International Journal of Innovations in Research
ISSN: 3048-9369 (Online)

A Tutorial Analysis of 16-Bit Addition, Alu Architecture, and PSW Behavior in the Intel 8085 Microprocessor

Dr. Giriraj Kumar Prajapati[1]; Prof. Dr. Ashok Kumar Gupta[2]
[1]Professor, Alpine Institute of Technology, Ujjain (M.P.) India
[2]Professor, Satpuda College of Engineering & Polytechnic Balaghat (M.P.) India
Email ids: prajapatigiri38@gmail.com; profdrashokkumargupta@gmail.com

Abstract
The Intel 8085 microprocessor is widely used in engineering education due to its simple accumulator-centric architecture. Although it is fundamentally an 8-bit processor, it supports limited 16-bit arithmetic operations primarily intended for address manipulation. This dual nature often creates conceptual confusion among students, particularly regarding Arithmetic Logic Unit (ALU) operation, condition flag behaviour, and Program Status Word (PSW) updates during 16-bit addition. A common misconception is that all arithmetic operations must involve the accumulator, raising questions about how 16-bit addition is possible without accumulator participation. This tutorial paper provides a clear architectural explanation of 8-bit and 16-bit arithmetic paths in the 8085, emphasizing the role of a separate internal 16-bit adder used by the DAD instruction. The paper further analyses flag and PSW behaviour during such operations and highlights common student misconceptions. Designed as a pedagogical tutorial, this work aims to enhance conceptual clarity in microprocessor education and serve as a reference for educators and learners.
Keywords: *Intel 8085, Program Status Word, ALU architecture, DAD instruction, 16-bit addition, microprocessor education.*

I. INTRODUCTION

Arithmetic operations are the foundation of all digital computing systems, and addition is the most performed operation in processors. Many microprocessors work with limited word lengths; therefore, multi-byte arithmetic is required to process larger data values. The Intel 8085 microprocessor, although an 8-bit processor, supports higher-order arithmetic through software-based techniques such as carry propagation and register pairing [1], [2].

In practical applications, operations on 16-bit data are frequently needed for address calculation, data processing, and control applications. The 8085 microprocessor achieves 16-bit addition by combining multiple 8-bit operations using its Arithmetic Logic Unit and status flags [3], [5]. This approach helps users understand how low-level arithmetic is implemented in real hardware systems [6].

The study of multi-byte arithmetic is essential for understanding processor organization and instruction execution. Concepts such as carry generation, flag utilization, and memory interfacing are fundamental to computer architecture and embedded system design [7], [9]. These

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

concepts remain relevant in modern computing, as they form the basis of arithmetic processing in contemporary digital systems [4], [10].
Moreover, binary arithmetic operations are closely related to the design of digital circuits and ALUs. Digital design principles emphasize the importance of such operations in both hardware and software implementations [8]. Therefore, implementing 16-bit addition using an 8-bit microprocessor provides valuable insight into the working of digital systems.

This paper presents a clear and systematic method for performing 16-bit addition using the 8085 microprocessors, focusing on program logic, instruction usage, and carry handling, as supported by existing literature [1]–[10].

The Intel 8085 microprocessor continues to be a foundational component in undergraduate curricula for computer organization and microprocessor architecture. Its accumulator-centric design offers a clear and accessible model for understanding early processor architectures. However, despite being an 8-bit microprocessor, the 8085 supports limited 16-bit arithmetic operations, particularly for address and pointer calculations.

This architectural characteristic often leads to confusion among students. A frequently raised question is whether arithmetic operations are possible without the accumulator, given that most instructions mandate the accumulator as an operand. The confusion becomes more pronounced when studying the DAD instruction, which performs 16-bit addition without explicitly involving the accumulator or updating most condition flags.

This paper addresses these conceptual challenges by presenting a tutorial-oriented explanation of the internal arithmetic paths of the 8085, with particular emphasis on ALU usage, 16-bit addition, and PSW behaviour.

II. ACCUMULATOR-CENTRIC ARITHMETIC IN 8085

A. 8-Bit ALU Operations

The 8085 contains an 8-bit Arithmetic Logic Unit designed for data arithmetic and logical operations. Instructions such as ADD, SUB, ANA, ORA, and XRA operate exclusively through the accumulator. In these operations:

- One operand is always the accumulator
- The second operand is a register, memory location, or immediate data
- The result is stored back in the accumulator
- All condition flags are updated

This design establishes the widely taught rule that **the accumulator is mandatory for ALU-based arithmetic operations**.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

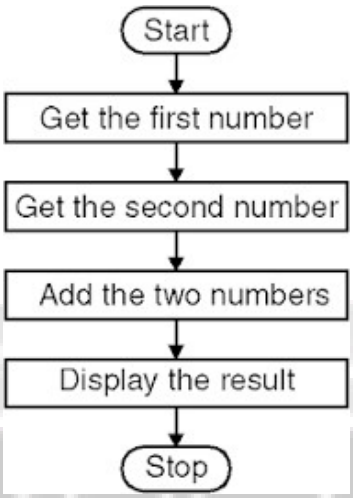


Fig. 1: Flow chart

B. Limitation of the 8-Bit ALU

The accumulator-centric ALU is optimized for data processing but is inefficient for frequent address calculations. Address manipulation involves 16-bit quantities, and performing such operations using multiple 8-bit instructions would increase execution time and program complexity.

