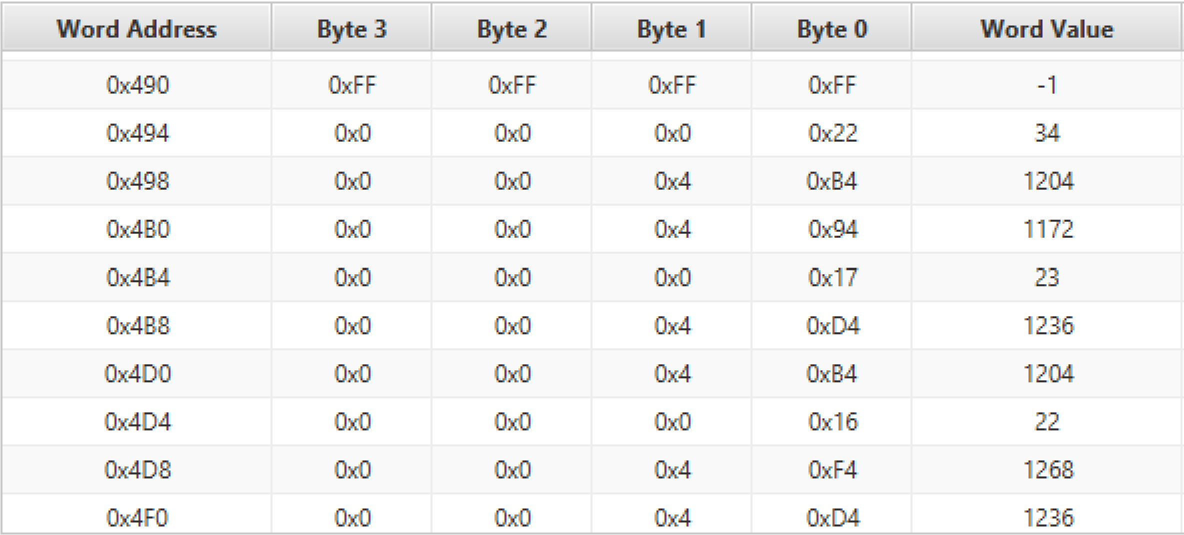
Luciano Giannini
Procedure Calls in ARM Assembly

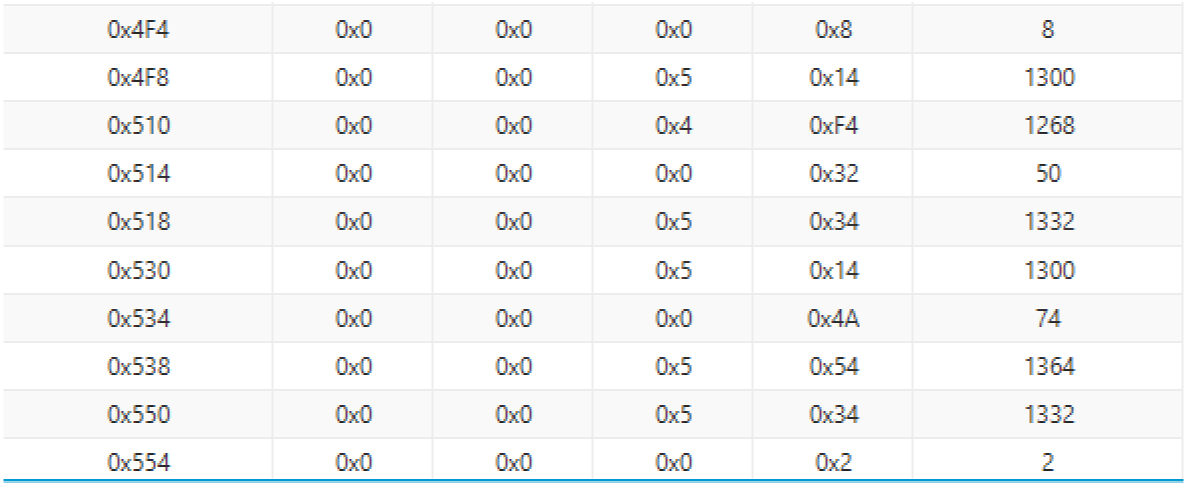
All code/methods are functional for creating the doubly linked list and performing insertion sort on that list.
If there is duplicate data, during the sorting method we will skip over that data node and connect to the next item in the list via pointer manipulation.
When you run the code as-is, it will create the doubly linked list, then sort the list and then run a print function at the end to display the values read from the head of the list to the end, which can be seen in the register 8's history.

