EE2016 – LAB REPORT

EXPERIMENT - 3

ARM ASSEMBLY - COMPUTATIONS IN ARM

JAWHAR S

EE20B049

AIM OF THE EXPERIMENT:

- To learn the architecture of ARM processor
- To learn the basics of ARM instruction set, in particular the ARM instructions pertaining to computations
- To go through example programs
- To write assembly language programs for the given set of (computational) problems

QUESTIONS (FROM THE HANDOUT):

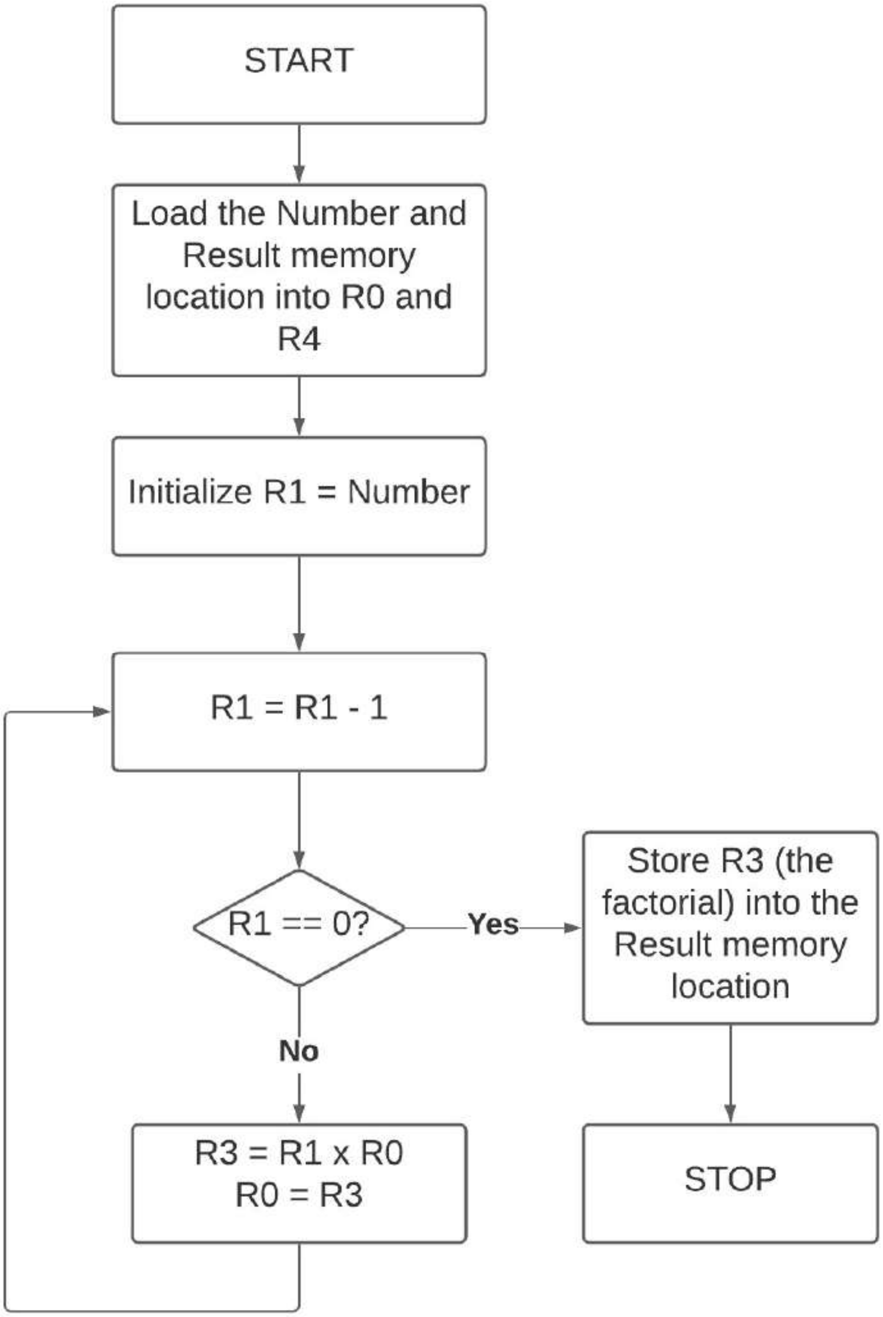
1. Compute the factorial of a given number using ARM processor through assembly programming

2. Combine the low four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword. The value at LIST goes into the most significant nibble of the result. Store the result in the 32-bit variable RESULT.

