CSCI 592

LAB ASSIGNMENT – 3

Written by

DINESH SEVETI

Date: 02-15-2025

**OBJECTIVE:**

The objective of this lab was to understand and implement memory manipulation and register operations using assembly language. The focus was on observing how data moves within registers and memory locations, modifying values, and analyzing execution logs.

**TECHNOLOGY USED**

* Easy68K Assembler software to run the code.
* Hypothetical or real CPU with registers and memory

**PROCEDURE**

* Initialize Address Registers: Assign memory locations to registers (A3, A4, A6) using the LEA.L instruction.
* Store Values in Memory: Use the MOVE.L instruction to place predefined values at specific memory locations.
* Modify Registers: Perform operations on data registers (D0–D7) to observe changes.
* Execute the Program: Run the assembly code in EASy68K to observe memory and register changes.
* Analyze Execution Log: Compare initial and final register/memory values to confirm correct execution.
* Verify Results: Ensure that values in registers and memory match expected outputs.

**OPERATIONS**

* Load Effective Address (LEA.L): Used to load addresses into registers.
* Move Operations (MOVE.L): Used to transfer values between registers and memory.
* Register Manipulation: Observed how values change after execution.

**ALGORITHM**

* Initialize registers and memory addresses.
* Store predefined values in memory locations.
* Modify register values using operations.
* Execute the program and analyze register/memory states.
* Compare execution log with expected output.

**CODE LISTING**

START: ; first instruction of program

LEA.L $002468FA,A3

LEA.L $00002544,A4

LEA.L $00002518,A6

MOVE.L #$44334241, $00002518

MOVE.L #$00000000, $0000251C

MOVE.L #$25530000, $00002520

MOVE.L #$01EFABCD, $00002524

MOVE.L #$5467CC22, $00002528

MOVE.L #$FF3412FF, $0000252C

MOVE.L #$A2671FEE, $00002544

MOVE.L #$FFFFFFFF, $00002548

MOVE.L #$01000000, $0000254C

MOVE.L #$ABCDFFFF, $00002550

MOVE.L #$00000000, $00002554

MOVE.L $00000000, D3

MOVE.L $FFFFFFFF, D5

MOVE.L $00000000, D6

MOVE.B 3(A6) ,D3

MOVE.B 4(A4) ,6(A6)

MOVE.B (A6)+,D5

MOVE.B (A4)+,$002522

MOVE.B $00252E, -(A4)

LEA.L $00252A, A3

MOVE.W (A3),D6

SIMHALT ; halt simulator

END START ; last line of source

**DESCRIPTION**

This lab involved executing a sequence of assembly instructions to manipulate memory and registers. Initially, memory locations were assigned to specific address registers using LEA.L. Values were then stored in memory using MOVE.L instructions. The final execution log was analyzed to confirm changes in both memory and register states.

**OBSERVATIONS**

* The LEA.L instruction correctly assigned memory addresses to registers.
* The MOVE.L instructions successfully transferred values to designated memory locations.
* Register and memory values changed as expected, confirming correct execution.
* The execution log matched the expected output, indicating successful implementation.

**RESULTS**

**LOG FILES**

Before execution

D0=00000000 D4=00000000 A0=00000000 A4=00000000 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00000000 US=00FF0000

D3=00000000 D7=00000000 A3=00000000 A7=01000000 SS=01000000

PC=00000000 Code=47F9 002468FA Line= 12 LEA.L $002468FA,A3

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00000000 US=00FF0000

D3=00000000 D7=00000000 A3=00000000 A7=01000000 SS=01000000

PC=00000006 Code=49F8 2544 Line= 13 LEA.L $00002544,A4

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00000000 US=00FF0000

D3=00000000 D7=00000000 A3=00000000 A7=01000000 SS=01000000

PC=0000000A Code=4DF8 2518 Line= 14 LEA.L $00002518,A6

After Execution

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=00000016 Code=21FC 44334241 2518 Line= 19 MOVE.L #$44334241, $00002518

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=0000001E Code=21FC 25530000 2520 Line= 20 MOVE.L #$25530000, $00002520

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=00000026 Code=21FC 01EFABCD 2524 Line= 22 MOVE.L #$01EFABCD, $00002524

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=0000002E Code=21FC 5467CC22 2528 Line= 23 MOVE.L #$5467CC22, $00002528

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000000000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=00000036 Code=21FC FF3412FF 252C Line= 24 MOVE.L #$FF3412FF, $0000252C

D0=00000000 D4=00000000 A0=00000000 A4=00002544 T\_S\_\_INT\_\_\_XNZVC

D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000001000

D2=00000000 D6=00000000 A2=00000000 A6=00002518 US=00FF0000

D3=00000000 D7=00000000 A3=002468FA A7=01000000 SS=01000000

PC=0000003E Code=21FC A2671FEE 2544 Line= 25 MOVE.L #$A2671FEE, $00002544

Before Execution:

| Address | Value |
| --- | --- |
| 0x2518 | 0x4433 |
| 0x251C | 0x0000 |
| 0x2520 | 0x2553 |
| 0x2524 | 0x01EF |
| 0x2528 | 0x5467 |
| 0x252C | 0xFF34 |
| 0x2530 | 0x12FF |
| 0x2544 | 0xA267 |
| 0x2548 | 0xFFFF |
| 0x254C | 0x0100 |
| 0x2550 | 0xABCD |
| 0x2554 | 0x0000 |

After Execution

| Address | Value |
| --- | --- |
| 0x2518 | 0x44334241 |
| 0x251C | 0x00000000 |
| 0x2520 | 0x25530000 |
| 0x2524 | 0x01EFABCD |
| 0x2528 | 0x5467CC22 |
| 0x252C | 0xFF3412FF |
| 0x2530 | 0x12FF |
| 0x2544 | 0xA2671FEE |
| 0x2548 | 0xFFFFFFF |
| 0x254C | 0x01000000 |
| 0x2550 | 0xABCDFFFF |
| 0x2554 | 0x00000000 |

**CONCLUSIONS**

The lab successfully demonstrated memory manipulation and register operations using assembly language. Understanding these low-level operations provides insight into CPU functionality and data management. The ability to track changes in registers and memory is crucial for debugging and optimizing assembly programs.