## Extra Information

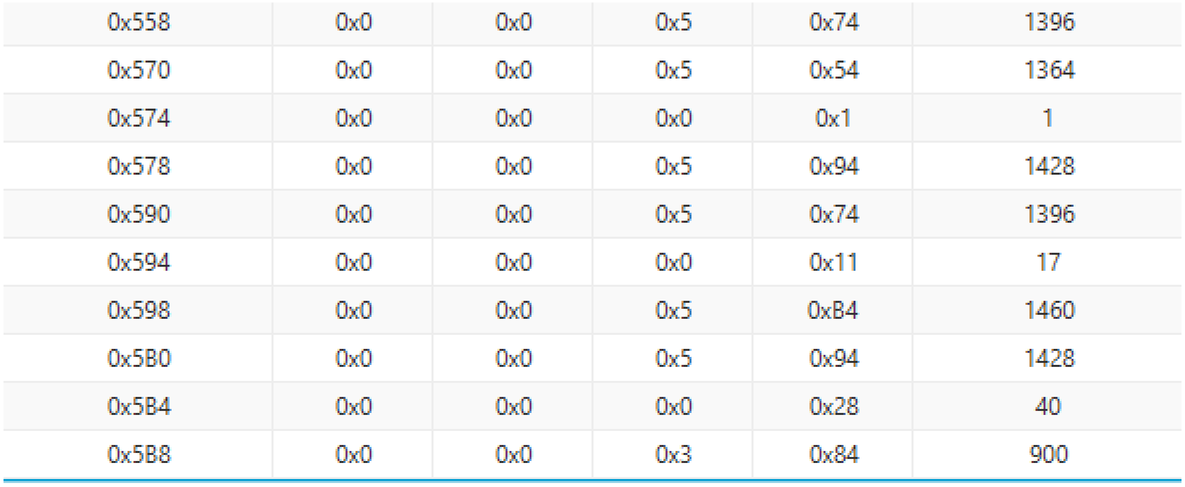
Some checks I performed to see if my code was working as intended (with screenshots of data memory and R8 history for the print function):
1: Checked memory location of each node created before the sort, for the given example data array.



| Word Address | Byte 3 | Byte 2 | Byte 1 | Byte 0 | Word Value |
|---|---|---|---|---|---|
| 0x490 | 0xFF | 0xFF | 0xFF | 0xFF | -1 |
| 0x494 | 0x0 | 0x0 | 0x0 | 0x22 | 34 |
| 0x498 | 0x0 | 0x0 | 0x4 | 0xB4 | 1204 |
| 0x4B0 | 0x0 | 0x0 | 0x4 | 0x94 | 1172 |
| 0x4B4 | 0x0 | 0x0 | 0x0 | 0x17 | 23 |
| 0x4B8 | 0x0 | 0x0 | 0x4 | 0xD4 | 1236 |
| 0x4D0 | 0x0 | 0x0 | 0x4 | 0xB4 | 1204 |
| 0x4D4 | 0x0 | 0x0 | 0x0 | 0x16 | 22 |
| 0x4D8 | 0x0 | 0x0 | 0x4 | 0xF4 | 1268 |
| 0x4F0 | 0x0 | 0x0 | 0x4 | 0xD4 | 1236 |



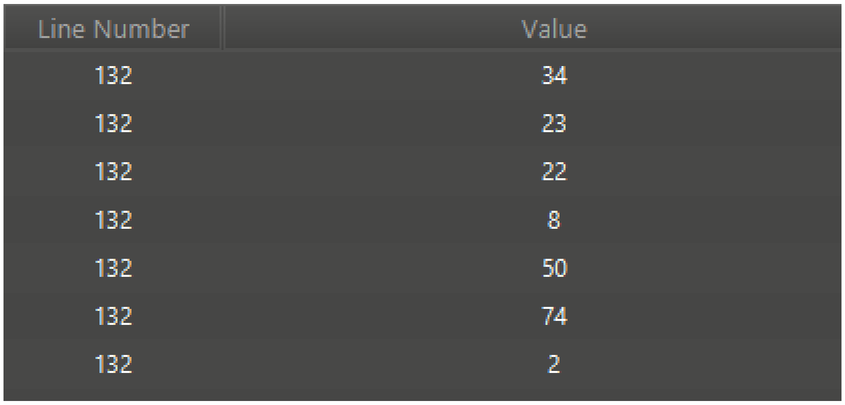
| | | | | | |
|---|---|---|---|---|---|
| 0x4F4 | 0x0 | 0x0 | 0x0 | 0x8 | 8 |
| 0x4F8 | 0x0 | 0x0 | 0x5 | 0x14 | 1300 |
| 0x510 | 0x0 | 0x0 | 0x4 | 0xF4 | 1268 |
| 0x514 | 0x0 | 0x0 | 0x0 | 0x32 | 50 |
| 0x518 | 0x0 | 0x0 | 0x5 | 0x34 | 1332 |
| 0x530 | 0x0 | 0x0 | 0x5 | 0x14 | 1300 |
| 0x534 | 0x0 | 0x0 | 0x0 | 0x4A | 74 |
| 0x538 | 0x0 | 0x0 | 0x5 | 0x54 | 1364 |
| 0x550 | 0x0 | 0x0 | 0x5 | 0x34 | 1332 |
| 0x554 | 0x0 | 0x0 | 0x0 | 0x2 | 2 |