3. Given a 32-bit number, identify whether it is an even or odd. (The implementation should not involve division).

SOLUTIONS TO THE QUESTIONS:

1. (a) Flowchart/Logic Explanation:


START
Load the Number and Result memory location into R0 and R4
Initialize R1 = Number
R1 = R1 - 1
R1 == 0?
Yes
Store R3 (the factorial) into the Result memory location
STOP
No
R3 = R1 x R0
R0 = R3

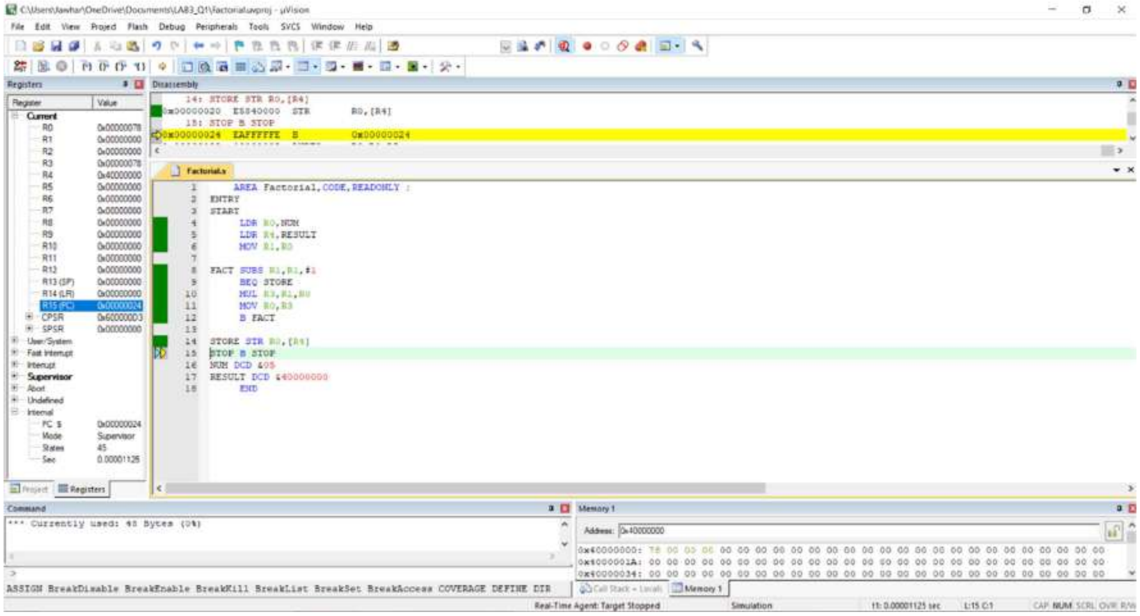
1.(b) Code:

```
        AREA Factorial,CODE,READONLY ;
ENTRY
START
    LDR R0,NUM
    LDR R4,RESULT
    MOV R1,R0

FACT SUBS R1,R1,#1
    BEQ STORE
    MUL R3,R1,R0
    MOV R0,R3
    B FACT

STORE STR R0,[R4]
STOP B STOP
NUM DCD &05
RESULT DCD &40000000
    END
```

The number taken in this program is 5. The factorial of the number (i.e., 120 which is 0x78) computed would be stored at location 0x40000000.



C:\Users\Jawhar\OneDrive\Documents\LAB3_Q1\Factorial.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Factorial.s
Project
Registers
Command
Memory 1
Call Stack + Locals
Memory 1

Register	Value
Current	
R0	0x00000078
R1	0x00000000
R2	0x00000000
R3	0x00000078
R4	0x40000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0x60000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000024
Mode	Supervisor
States	45
Sec	0.00001125

```
    14: STORE STR R0,[R4]
0x00000020  E5840000  STR        R0,[R4]
    15: STOP B STOP
0x00000024  EAFFFFFE  B          0x00000024
```

```
    AREA Factorial,CODE,READONLY ;
ENTRY
START
     LDR R0,NUM
     LDR R4,RESULT
     MOV R1,R0

FACT SUBS R1,R1,#1
     BEQ STORE
     MUL R3,R1,R0
     MOV R0,R3
     B FACT

STORE STR R0,[R4]
STOP B STOP
NUM DCD &05
RESULT DCD &40000000
     END
```

