedge$ or $\cdot$), and OR ($\vee$ or $+$) are defined as

[6] Boolean gets its name from the English mathematician, George Boole. Boole created a new branch of algebra, now known as Boolean Algebra, where the value of true is 1 and the value of false is 0. In Boolean Algebra, there are three main logical operations: AND, OR, and NOT.

$$0 \cdot 0 = 0 \quad 0 + 0 = 0$$
$$0 \cdot 1 = 0 \quad 0 + 1 = 1$$
$$1 \cdot 0 = 0 \quad 1 + 0 = 1$$
$$1 \cdot 1 = 1 \quad 1 + 1 = 1$$

Note: Sometimes the symbol for the AND operation is dropped completely and $x_1 \cdot x_2$ is represented as $x_1 x_2$.

BOOLEAN ALGEBRA LAWS

The process of reducing logical expressions to simpler forms is aided by the use of the following relationships, or laws of switching algebra.[7] In the following, *x* can represent a single variable or a general logical function.

- Special Properties of 0 and 1

$$0 + x = x \quad 0 \cdot x = 0$$
$$1 + x = 1 \quad 1 \cdot x = x$$

- Idempotence Laws

$$x + x = x \quad x \cdot x = x$$

- Complementation Laws

$$x + \bar{x} = 1 \quad x \cdot \bar{x} = 0$$

- Involution

$$\overline{(\bar{x})} = x$$

- Commutative Laws

[7] The reduction of such expressions is desired to minimize circuit complexity and costs when realizing a given expression with electronic logic circuits.

$$x + y = y + x \qquad x \cdot y = y \cdot x$$

- Associative Laws

$$x + (y + z) = (x + y) + z$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

- Distributive Laws

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

- Absorption Laws

$$x + (x \cdot y) = x \qquad x \cdot (x + y) = x$$
$$x + (\bar{x} \cdot y) = x + y \qquad x \cdot (\bar{x} + y) = x \cdot y$$

BOOLEAN ALGEGRA THEOREMS

The simplification of logical expression is further enhanced by a set of relationships that exist as theorems, although one takes the name of a law. These theorems follow.

- Simplification Theorems

$$xy + x\bar{y} = x \qquad (x + y)(x + \bar{y}) = x$$
$$x + xy = x \qquad x(x + y) = x$$
$$(x + \bar{y}) = xy \qquad x\bar{y} + y = x + y$$

Note: The AND symbol is often assumed between parenthesis.

- DeMorgan's Laws

$$\overline{(x+y+z+\cdots)} = \bar{x}\,\bar{y}\,\bar{z}\cdots$$
$$\overline{(xyz\cdots)} = \bar{x}+\bar{y}+\bar{z}\cdots$$
$$\overline{[f(x_1,x_2,x_3,\cdots,x_n,0,1,+,\cdot)]} = f(\bar{x}_1,\bar{x}_2,\bar{x}_3,\cdots,\bar{x}_n,1,0,\cdot,+)$$

- Duality

$$(x+y+z+\cdots)^{\mathrm{D}} = xyz\cdots$$
$$(xyz\cdots)^{\mathrm{D}} = x+y+z+\cdots$$
$$[f(x_1,x_2,x_3,\cdots,x_n,0,1,+,\cdot)]^{\mathrm{D}} = f(x_1,x_2,x_3,\cdots,x_n,1,0,\cdot,+)$$

- Consensus Theorem

$$xy+yz+\bar{x}z = xy+\bar{x}z$$
$$(x+y)(y+z)(\bar{x}+z) = (x+y)(\bar{x}+z)$$
$$(x+y)(\bar{x}+z) = xz+\bar{x}y$$

CONCLUSION

A simplified view of computer architecture is shown in Figure 1. In general, the arithmetic operations referred to in this course take place in the Central Processing Unit (CPU). Inside the CPU the arithmetic operations take place in the Arithmetic Logic Unit (ALU) shown in Figure 2. Both of these units are Hardware. The main memory contains all or part of the operating system used to control the computer. The control is accomplished via Software. The basic concepts surrounding both can be studied in Reference [D]. More in-depth information on the many aspects of computer science and engineering can be found in Reference [E].

