|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FAKULTI TEKNOLOGI KEJURUTERAAN**  **ELEKTRIK DAN ELEKTRONIK**  **UNIVERSITI TEKNIKAL MALAYSIA MELAKA** | | | | | |
| **COMPUTER ORGANIZATION AND ARCHITECHTURE** | | | | | |
| BEEC 2373 | | | SEMESTER 2 | SESI 2019/2020 | |
| LAB 5: PROCEDURES, STACKS AND ARRAYS | | | | | |
| **NO.** | **STUDENTS' NAME** | | | | **MATRIC. NO.** |
| **1.** | **Ahmad Irfan Bin Harman** | | | | **B081910068** |
| **2.** |  | | | |  |
| **3.** |  | | | |  |
| **PROGRAMME** | | **BEEC** | | | |
| **SECTION / GROUP** | | **1/1** | | | |
| **DATE** | |  | | | |
| **NAME OF INSTRUCTOR(S)** | |  | | | |
|  | | | |
| **EXAMINER’S COMMENT(S)** | | | | **TOTAL MARKS** | |

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. No. | Date | Author(s) | Description |
| 1.0 | 12 FEB 2020 | 1. Noor Mohd Ariff 2. Ahmad Nizamudin | 1. Update to new UTeM logo 2. Update faculty's name 3. Change "course" to "programme" 4. Remove verification stamp |
|  |  |  |  |
|  |  |  |  |

## OBJECTIVES

* To construct and test programs using the following categories of 8086 Instruction Set:

• Data Movement

• Arithmetic Operations

* To understand 8086 microprocessor architecture.

## EQUIPMENT/COMPONENTS

1. Personal computer installed with 8086 Software.

## SYNOPSIS & THEORY

The table below summarizes the different categories of 8086 Instruction Set. In this experiment you will be creating and testing codes written in assembly language programs using 8086 Instruction Set. Observe how the 8086 internal registers are affected by the different instructions.

**TABLE 1: Categories of 8086 Instruction**

**Set**

**Type Description Operation Name**

**Data Transfer** Transfer data from one location to another

Move, Store, Load, Exchange, Clear, Set, Push, Pop

**Arithmetic** Perform arithmetic function in ALU

Add, Subtract, Multiply, Divide, Absolute, Negate, Increment, Decrement

**Logical** Perform logic function in ALU AND, OR, NOT, XOR, Test, Compare, Shift, Rotate

**Transfer of Control** Update program counter Jump, Jump Conditional, Jump to Subroutine, Return, Skip, Skip Conditional, Halt, Wait,No opertion

**Input/Output** Issue command to I/O

module

Input, Output, Start I/O, Test I/O

**Conversion** May involve special logic Translate, Convert

to perform conversion

1. **PROCEDURE**

## Introduction to stack

1. Run the emulator8086.
2. Perform this operation in single step mode and write the values of registers for every step.

CODE SEGMENT

ASSUME CS:CODE, DS:CODE

MOV AX, 42AH

MOV BX, 2E5H

MOV CX, 2H

PUSH AX

PUSH BX

LEV:ADD AX,BX

LOOP LEV

MOV BX, AX

POP AX

HLT

CODE ENDS

END

1. Go to **view > stack** to help you understand how Stack and PUSH & POP instructions work. Explain in Discussion.

## Introduction to Arrays

1. Array of A, B, C is declared as below codes. Run this code and find out the registers output.

ORG 100h

A DB 0Ah , 1Ah , 2Ah , 3Ah , 4Ah , 5Ah

B DB 0Bh , 1Bh , 2Bh , 3Bh

C DB 0Ch , 1Ch

MOV AL, A+4

MOV AL, [B+3]

MOV AL, [B+4]

MOV AL, [A+11]

MOV AL, B+2

XCHG A+3, AL

MOV B+2, AL

HLT

RET

1. Open the memory and variables window. Identify where the array values stored in the memory. Observe how the value in variables/memory changes when you execute the XCHG operation. Explain in Discussion.

