Name : Nguyen Quoc Thang

ID : 104193360

**LAB 7 : Submission**

*Click on any visible memory word and type in 101 (followed by  the "Enter" key).*

**7.1.1 What value is displayed ?    Why ?**

The memory display 0x00000065 because it convert 101 decemal into hexadecimal.

*Click on another memory word, enter 0x101*

**7.1.2 What value is displayed, and why?**

Lalue starting with 0x means the value is represented in hexadecimal so the word memory will display 0x00000101

On another memory word, enter 0b101

***7.1.3 What value is displayed, and why?***

The memory display 0x00000005 because it convert 0b101 form binary to hexadecimal.

The Tooltip function display binary number of hexadecimal memory of value that have been filled in memory before

When changee the base from hexadecimal from Decimal(unsigned) the Tooltip function display both

***7.1.4: Does changing the representation of the data in memory also change the representation of the row and column-headers (the white digits on a blue background)?  Should it ?***

Changing the representation of the data in memory does not changes the representation of the row and column because row and column headers usually represent the address of the memory, which is independent of the memory's value

***7.2.1  Notice these column header memory address offsets go up in multiples of 0x4.   Why is this ?***

Each memory word has 32 bits it is equivalent to 4 bytes so this column header memory address offsets increases by the number set of 0x4

**7.3.1  Take a screen shot of the simulator in full and add it to your submission document**

A picture containing text, screenshot, software, computer icon

Description automatically generated

**7.3.2  Based on what we've learnt about assemblers and Von Neuman architectures,  explain what you think just happened.**

**After the code has been submitted, the assembler proram convert the source code into machine code and then the machine code was stored into the memory space**

**7.3.3  Based on what we have learnt about memory addressing in ARMlite, and your response to 7.3.2, what do you think this value represents ?**

The hexadecimal values appears whenever I hover the mouse over one of the line numbers of the source code. It represents the location of the memory space that contains the instruction.

***The answer to the try inserting part is below question 7.3.3***

-The blank lines disappeared and it does not save in the memory space.

- When we add comment on a line of its own or after an instruction after the code has sumitted the comment will change the color difference from the code

- If add the comment on a line , the line numbers increases with the number of comments added

**7.4.1 What do you think the highlighting in both windows signifies ?**

The highlighting indicates what line of code run in the program and the highlighting in the memory represents for the location where the line of code is stored in the memory space.

**7.4.2  What do you think happens when you click the button circled in red**  ?

When I click the button circle red , the instructions run step by step , line by line.

**7.4.3 Has the processor paused just before, or just after executing the line with the breakpoint ?**

**The processor pause jusst before executing the line with the breakpoint .**

**7.5.1 Before executing this instruction, describe in words what you think this instruction is going to do, and what values you expect to see in R0 and R1  when it is complete ?**

MOV R0,#1 The value 1 is moved to register 0 . R0 will be equal to 1

ADD R1,R0,#8 The value R0 is added to 8 and the result will be stored in R1

ADD R2,R1,#100 The value R0 is added to 100 and the result wil be store in R2

SUB R3,R2,#25 The value of R2 subtracted fron 25, the result will be store in R3, the value of R3 in hex will display 54 but in decimal it will display 84

**-Value of R0 will be 1 and value of R1 will be 9 when it is complete.**

**7.5.2 When the program is complete, take a screen shot of the register table showing the values.A picture containing text, screenshot, software, computer icon

Description automatically generated**

**MOV R0, #1**

**ADD R1, R0 , #8**

**ADD R2, R1, #100**

**SUB R3, R2, #25**

**HALT**

**7.5.3  Task:  Your 6 initial numbers are now 300, 21, 5, 64, 92, 18.   Write an Assembly Program that uses these values to compute a final value of 294  (you need only use MOV, ADD and SUB).  Place your final result in register R7  (don't forget the HALT instruction)**

A picture containing text, screenshot, software, display

Description automatically generated

MOV R0, #300

SUB R1, R0, #21

ADD R2, R1, #5

MOV R3, #64

SUB R4, R2, R3

MOV R5 , #92

ADD R6, R4, R5

SUB R7, R6, #18

HALT

**7.5.4  Task:  Write your own simple program, that starts with a MOV (as in the previous example) followed by five instructions, using each of the five new instructions listed above, once only, but in any order you like – plus a HALT at the end, and with whatever immediate values you like.**

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Decimal value of the destination register after executing this instruction** | **Binary value of the destination register after executing this instruction** |
| MOV R0,#13 | 13 | 00001101 |
| AND R1, R0, #10 | 8 | 00000100 |
| ORR R2, R1, #3 | 11 | 00001011 |
| EOR R3, R1, R0 | 5 | 00000101 |
| LSL R4, R1, #3 | 64 | 01000000 |
| LSR R5, R0, #5 | 0 | 00000000 |

**Task 7.5.5  Lets play the game we played in 7.5.3, but this time you can use any of the instructions listed in this lab so far (ie,. MOV, AND, OR, and any of the bit-wise operators).**

A picture containing text, screenshot, software, computer icon

Description automatically generated

MOV R0, #12

MOV R1, #11

MOV R10, #3

ADD R2, R0, R1

ADD R3, R2, #5

SUB R4, R3, #7

ADD R5, R10, #2

ADD R6,R4,R5

LSL R7, R0, #2

ADD R8, R7, #5

ADD R9, R8,R6

HALT

**Task 7.5.6:  Let's play again !**

Your six initial numbers are: 99, 77, 33, 31, 14, 12 and your target number is: 32

***When the program is complete, take a screen shot of the code and the register table and paste into your submission document.***

A picture containing text, screenshot, software, computer icon

Description automatically generated

MOV R0, #12

LSR R1, R0, #1

LSL R2, R0, #2

ADD R3, R2, R1

MOV R4, #99

ADD R5, R4 , #77

SUB R6, R5, #33

SUB R7, R6, #31

SUB R8, R7, #14

SUB R9, R8, #12

SUB R10, R9, R3

HALT

**7.6.1** **- Why is the result shown in R1 a negative decimal number, and with no obvious relationship to 9999 ?**

**The value of R1 is a negative decimal number because the instruction is shifting the 18 bits of the value in R0 to the left. In this case, the value of R0 is 9999, which is equivalent to the binary value 10011100001111. When it is shifted 18 bits to the left, it exceeds the range of positive numbers, resulting in a negative value. This negative value is then stored in R1 because R1 is not directly related to the value 9999.**

**7.6.3 - What is the binary representation of each of these signed decimal numbers: 1, -1, 2, -2  
What pattern do you notice ? Make a note of these in your submission document before reading on.**

**1 🡺 0b00000000000000000000000000000001**

**-1 🡺 0b11111111111111111111111111111111**

**2 🡺 0b00000000000000000000000000000010**

**-2 🡺 0b11111111111111111111111111111110**

**7.6.4** **- Write an ARM Assembly program that converts a positive decimal integer into its negative version.  Start by moving the input value into R0, and leaving the result in R1.**

A screenshot of a computer

Description automatically generated with medium confidence

***Using the program you wrote above, enter the negative version of the number you previously tested as the input (ie use the output of the previous test as the input).***

***What do you notice* ?**

* Negative value has changed to positive value

A screenshot of a computer

Description automatically generated with medium confidence