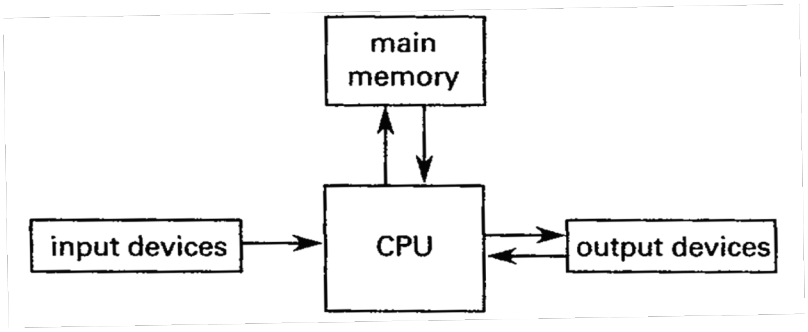

Figure 1: Simplified Computer Architecture

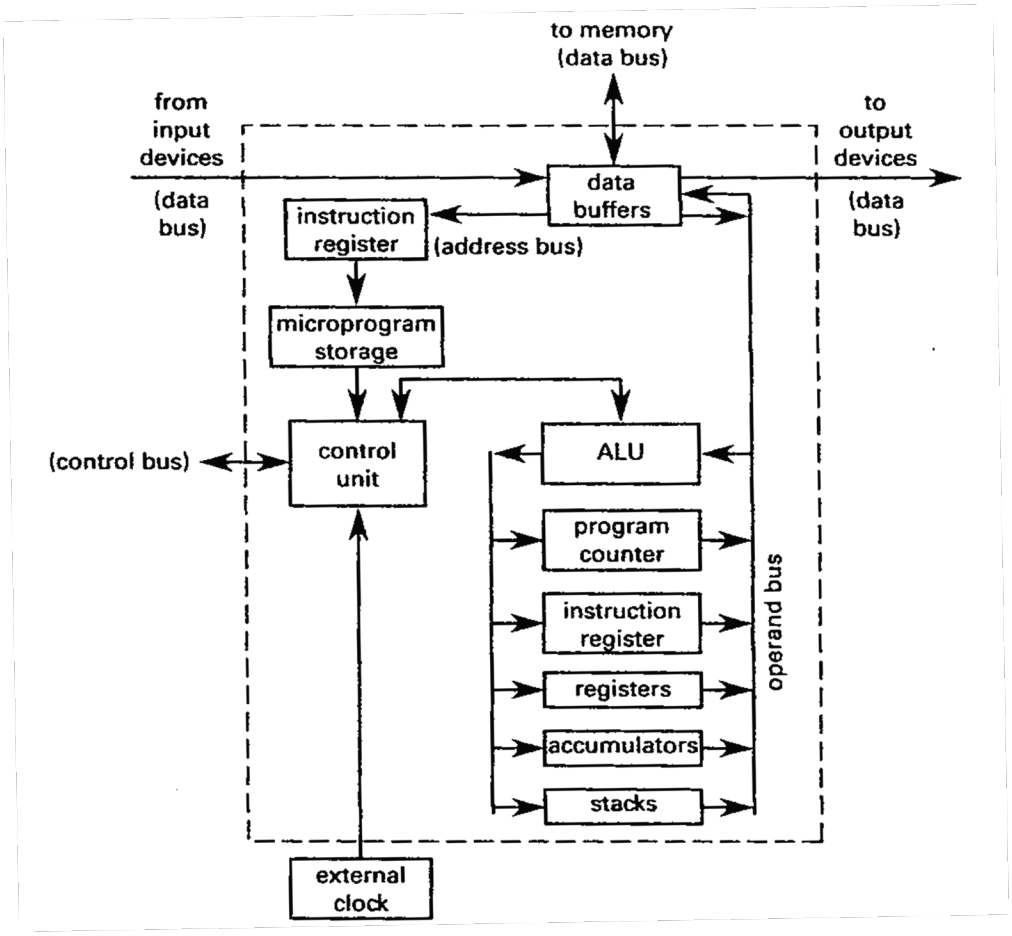

Figure 2: Microprocessor Architecture

REFERENCES

A. Camara, John A. "*Computer Engineering Reference Manual for the Electrical and Computer PE Exam*." Belmont, CA: PPI (Kaplan), 2010.
B. Wikipedia, The Free Encyclopedia (2023). "Double dabble." https://en.wikipedia.org/wiki/Double_dabble#:~:text=In%20computer%20science%2C%20the%20double,the%20expense%20of%20high%20latency (accessed 21 SEP 2023).
C. Carter, Nicholas, PhD. "Computer Architecture—Chap. 2." New York: McGraw-Hill, 2002.
D. Booth, Taylor L. "*Introduction to Computer Engineering, Hardware and Software Design—Chapters 4 & 5*." 3rd ed. New York: Wiley, 1984.
E. Alty, James L, et al. "*Computer Science and Engineering Handbook.* Edited by Allen B. Tucker, Jr. Boca Raton, FL: CRC Press, Inc, 1997.