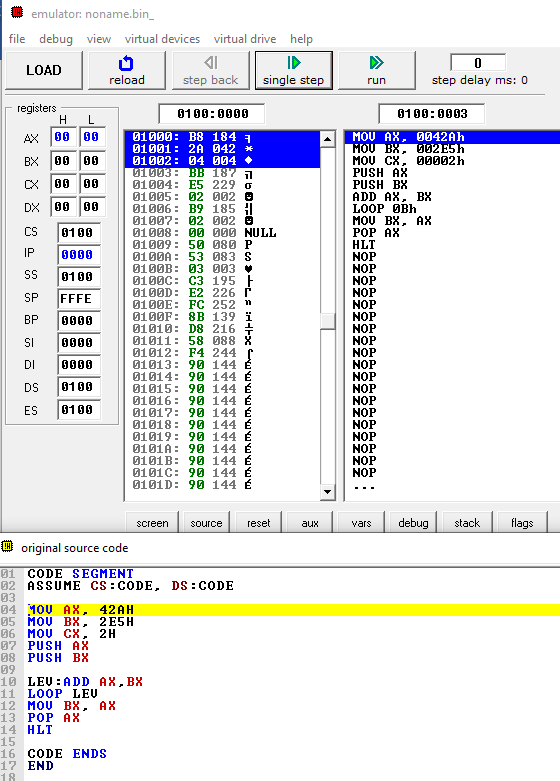
## RESULT

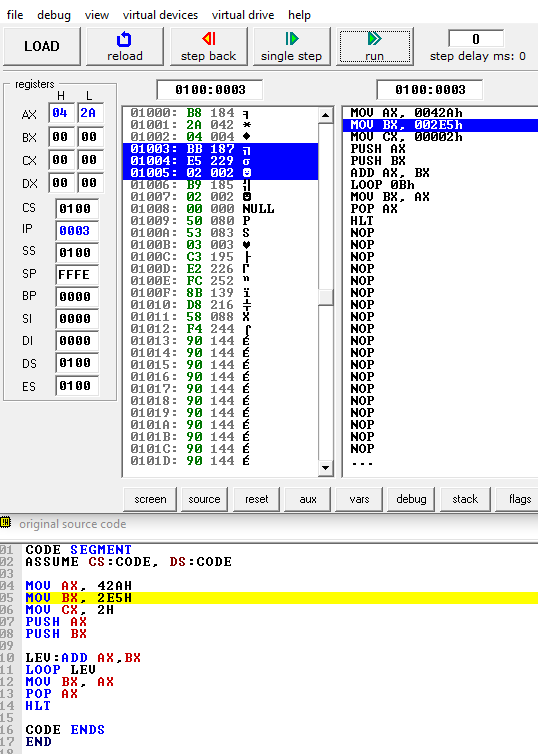
* 1. **Introduction to stack**

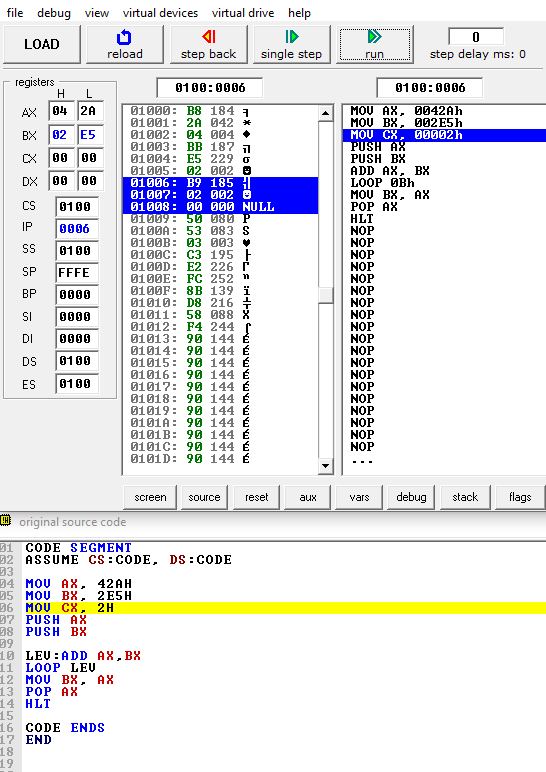
1. Observe the contents of the specified registers below after each instruction has been executed and record your result in Table 1.