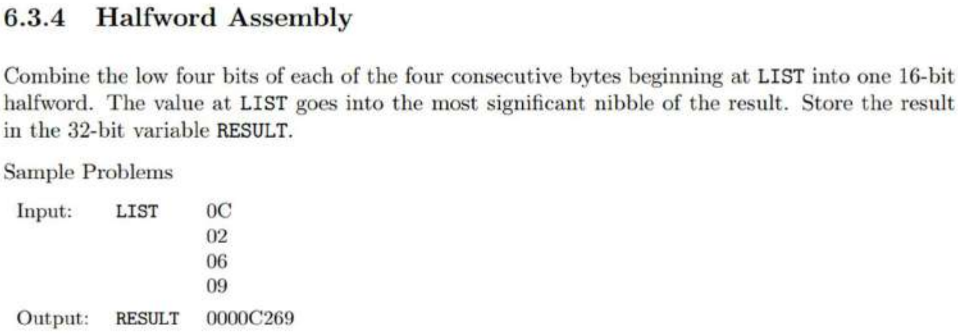
```
*** Currently used: 48 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

```
Address: 0x40000000
0x40000000: 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Real-Time Agent: Target Stopped   Simulation   t1: 0.00001125 sec   L:15 C:1   CAP NUM SCRL OVR R/W

2.(a) Flowchart/Logic Explanation:

The test case used for understanding the problem statement is given in the ARM Book by Welsh.



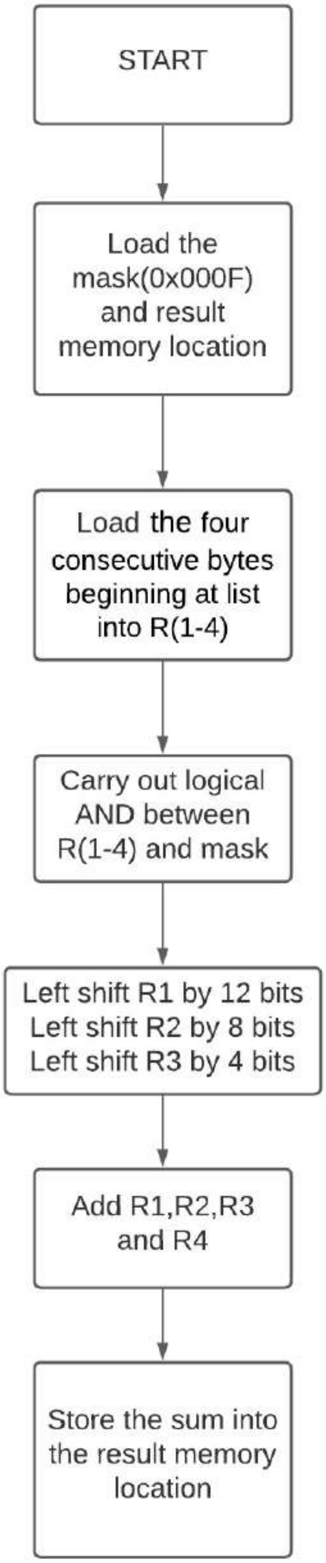
6.3.4 Halfword Assembly

Combine the low four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword. The value at LIST goes into the most significant nibble of the result. Store the result in the 32-bit variable RESULT.

Sample Problems

Input:	LIST	0C
		02
		06
		09
Output:	RESULT	0000C269

The code and logic used for solving the problem is by assuming the above Input-Output requirements.