III. NEED FOR 16-BIT ADDITION

Address arithmetic is fundamental to microprocessor operation. Tasks such as stack manipulation, array traversal, and pointer updates inherently require 16-bit calculations. To efficiently support these operations, the 8085 provides limited 16-bit arithmetic capability without expanding the complexity of its main ALU.

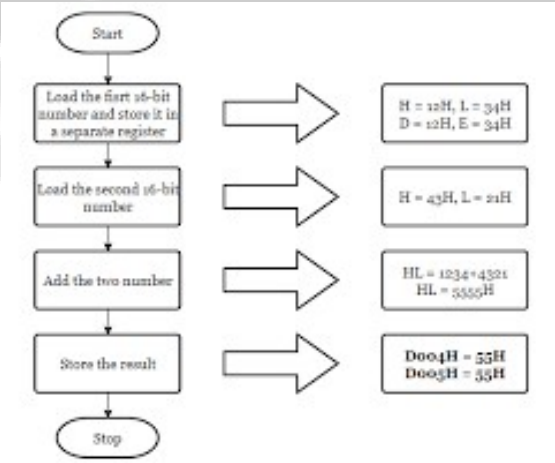


Fig. 2: 16 bits addition process

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

IV. 16-BIT ADDITION USING DAD INSTRUCTION

The instruction used for 16-bit addition in the 8085 is:

HL ←HL + rp

where rp represents a register pair (BC, DE, HL, or SP).
Key characteristics of the DAD instruction are:

- The accumulator is not involved
- The result is stored in the HL register pair
- The instruction is intended for address arithmetic

LHLD 2050H
XCHG
LHLD
2052H
DAD D
SHLD
2060H HLT
Input: 2050H: 1234H
2052H: 1312H
Output

V. RESOLUTION OF A COMMON ARCHITECTURAL MISCONCEPTION

A. The Core Doubt

A common student doubt is:
If ALU operations require the accumulator, how can 16-bit addition be performed without it?

B. Architectural Explanation

The resolution lies in recognizing that **DAD does not use the 8-bit accumulator-based ALU**. Instead, the 8085 incorporates a **separate internal 16-bit adder**, often referred to as an address adder. This adder is used exclusively for:

- DAD
- INX and DCX
- Stack pointer updates
- Program counter increments

During these operations, the accumulator and the main ALU are completely bypassed.

C. Distinct Arithmetic Paths

8-bit data arithmetic path:
Register/Memory → Accumulator → 8-bit ALU → Accumulator
16-bit address arithmetic path:
HL Register Pair → 16-bit Internal Adder ← Register Pair / SP
This distinction explains why accumulator involvement is mandatory for 8-bit arithmetic but unnecessary for 16-bit address arithmetic.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

VI. FLAG AND PSW BEHAVIOR DURING 16-BIT ADDITION

A. Flag Register Overview

The 8085-flag register consists of the Sign, Zero, Auxiliary Carry, Parity, and Carry flags. These flags are defined to reflect the outcome of accumulator-based operations.

B. Effect of DAD on Flags

Since the accumulator is not involved and the 8-bit ALU is bypassed:

- Sign, Zero, Auxiliary Carry, and Parity flags remain unchanged
- Only the Carry flag is updated to indicate 16-bit overflow

C. Program Status Word (PSW)

The Program Status Word is a 16-bit register pair:

[
PSW = Accumulator + Flag
]

During execution of the DAD instruction:

- The accumulator remains unchanged
- Only the Carry bit in the flag register may change

Thus, PSW is only partially affected.

VII. ILLUSTRATIVE EXAMPLES

Example 1: No Carry

HL = 1234H, DE = 1111H
DAD D → HL = 2345H
Carry flag remains reset, PSW unchanged.

Example 2: With Carry

HL = FFFFH, DE = 0001H
DAD D → HL = 0000H
Carry flag is set; other flags remain unchanged.

A frequent misconception is that the Zero flag should be set when HL becomes zero. This is incorrect because the Zero flag monitors the accumulator, not register pairs.