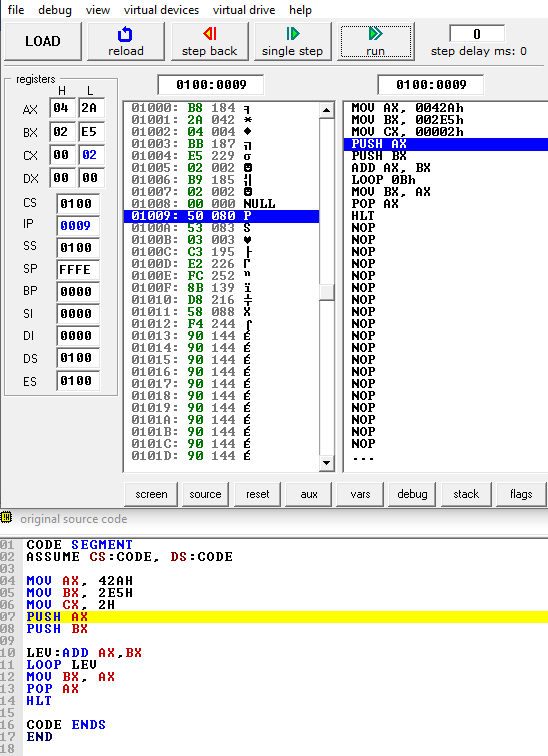
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Register Content | | | | |
| AX | BX | CX | IP | SP |
| 1. MOV AX, 42AH | 042A | 0000 | 0000 | 0003 | FFFE |
| 2. MOV BX, 2E5H | 042A | 02E5 | 0000 | 0006 | FFFE |
| 3. MOV CX, 2H | 042A | 02E5 | 0002 | 0009 | FFFE |
| 4. PUSH AX | 042A | 02E5 | 0002 | 000A | FFFC |
| 5. PUSH BX | 042A | 02E5 | 0002 | 000B | FFFA |
| 6. LEV:ADD AX,BX | 070F | 02E5 | 0002 | 000D | FFFA |
| 7. LOOP LEV | 070F | 02E5 | 0001 | 000B | FFFA |
| 8. ADD AX, BX | 09F4 | 02E5 | 0001 | 000D | FFFA |
| 9. LOOP LEV | 09F4 | 02E5 | 0000 | 000F | FFFA |
| 10. MOV BX, AX | 09F4 | 09F4 | 0100 | 0011 | FFFA |
| 11. POP AX | 02E5 | 09F4 | 0000 | 0012 | FFFC |