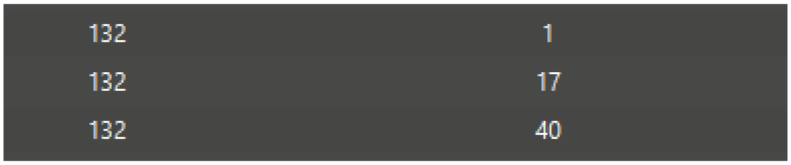
| | | | | | |
|---|---|---|---|---|---|
| 0x558 | 0x0 | 0x0 | 0x5 | 0x74 | 1396 |
| 0x570 | 0x0 | 0x0 | 0x5 | 0x54 | 1364 |
| 0x574 | 0x0 | 0x0 | 0x0 | 0x1 | 1 |
| 0x578 | 0x0 | 0x0 | 0x5 | 0x94 | 1428 |
| 0x590 | 0x0 | 0x0 | 0x5 | 0x74 | 1396 |
| 0x594 | 0x0 | 0x0 | 0x0 | 0x11 | 17 |
| 0x598 | 0x0 | 0x0 | 0x5 | 0xB4 | 1460 |
| 0x5B0 | 0x0 | 0x0 | 0x5 | 0x94 | 1428 |
| 0x5B4 | 0x0 | 0x0 | 0x0 | 0x28 | 40 |
| 0x5B8 | 0x0 | 0x0 | 0x3 | 0x84 | 900 |

The beginning terminal is -1 for the previous of the first element of the list. The last address points to the next element in the list but I set it to 900.

2. The next thing I did was print these values out. We get:



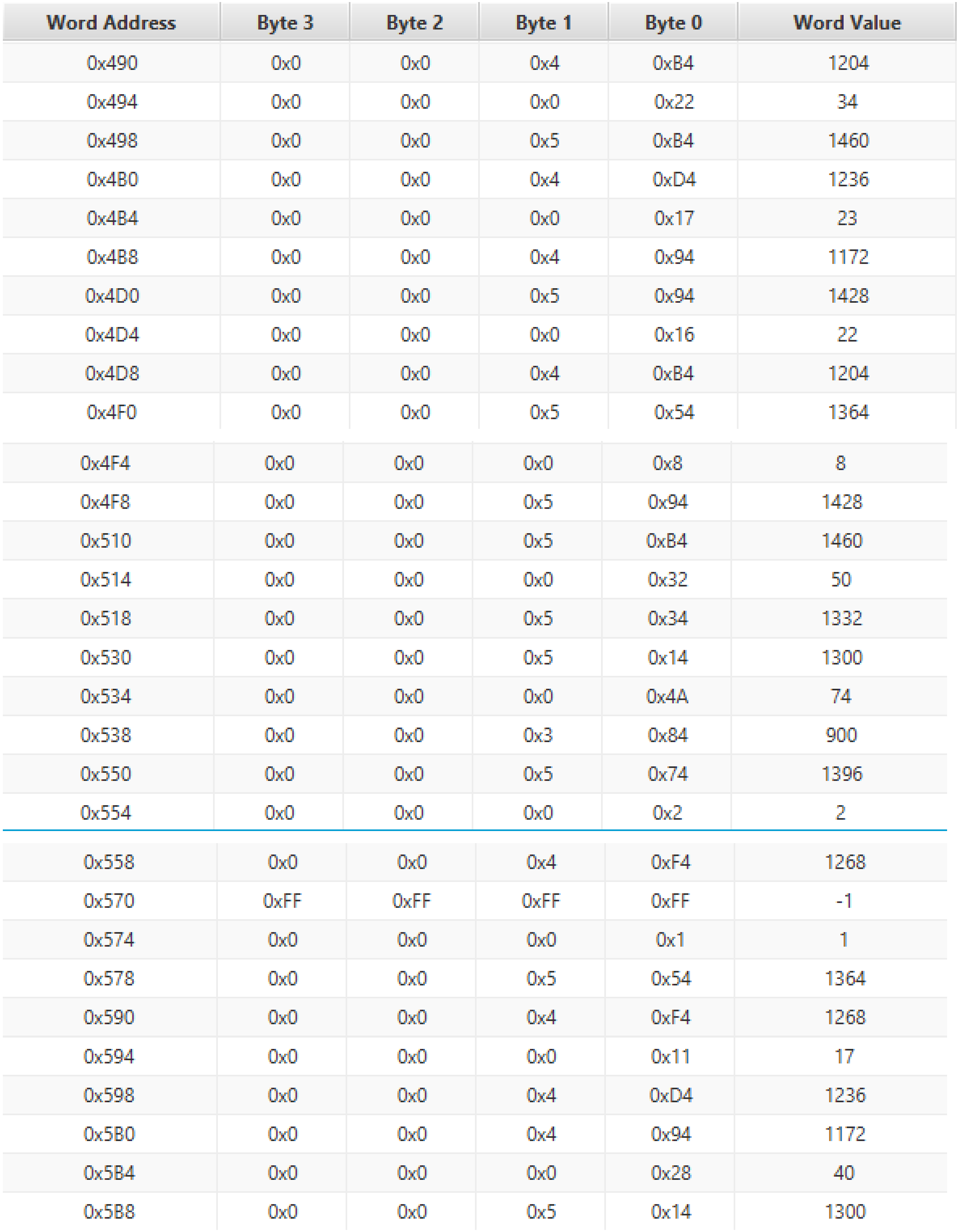
| Line Number | Value |
|---|---|
| 132 | 34 |
| 132 | 23 |
| 132 | 22 |
| 132 | 8 |
| 132 | 50 |
| 132 | 74 |
| 132 | 2 |