VIII. EDUCATIONAL SIGNIFICANCE AND LEARNING OUTCOMES

A. Educational Significance

Classroom experience indicates that students often overgeneralize the accumulator requirement for all arithmetic operations. This paper directly addresses this misconception by exposing the existence of separate arithmetic paths within the 8085.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

Output:

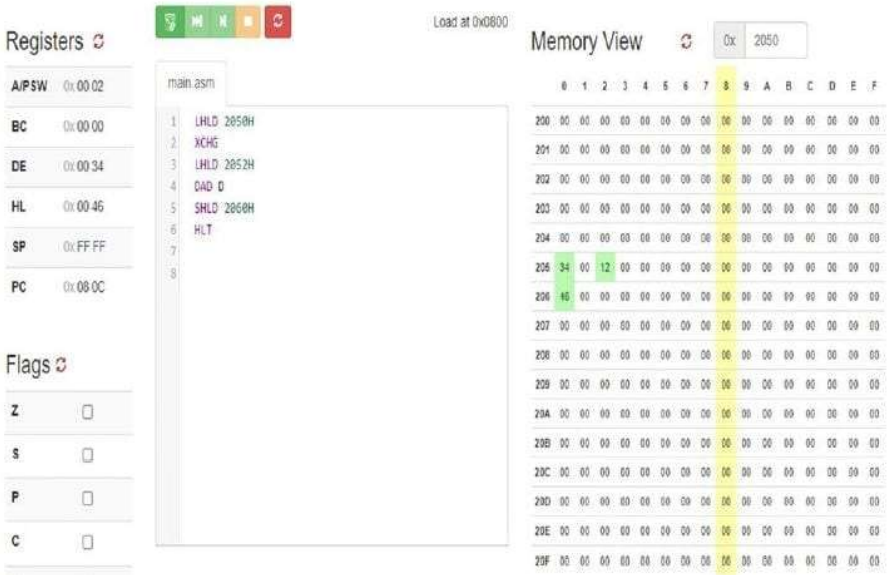


Fig.3 Result in Sim Simulator 8085

B. Learning Outcomes

After studying this tutorial, learners should be able to:

- Differentiate between data arithmetic and address arithmetic
- Explain why DAD bypasses the accumulator
- Predict correct flag and PSW behavior during 16-bit addition
- Avoid common examination and programming errors

IX. CONCLUSION

The apparent contradiction between accumulator-dependent ALU operation and accumulator-independent 16-bit addition in the Intel 8085 is resolved by recognizing the presence of a dedicated internal 16-bit address adder. The DAD instruction leverages this adder to perform efficient address arithmetic without involving the accumulator, thereby affecting only the Carry flag and partially updating the PSW. Understanding this architectural design choice enhances conceptual clarity, improves examination performance, and strengthens foundational knowledge in microprocessor education.

FUNDING DECLARATION

The authors declare that no financial support, funding, grant, or sponsorship was received from any public, private, governmental, or non-governmental agency for the conduct of this research. The study was carried out independently using the institutional resources available to the authors.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)

No external funding influenced the design of the study, data collection, analysis, interpretation of results, or the preparation of the manuscript.

REFERENCES

[1] R. S. Gaonkar, *Microprocessor Architecture, Programming, and Applications with the 8085*, 6th ed. New Delhi, India: Penram International Publishing, 2013.
[2] A. K. Ray and K. M. Bhurchandi, *Advanced Microprocessors and Peripherals*, 3rd ed. New Delhi, India: McGraw-Hill Education, 2011.
[3] B. Ram, *Fundamentals of Microprocessors and Microcomputers*, 8th ed. New Delhi, India: Dhanpat Rai Publications, 2016.
[4] Douglas V. Hall, *Microprocessors and Interfacing: Programming and Hardware*, 2nd ed. New York, NY, USA: McGraw-Hill, 1992.
[5] Intel Corporation, *8085 Microprocessor User's Manual*, Santa Clara, CA, USA, 1980.
[6] T. R. Padmanabhan, *Programming and Interfacing the 8085 Microprocessor*, New Delhi, India: BPB Publications, 2007.
[7] William Stallings, *Computer Organization and Architecture: Designing for Performance*, 10th ed. Boston, MA, USA: Pearson, 2016.
[8] M. Morris Mano and M. D. Ciletti, *Digital Design: With an Introduction to the Verilog HDL*, 5th ed. Upper Saddle River, NJ, USA: Pearson, 2013.
[9] John P. Hayes, *Computer Architecture and Organization*, 3rd ed. New York, NY, USA: McGraw-Hill, 1998.
[10] S. Salivahanan, *Digital Circuits and Design*, New Delhi, India: Oxford University Press, 2012.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)