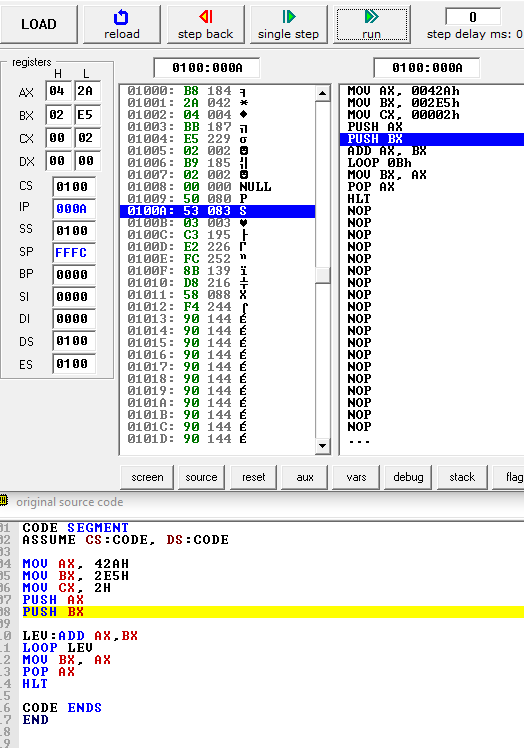
**Table 1**: Registers value

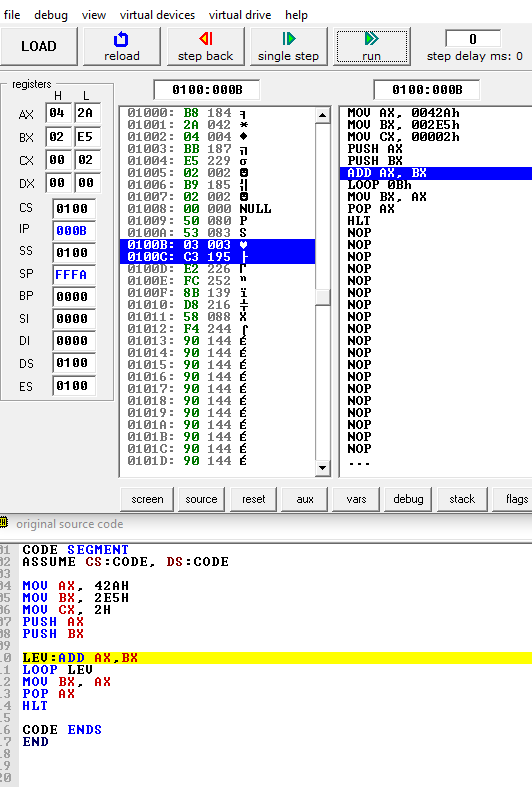


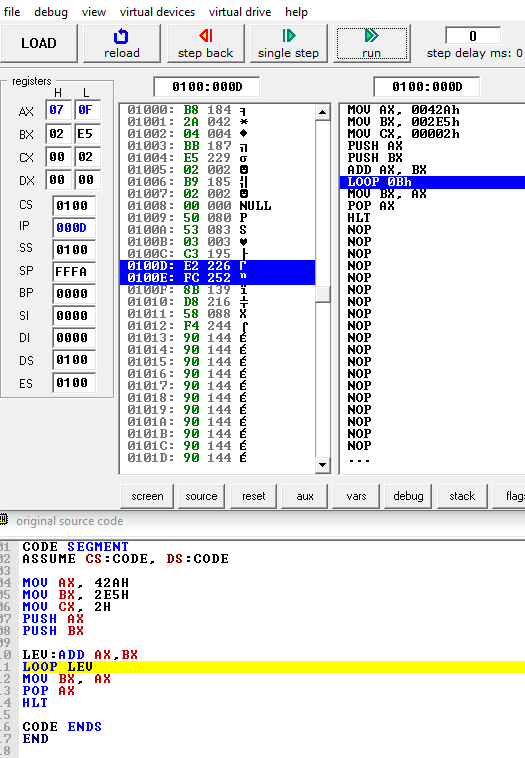


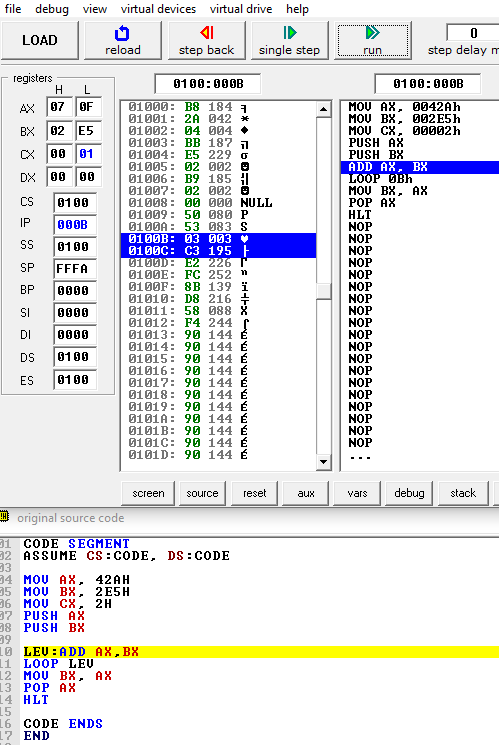


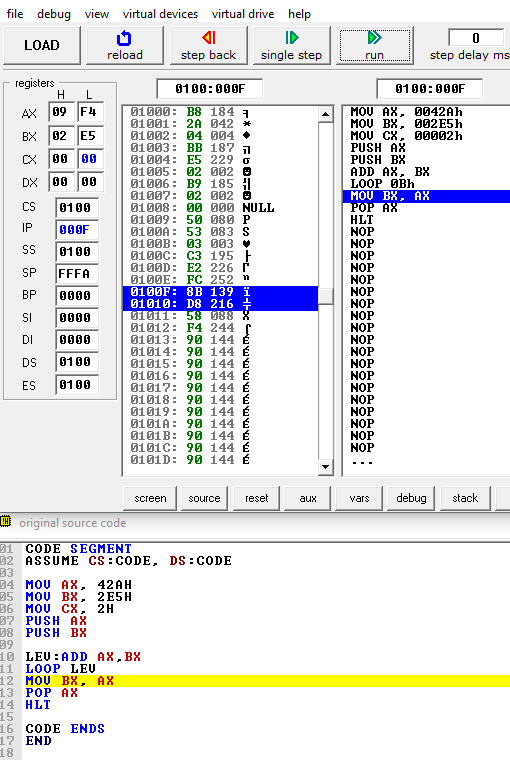


