**5. The Role of Binary Numbers in Computer Architecture**

Binary Number system is a base 2 which uses only two digits 0 and 1. In this system we represent sequence of these two digits as (1001)2.

Computers use the binary system because it directly corresponds to the two states of electronic circuits:

* 0 represents low voltage (Off state)
* 1 represents high voltage (On state)

In digital circuits, these states od 0s and 1s are managed using transistors, where:

* 1 indicates the presence of an electrical current (**high state**).
* 0 indicates the absence of an electrical current (**low state**).

<https://www.geeksforgeeks.org/binary-number-system/#binary-number-system>

<https://www.techtarget.com/whatis/definition/binary>

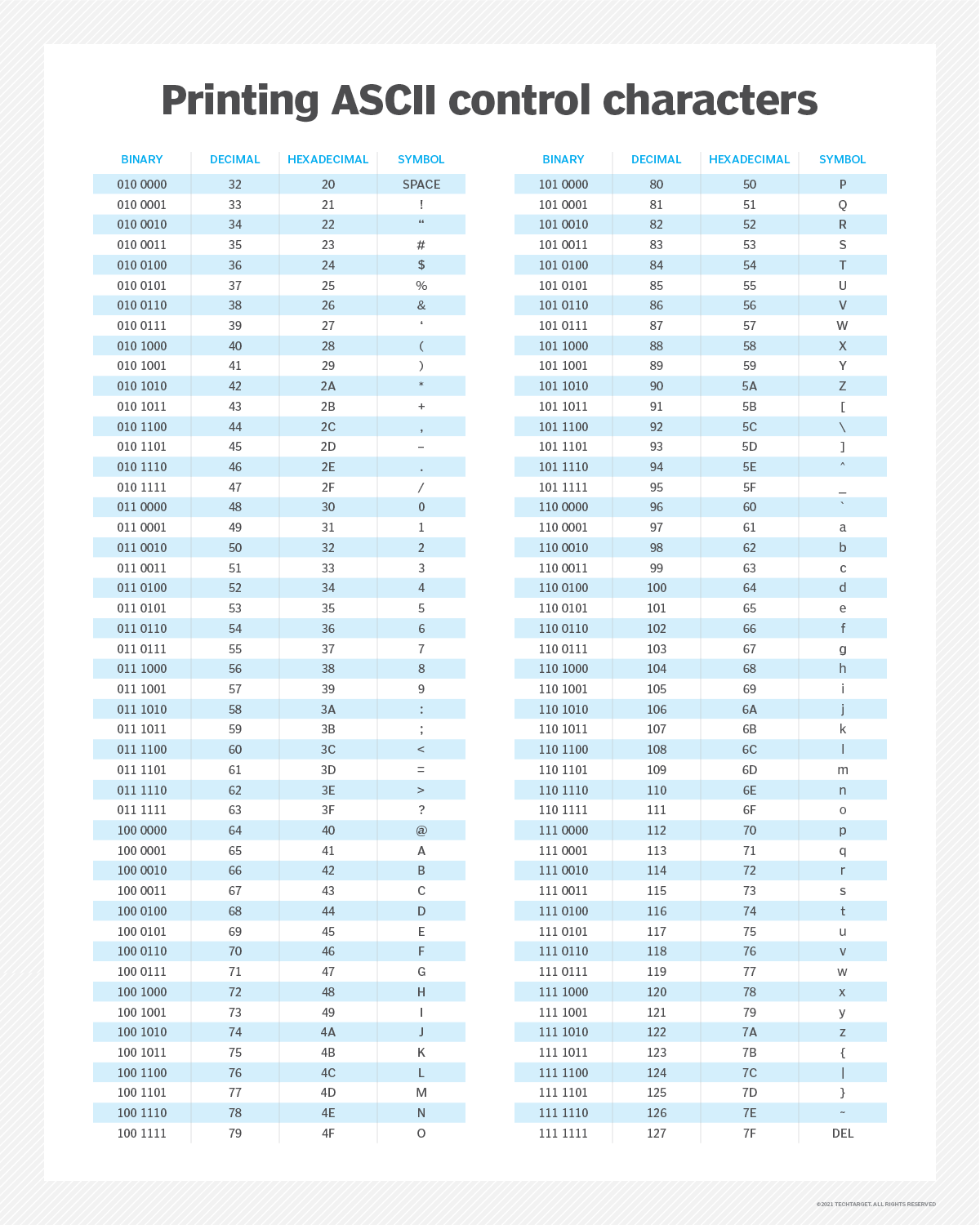
In computing, data is represented in units of 8 bits, also known as a byte. A byte can store values that range from 0 to 255 (in decimal), allowing it to represent characters, numbers, and symbols in various encoding standards, such as ASCII (American Standard Code for Information Interchange) like:

|  |  |
| --- | --- |
| Binary | ASCII(characters) |
| 1000001 | A |
| 1111010 | z |

<https://www.bbc.co.uk/bitesize/guides/zfspfcw/revision/7>

This binary encoding enables computers to store and process text, numerical values, and symbols efficiently.

