CSCI 592

LAB ASSIGNMENT – 6

LAB 5.C

Written by

DINESH SEVETI

Date: 03-22-2025

**OBJECTIVE**

The objective of this lab is to understand the concepts of pointers and linked data structures at the architecture level. The experiment focuses on constructing a memory image, establishing pointers to linked elements, and inserting a new element between two existing elements in a linked list.

**TECHNOLOGY USED**

1. **Easy68K Assembler** software to write and execute the assembly code.

**PROCEDURE**

* Initialize memory locations with the given linked list structure.
* Set up pointer registers (A1, A2, A5, A6) to establish links between elements.
* Insert a new element by modifying pointer values so that it is positioned between two existing elements.
* Display the memory before and after insertion to verify the operation.
* Halt the simulator to stop execution after the insertion operation.

**OPERATIONS**

* Memory Initialization: Setting up elements in memory to simulate a linked list.
* Pointer Manipulation: Using registers to access and modify linked list elements.
* Element Insertion: Adjusting the linked list structure to insert a new element between two existing elements.

**ALGORITHM**

* Define memory locations for the linked list elements.
* Load **A6** with the address of the first element (e1).
* Set **A1** to point to the second element (e2) using its stored address.
* Set **A2** to point to the third element (e3) using e2’s next pointer.
* Load **A5** with the address of the new element.
* Modify **A5's next pointer** to point to e3.
* Update **e2’s next pointer** to point to A5.
* Halt execution after completing the insertion.

**CODE LISTING**

START: ; first instruction of program

\* Put program code here

MOVE.L #$41414141, $74A8

MOVE.L #$000074D0, $74AC

MOVE.L #$00000000, $74B0

MOVE.L #$00000000, $74B4

MOVE.L #$43434343, $74B8

MOVE.L #$000074C0, $74BC

MOVE.L #$44444444, $74C0

MOVE.L #$00000000, $74C4

MOVE.L #$00000000, $74C8

MOVE.L #$00000000, $74CC

MOVE.L #$42424242, $74D0

MOVE.L #$000074B8, $74D4

LEA.L $00000000, A1

LEA.L $00000000, A2

LEA.L $00000000, A6

LEA.L $00000000, A5

LEA.L $00000000, A3

LEA.L $000074A8,A6

MOVE.L 4(A6),A1

MOVE.L 4(A1),A2

LEA.L $000074B0,A5

MOVE.L A2,4(A5)

MOVE.L A5,4(A1)

SIMHALT ; halt simulator

\* Put variables and constants here

END START ; last line of source

**DESCRIPTION**

* The given code initializes a linked list in memory with elements e1, e2, e3, and e4.
* A6, A1, and A2 are used to traverse the linked list.
* A new element at $74B0 is inserted between e2 and e3 by updating their respective next pointers.
* The simulator halts after the insertion to allow verification.

**OBSERVATIONS**

* Before insertion, **e2’s next pointer** pointed to **e3**.
* After execution, **e2 now points to the new element** and the new element points to **e3**.
* The linked list structure is maintained correctly after insertion.

**RESULTS**

|  |  |  |
| --- | --- | --- |
| Address | Data | Description |
| $74A8 | 41414141 | e1 Data |
| $74AC | 000074D0 | e1 → e2 |
| $74B0 | 00000000 | New element Data |
| $74B4 | 000074B8 | New element → e3 |
| $74B8 | 43434343 | e3 Data |
| $74BC | 000074C0 | e3 → e4 |
| $74C0 | 44444444 | e4 Data |
| $74C4 | 00000000 | NULL |
| $74D0 | 42424242 | e2 Data |
| $74D4 | 000074B0 | e2 → New element |

**CONCLUSIONS**

The experiment successfully demonstrated **pointer manipulation and linked list insertion** using **assembly language**. The linked list structure was maintained correctly after modifying the next pointers. Understanding pointers at the architecture level is essential for **efficient memory management and dynamic data structures**.