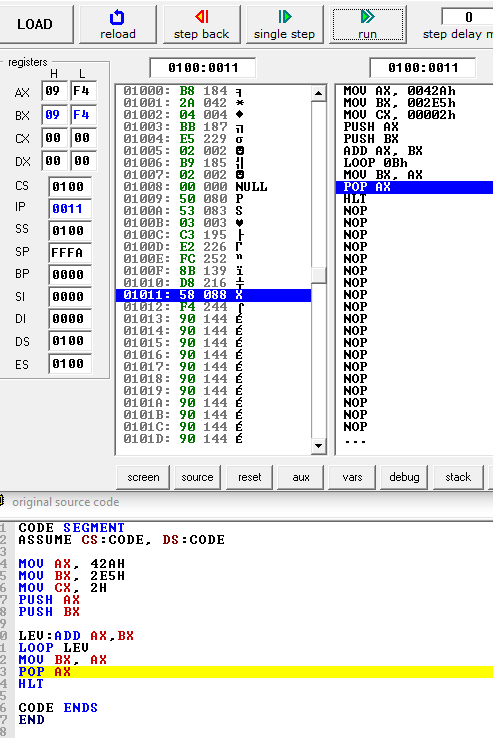


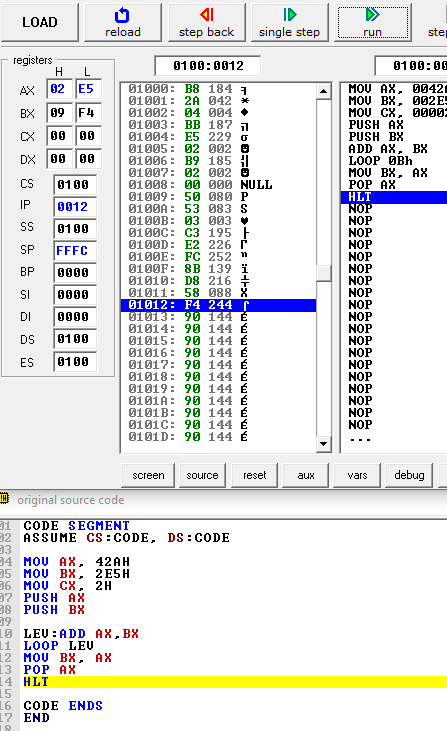












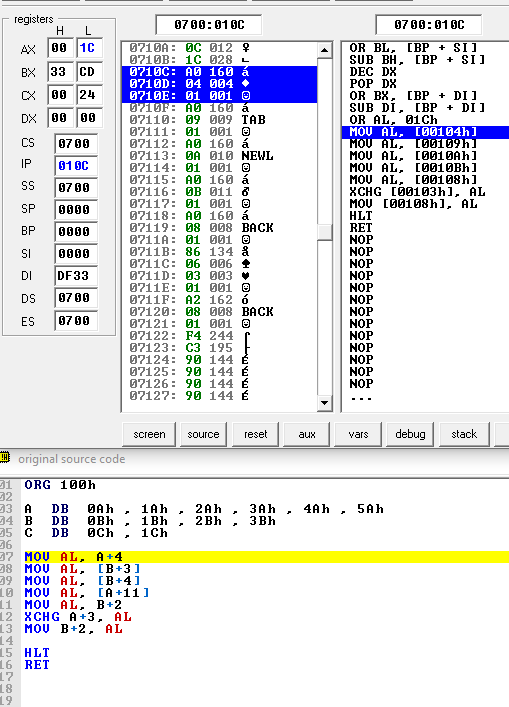
1. Clearly observe the program and state what happens in every line. Explain in your discussion.