| | |
|---|---|
| 132 | 1 |
| 132 | 17 |
| 132 | 40 |

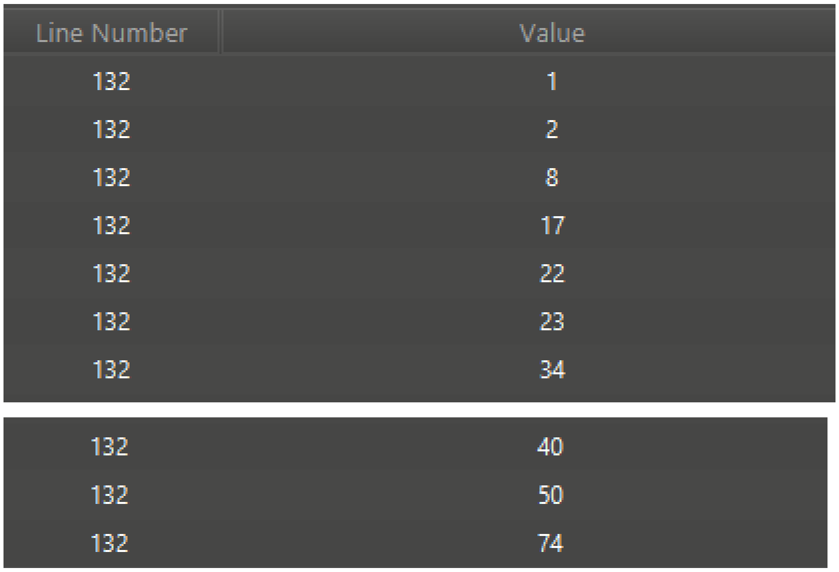
Looks good!