Thus, in computer architecture, the binary number system is fundamental for representing characters, performing data operations, and encoding information across different computational processes.

Image source: <https://www.techtarget.com/whatis/definition/binary>Image link: <https://www.techtarget.com/rms/onlineimages/printing_ascii_control_characters-f.png>

**The Role of binary in computer architectures**

Binary can be used in representing and storing data, performing calculations, retrieving data and controlling and organizing tasks. Since there is only two state 0 as low voltage “off” and 1 as high voltage “on” state, we can use this feature of binary to perform calculations using components like transiters and circuits.

We can also represent signed numbers using representations like sign and magnitude representation, 1’s compliment and 2’s compliment. Of all, Two’s complement is commonly favored in contemporary systems due to its advantages over other methods. It simplifies arithmetic operations and eliminates the issue of having two representations for zero.

<https://www.geeksforgeeks.org/binary-representations-in-digital-logic/>

<https://eepower.com/technical-articles/digital-electronics-basics-digital-binary-operation/>

**Binary bit Operations:**

* **Addition:**
  + 1 + 1 = 0, 1 carried over
  + 1 + 0 = 1,
  + 0 + 0 = 0

Example:

1101 (13)

+ 1011 (11)

------------------

1|1000 (24)

* **Ones column:** 1 + 1 = 10 (write down 0, carry-over 1)
* **Twos column:** 0 + 1 + 1 = 10 (write down 0, carry-over 1)
* **Fours column:** 1 + 0 + 1 = 10 (write down 0, carry-over 1)
* **Eights column:** 1 + 1 + 1 = 11 (write down 1, carry-over 1)
* **Sixteenths column:** 1 (write down 1), exceeds the current bit size.

Addition can be performed in computing with the help of half adders and full adders.

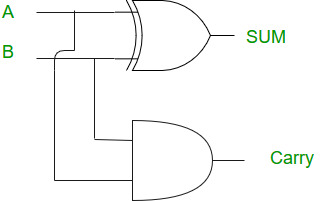
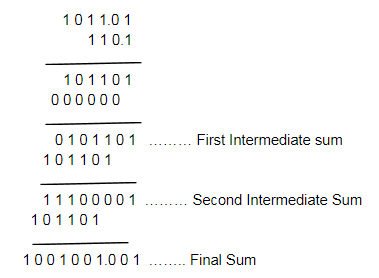


Image Source: <https://www.geeksforgeeks.org/half-adder-in-digital-logic/>

Image link: <https://media.geeksforgeeks.org/wp-content/cdn-uploads/Half_Adder.jpg>

* **Multiplication:**
  + 1 x 1 = 1,
  + 1 x 0 = 0,
  + 0 x 0 = 0



Admin. (2020, July 29). *Binary Multiplication (Rules and Solved Examples)*. BYJUS. <https://byjus.com/maths/binary-multiplication/>

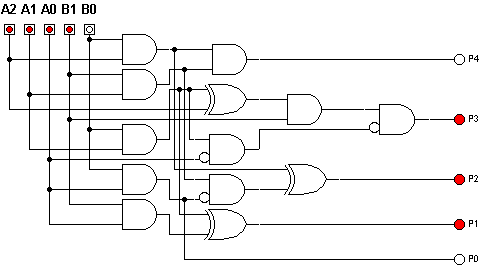


Image source:

Cluster-based evolutionary design of digital circuits using all improved multi-expression programming - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/Evolved-3x2-bit-multiplier-13-gates-with-4-levels-using-and-and-with-one-input_fig1_220742040> [accessed 14 Mar 2025]

* **Subtraction**

1. Find the two’s complement of the subtrahend

* **Invert all bits** (one’s complement).
* **Add 1** to the inverted bits

1. Add the result to the minuend
2. If there is a carry-out, discard it.
3. If no carry-out occurs, the result is negative (already in two’s complement form)

Example: 7 - 5 in Binary in following steps:

1. Convert Numbers to Binary

7 → 0111

5 → 0101

1. Find Two’s Complement of 5

Invert all bits of 5: 0101 → 1010

Add 1: 1010 + 1 = 1011

1. Add to 7

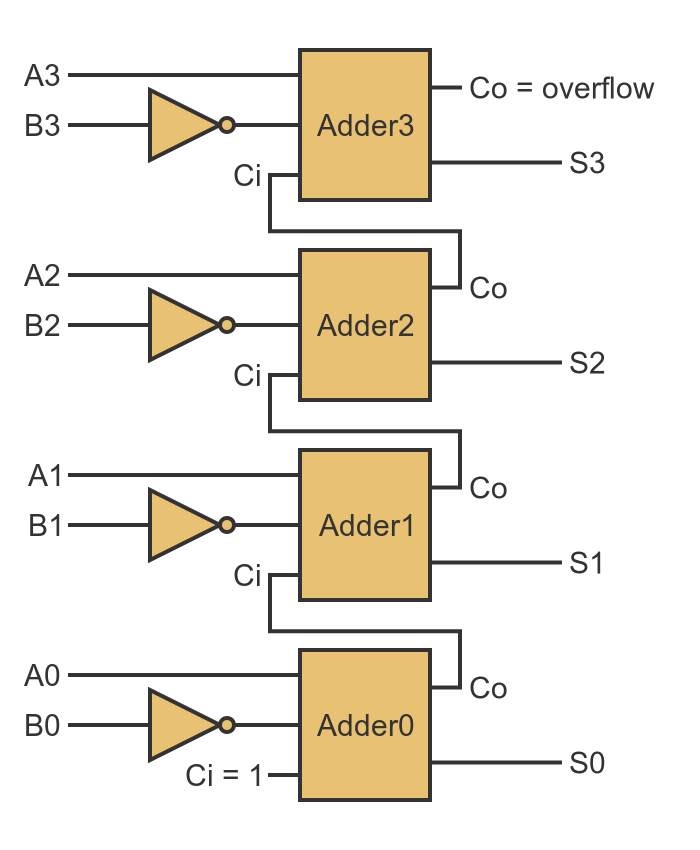
0111 (7)

+ 1011 (-5 in two’s complement)

------------

1|0010 (Ignore carry-out → Final answer: 0010)

In digital logic, subtraction is possible through addition.



McBride, M. (n.d.). *GraphicMaths - Creating a subtractor with logic gates*. <https://graphicmaths.com/computer-science/logic/subtractor/>

* **Division**

There are various ways of achieving division in digital logics, like:

1. Restoring Division Algorithm:

* Involves repeated subtraction of the divisor from the dividend.
* Restores the previous value if the result is negative.
* Utilizes subtractors and control logic in the Arithmetic Logic Unit (ALU).

1. Non-Restoring Division Algorithm:

* Enhances efficiency by eliminating the restore step.
* Adjusts the quotient based on the sign of the remainder.
* Incorporates adders and control units for sign evaluations.

1. Newton–Raphson Division:

* An iterative method that approximates the reciprocal of the divisor.
* Multiplies the reciprocal by the dividend to obtain the quotient.
* Requires multipliers and adders to perform iterative calculations.

1. Goldschmidt Division:

* Transforms division into a multiplication problem.
* Simultaneously scales the dividend and divisor to converge the divisor to 1.
* Uses parallel multipliers to enhance computation speed.

Example, Divide 13 by 3 in Binary

Dividend (13₁₀) = 1101₂

Divisor (3₁₀) = 11₂

Using **restoring division** algorithm in the following steps:

1. **Initialize:**

Dividend: 1101₂

Divisor: 11₂

Quotient: 0000 (initially zero)

Remainder: 0000 (initially zero)

1. **Bring down the first bit of the dividend.**

Remainder: 110 (first three bits of dividend)

Perform subtraction: 110 - 11 = 01

Quotient: 1 (since subtraction was successful, we write 1)

1. **Bring down the next bit of the dividend.**

Remainder: 010 (bring down next bit, which is 0)

Perform subtraction: 010 - 11 (this goes negative)

Restore the remainder: Return to 110 (previous remainder)

Quotient: 10 (since subtraction went negative, we write 0)

1. **Bring down the final bit of the dividend.**

Remainder: 110 (bring down the final bit, which is 1)

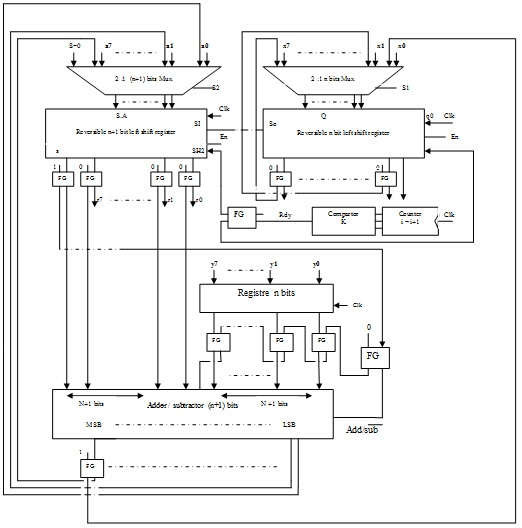
Perform subtraction: 110 - 11 = 01

Quotient: 101 (successful subtraction, so write 1)