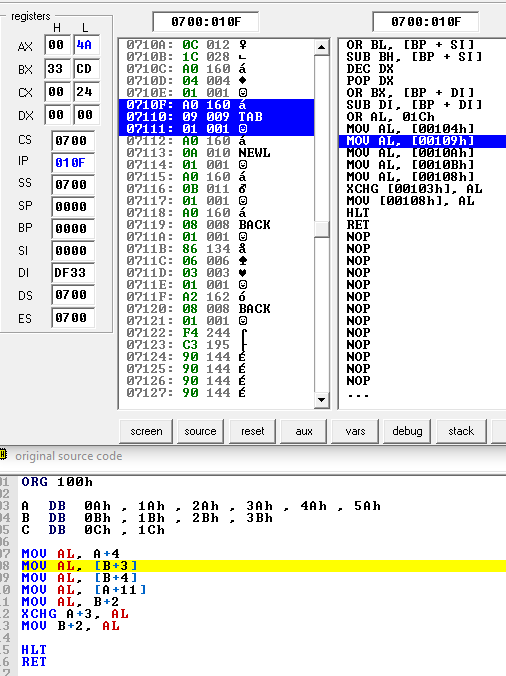
**5.2 Introduction to array**

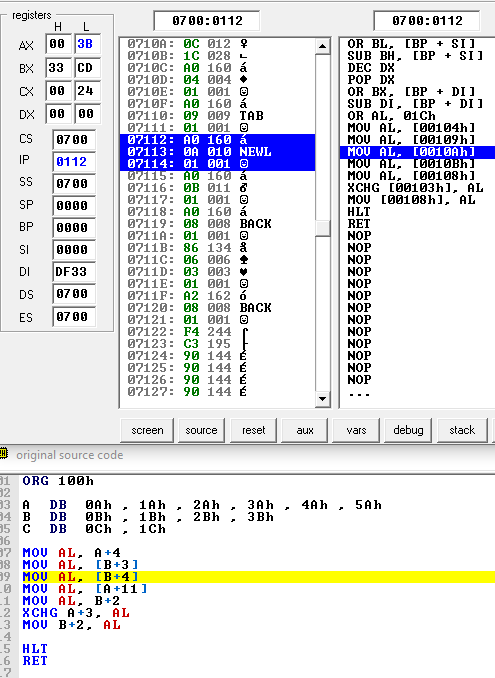
1. Observe the contents of the specified registers below after each instruction has been executed and record your result in Table 2.

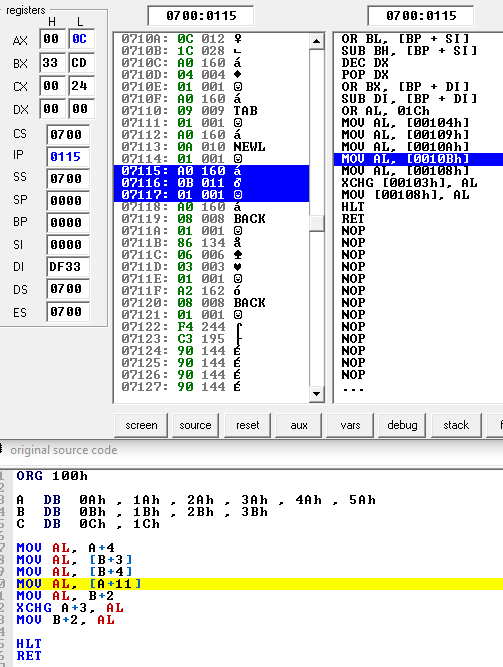
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Register Content | | | | |
| AX | BX | CX | IP | SP |
| 1. MOV AL, A+4 | 004A | 33CD | 0024 | 010F | 0000 |
| 2. MOV AL, [B+3] | 003B | 33CD | 0024 | 0112 | 0000 |
| 3. MOV AL, [B+4] | 000C | 33CD | 0024 | 0115 | 0000 |
| 4. MOV AL, [A+11] | 001C | 33CD | 0024 | 0118 | 0000 |
| 5. MOV AL, B+2 | 002B | 33CD | 0024 | 011B | 0000 |
| 6. XCHG A+3, AL | 003A | 33CD | 0024 | 011F | 0000 |
| 7. MOV B+2, AL | 003A | 33CD | 0024 | 0122 | 0000 |