3. Now sorting time.



| Word Address | Byte 3 | Byte 2 | Byte 1 | Byte 0 | Word Value |
|---|---|---|---|---|---|
| 0x490 | 0x0 | 0x0 | 0x4 | 0xB4 | 1204 |
| 0x494 | 0x0 | 0x0 | 0x0 | 0x22 | 34 |
| 0x498 | 0x0 | 0x0 | 0x5 | 0xB4 | 1460 |
| 0x4B0 | 0x0 | 0x0 | 0x4 | 0xD4 | 1236 |
| 0x4B4 | 0x0 | 0x0 | 0x0 | 0x17 | 23 |
| 0x4B8 | 0x0 | 0x0 | 0x4 | 0x94 | 1172 |
| 0x4D0 | 0x0 | 0x0 | 0x5 | 0x94 | 1428 |
| 0x4D4 | 0x0 | 0x0 | 0x0 | 0x16 | 22 |
| 0x4D8 | 0x0 | 0x0 | 0x4 | 0xB4 | 1204 |
| 0x4F0 | 0x0 | 0x0 | 0x5 | 0x54 | 1364 |
| 0x4F4 | 0x0 | 0x0 | 0x0 | 0x8 | 8 |
| 0x4F8 | 0x0 | 0x0 | 0x5 | 0x94 | 1428 |
| 0x510 | 0x0 | 0x0 | 0x5 | 0xB4 | 1460 |
| 0x514 | 0x0 | 0x0 | 0x0 | 0x32 | 50 |
| 0x518 | 0x0 | 0x0 | 0x5 | 0x34 | 1332 |
| 0x530 | 0x0 | 0x0 | 0x5 | 0x14 | 1300 |
| 0x534 | 0x0 | 0x0 | 0x0 | 0x4A | 74 |
| 0x538 | 0x0 | 0x0 | 0x3 | 0x84 | 900 |
| 0x550 | 0x0 | 0x0 | 0x5 | 0x74 | 1396 |
| 0x554 | 0x0 | 0x0 | 0x0 | 0x2 | 2 |
| 0x558 | 0x0 | 0x0 | 0x4 | 0xF4 | 1268 |
| 0x570 | 0xFF | 0xFF | 0xFF | 0xFF | -1 |
| 0x574 | 0x0 | 0x0 | 0x0 | 0x1 | 1 |
| 0x578 | 0x0 | 0x0 | 0x5 | 0x54 | 1364 |
| 0x590 | 0x0 | 0x0 | 0x4 | 0xF4 | 1268 |
| 0x594 | 0x0 | 0x0 | 0x0 | 0x11 | 17 |
| 0x598 | 0x0 | 0x0 | 0x4 | 0xD4 | 1236 |
| 0x5B0 | 0x0 | 0x0 | 0x4 | 0x94 | 1172 |
| 0x5B4 | 0x0 | 0x0 | 0x0 | 0x28 | 40 |
| 0x5B8 | 0x0 | 0x0 | 0x5 | 0x14 | 1300 |

4. And print list out:

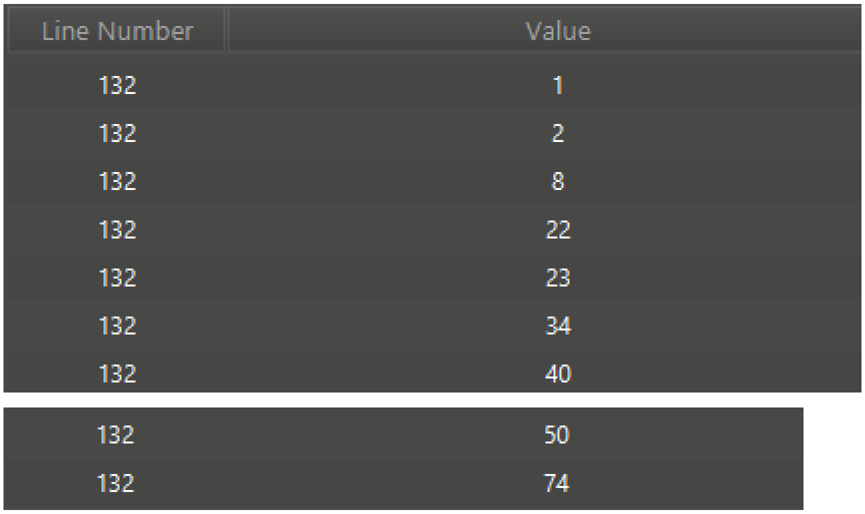


| Line Number | Value |
|---|---|
| 132 | 1 |
| 132 | 2 |
| 132 | 8 |
| 132 | 17 |
| 132 | 22 |
| 132 | 23 |
| 132 | 34 |
| 132 | 40 |
| 132 | 50 |
| 132 | 74 |

Looks good!

5. The last function was to delete. This will delete any value that is a duplicate in the list.
Example List: 34, 23, 22, 8, 50, 74, 2, 1, 8, 40

Printing out we get:



| Line Number | Value |
|---|---|
| 132 | 1 |
| 132 | 2 |
| 132 | 8 |
| 132 | 22 |
| 132 | 23 |
| 132 | 34 |
| 132 | 40 |
| 132 | 50 |
| 132 | 74 |

No double 8! Looks Good! And the node with the duplicate 8 has its next and previous pointer set to -1.