1. **Final Result:**

Quotient = 101₂ (which is 5 in decimal)

Remainder = 1₂ (which is 1 in decimal)



Division Circuit Using Reversible Logic Gates - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/Proposed-reversible-8-bit-divider-circuit_fig5_324151648> [accessed 14 Mar 2025]

**Binary Arithmetic Implemented in Arithmetic Logic Unit (ALU)**

**ALU** is responsible for performing binary arithmetic and logical operation inside of a CPU. It can process addition, subtraction, multiplication, and division using logic gates, adders, and control circuits.

**Major Components of the ALU for Binary Arithmetic:**

1. Half Adder & Full Adder (For Addition & Subtraction)

* The Half Adder performs single-bit binary addition using XOR and AND gates.
* The Full Adder extends this concept to multi-bit addition by handling carry-over bits.
* Subtraction is done using Two’s Complement, so subtraction circuits often use adders with inverters instead of separate subtractors.

1. Two’s Complement Circuit (For Subtraction)

* The ALU converts subtraction into addition by taking the two’s complement of the subtrahend and adding it to the minuend.
* Uses NOT gates (for bit inversion) and increment circuits (to add 1).

1. Shift-and-Add Multiplication (For Multiplication)

* Multiplication is performed using bitwise shifting and addition.
* The ALU checks each bit of the multiplier:
  + If 1, it adds the multiplicand.
  + If 0, it only shifts left (like long multiplication).
* High-performance processors use Booth’s Algorithm to optimize this process.

1. Restoring & Non-Restoring Division (For Division)

* The ALU uses subtraction-based division algorithms like:
* Restoring Division (restores remainder if negative).
* Non-Restoring Division (avoids unnecessary restores for efficiency).
* More advanced processors use Newton–Raphson and Goldschmidt algorithms for fast division.

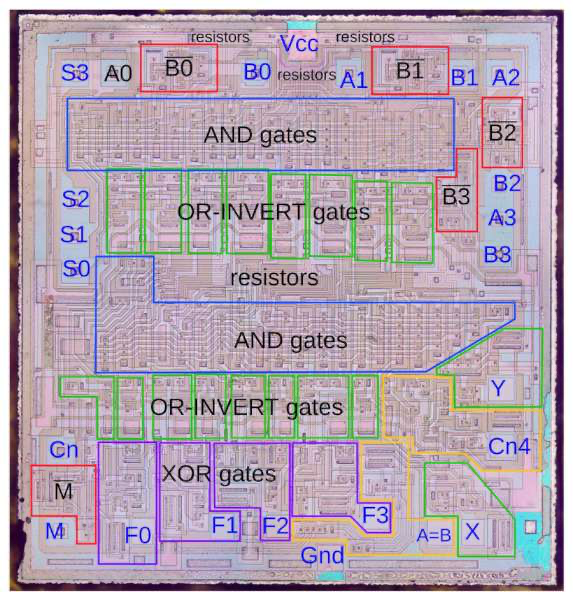
1. Logic Gates (For Boolean Operations)

* The ALU also performs logic operations like AND, OR, XOR, and NOT to manipulate binary values.
* These operations are used for bitwise operations, comparisons, and decision-making in programs.

**How ALU Executes Binary Arithmetic Step-by-Step**

1. **Addition**: Uses full adders to sum binary numbers, handling carry propagation.
2. **Subtraction**: Uses two’s complement to convert subtraction into addition.
3. **Multiplication**: Uses shift-and-add method, optimized with Booth’s Algorithm.
4. **Division**: Uses repeated subtraction or advanced division algorithms.
5. **Logic Operations**: Uses AND, OR, XOR, and NOT gates for bitwise computations.

Rao, R. (2024, November 23). Building a computer from scratch: Understanding Arithmetic Logic Unit #2. Medium. <https://medium.com/%40ruthurao/building-a-computer-from-scratch-understanding-arithmetic-logic-unit-2-315ce860c972>



Mixos. (2017, January 13). *Inside the 74181 ALU chip: die photos and reverse engineering*. Electronics-Lab.com. <https://www.electronics-lab.com/inside-74181-alu-chip-die-photos-reverse-engineering/>