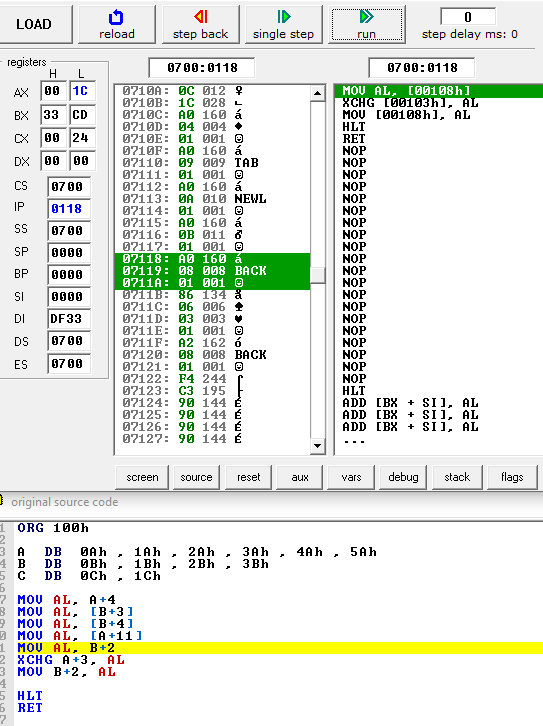
**Table 2**: Registers value

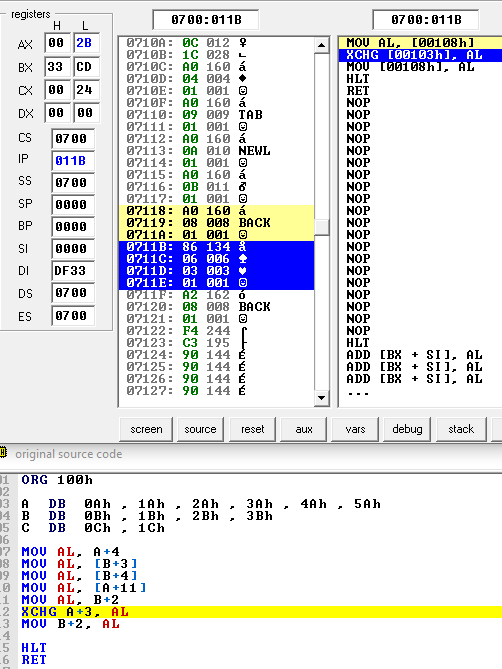


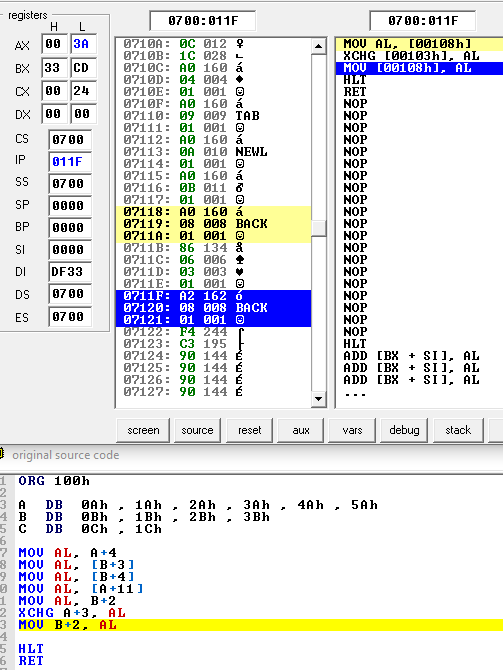


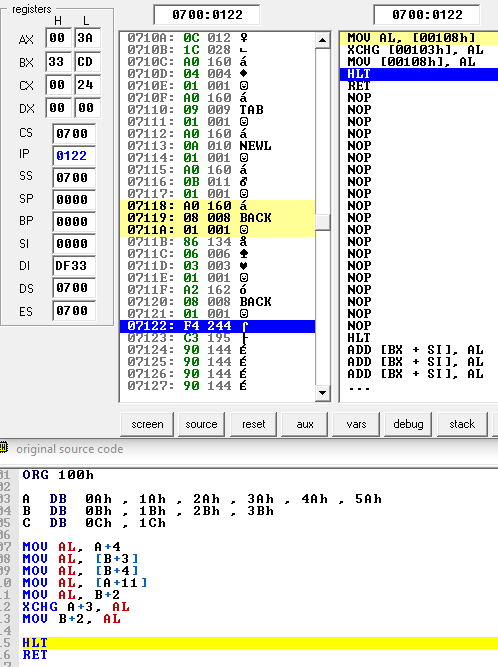












## DISCUSSION

## CONCLUSION