ECE IoT 505L – Embedded Programming – Lab Exercises

|  |  |  |
| --- | --- | --- |
| **S.No** | **Experiments** | **Tools used / Remarks** |
| 1. | 64 bit Addition – Assembly | Keil uVision IDE |
| 1. | Memory Block transfer - Assembly | Keil uVision IDE |
|  |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |

**Exercise 1**: Write an assembly program to add two 64 bit numbers in memory.

**Objective**: To write an assembly language program to add the two 64 bit numbers stored in the data memory of LPC1768 (ARM Cortex M3 Microcontroller from NXP).

**Let us consider two 64 bit numbers**

**X = 0x1234567812345678**

**Y = 0x2345678023456789 +**

**--------------------------------------**

**Z =** 0x3579BE013579BE01

**---------------------------------------**

**Update the memory 0x10000100 with the Least significant word of ”X” (0x12345678) (in Little Endian Order)**

**0x10000100 – 0x78**

**0x10000101 – 0x56**

**0x10000102 – 0x34**

**0x10000103 – 0x12**

**Update the memory 0x10000104 with the Most significant word of ”X” (0x12345678) (in Little Endian Order)**

**0x10000104 – 0x78**

**0x10000105 – 0x56**

**0x10000106 – 0x34**

**0x10000107 – 0x12**

**Update the memory 0x10000108 with the Least significant word of ”Y” (0x23456789) (in Little Endian Order)**

**0x10000108 – 0x89**

**0x10000109 – 0x67**

**0x1000010A – 0x45**

**0x1000010B – 0x23**

**Update the memory 0x1000010C with the Most significant word of ”Y” (0x23456789) (in Little Endian Order)**

**0x1000010C – 0x78**

**0x1000010D – 0x56**

**0x1000010E – 0x34**

**0x1000010F – 0x12**

1. g

**Assembly Program:**

**AREA mydata, DATA**

**X0 EQU 0x10000100**

**X1 EQU 0x10000104**

**Y0 EQU 0x10000108**

**Y1 EQU 0x1000010C**

**RE0 EQU 0x10000110**

**RE1 EQU 0x10000114**

**AREA gpio, CODE, READONLY, ALIGN=2**

**EXPORT \_\_main**

**ENTRY**

**\_\_main**

**LDR R0, =X0 ;Get the address of Least significant word(X0)'s address**

**LDR R1, [R0]**

**LDR R0, =X1 ;Get the address of Most significant word(X1)'s address**

**LDR R2, [R0]**

**LDR R0, =Y0 ;Get the address of Least significant word(Y0)'s address**

**LDR R3, [R0]**

**LDR R0, =Y1 ;Get the address of Most significant word(Y1)'s address**

**LDR R4, [R0]**

**ADDS R1, R1,R3**

**ADC R2, R2, R4**

**LDR R0, =RE0**

**STR R1, [R0]**

**LDR R0, =RE1**

**STR R2, [R0]**

**LBL B LBL**

**END**

**Output:**

Figure 1. Watch window showing PORTB value as 0x01

A screenshot of a computer

Description automatically generated

Figure 1. Memory Window View

**Work for Students: (To be submitted on or before 29/10/21)**

1. Modify the program to include carry.
2. Modify the program to do subtraction of two 64 bit numbers with borrow.
3. Write a program to Multiply two 64 bit numbers and get the 64 bit results

**Conclusion:** Thus, an assembly program is written to add two 64 bit numbers in memory and put the result back to memory.

Project folder: Arithmetic

**Exercise 2:** Write a ARM assembly program to fill the section of memory with particular byte and then transfer the block of memory to another location

**Objective**: To write an assembly language program to fill the section of memory with 0xAA and then transfer the block of memory (100 bytes) to another location. And the program will be simulated.

**Algorithm:**

**Assembly Program:**

AREA mydata, DATA

FILL\_VAL EQU 0xAAAAAAAA

SOURCE EQU 0x10000100

DEST EQU 0x10000300

COUNT EQU 100

AREA MEM\_OP, CODE, READONLY, ALIGN=2

EXPORT \_\_main

ENTRY

\_\_main

LDR R0, =SOURCE

LDR R1, =FILL\_VAL

LDR R2, =COUNT

LDR R3,=DEST

;FILL THE MEMORY WITH 0xAA

STO STR R1,[R0], #4

SUBS R2,R2,#4

BNE STO

;RESTORE THE REGISTER VALUES

LDR R0, =SOURCE

LDR R2, =COUNT

;MOVE THE SOURCE TO DEST

STO1 LDR R4,[R0],#4

STR R4, [R3],#4

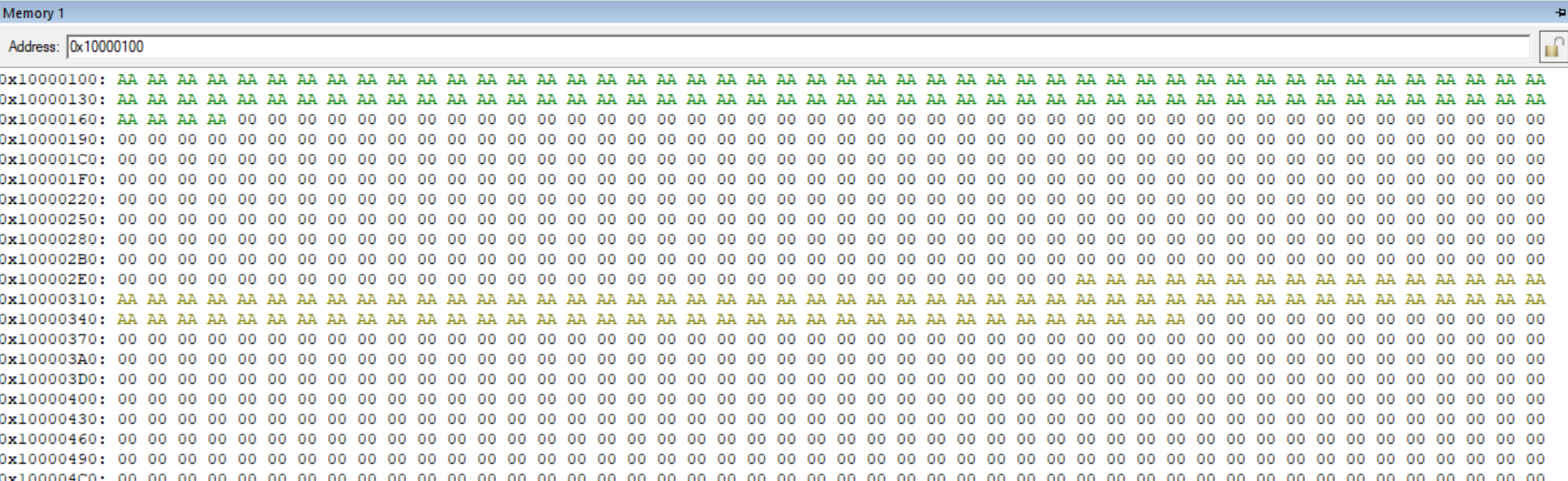
SUBS R2,R2,#4

BNE STO1

L B L

END

Output:



**Work for students:**

Complete the algorithm section for exercise 2.

**Conclusion:** Thus, an assembly program is written to fill the memory block with particular byte pattern and copy the block of memory to another location.

Project folder: mem\_op