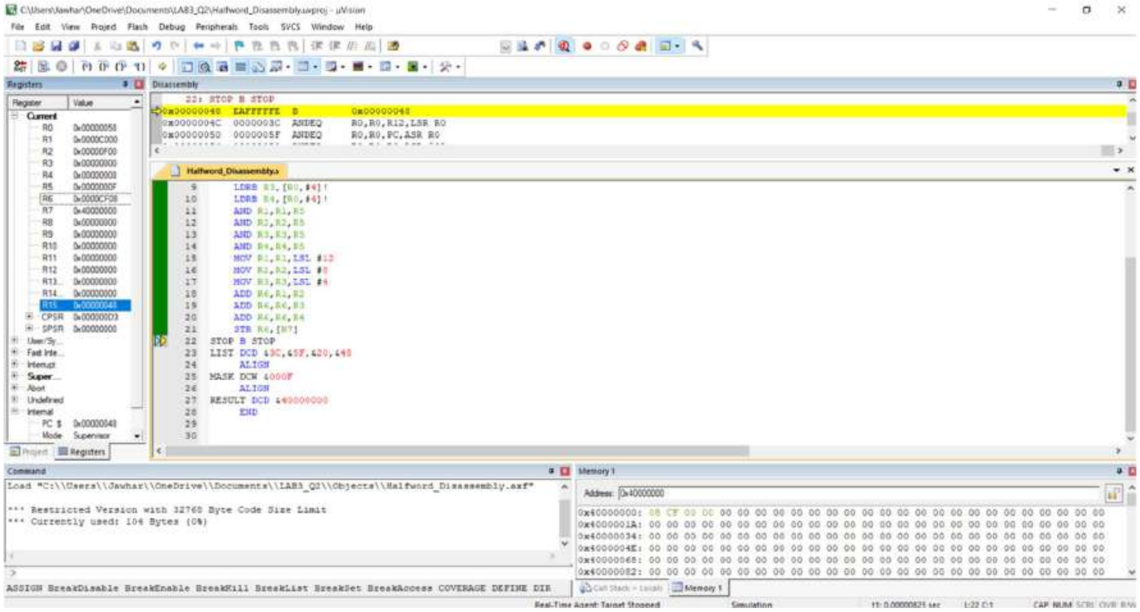

START
Load the mask(0x000F) and result memory location
Load the four consecutive bytes beginning at list into R(1-4)
Carry out logical AND between R(1-4) and mask
Left shift R1 by 12 bits
Left shift R2 by 8 bits
Left shift R3 by 4 bits
Add R1,R2,R3 and R4
Store the sum into the result memory location

R(1-4) stands for R1,R2,R3 and R4

2.(b) Code:

```
        AREA HALFWORDASSEMBLY,CODE,READONLY
ENTRY
START
        LDR R5,MASK
        LDR R7,RESULT
        LDR R0,=LIST
        LDRB R1,[R0]
        LDRB R2,[R0,#4]!
        LDRB R3,[R0,#4]!
        LDRB R4,[R0,#4]!
        AND R1,R1,R5
        AND R2,R2,R5
        AND R3,R3,R5
        AND R4,R4,R5
        MOV R1,R1,LSL #12
        MOV R2,R2,LSL #8
        MOV R3,R3,LSL #4
        ADD R6,R1,R2
        ADD R6,R6,R3
        ADD R6,R6,R4
        STR R6,[R7]
STOP B STOP
LIST DCD &3C,&5F,&20,&48
```

```
          ALIGN
MASK DCW &000F
     ALIGN
RESULT DCD &40000000
          END
```



C:\Users\Jawhar\OneDrive\Documents\LAB3_Q2\Halfword_Disassembly.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Halfword_Disassembly.s
Project
Registers
Command
Memory 1
Call Stack + Locals
Memory 1

```
Register   Value
Current
  R0       0x00000058
  R1       0x0000C000
  R2       0x00000F00
  R3       0x00000000
  R4       0x00000008
  R5       0x0000000F
  R6       0x0000CF08
  R7       0x40000000
  R8       0x00000000
  R9       0x00000000
  R10      0x00000000
  R11      0x00000000
  R12      0x00000000
  R13...   0x00000000
  R14...   0x00000000
  R15...   0x00000048
  CPSR     0x000000D3
  SPSR     0x00000000
User/Sy...
Fast Inte...
Interrupt
Super...
Abort
Undefined
Internal
  PC $     0x00000048
  Mode     Supervisor
```

```
    22: STOP B STOP
0x00000048  EAFFFFFE  B         0x00000048
0x0000004C  0000003C  ANDEQ     R0,R0,R12,LSR R0
0x00000050  0000005F  ANDEQ     R0,R0,PC,ASR R0
```

```
        LDRB R3,[R0,#4]!
        LDRB R4,[R0,#4]!
        AND R1,R1,R5
        AND R2,R2,R5
        AND R3,R3,R5
        AND R4,R4,R5
        MOV R1,R1,LSL #12
        MOV R2,R2,LSL #8
        MOV R3,R3,LSL #4
        ADD R6,R1,R2
        ADD R6,R6,R3
        ADD R6,R6,R4
        STR R6,[R7]
STOP B STOP
LIST DCD &3C,&5F,&20,&48
        ALIGN
MASK DCW &000F
        ALIGN
RESULT DCD &40000000
        END
```

```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB3_Q2\\Objects\\Halfword_Disassembly.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 104 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

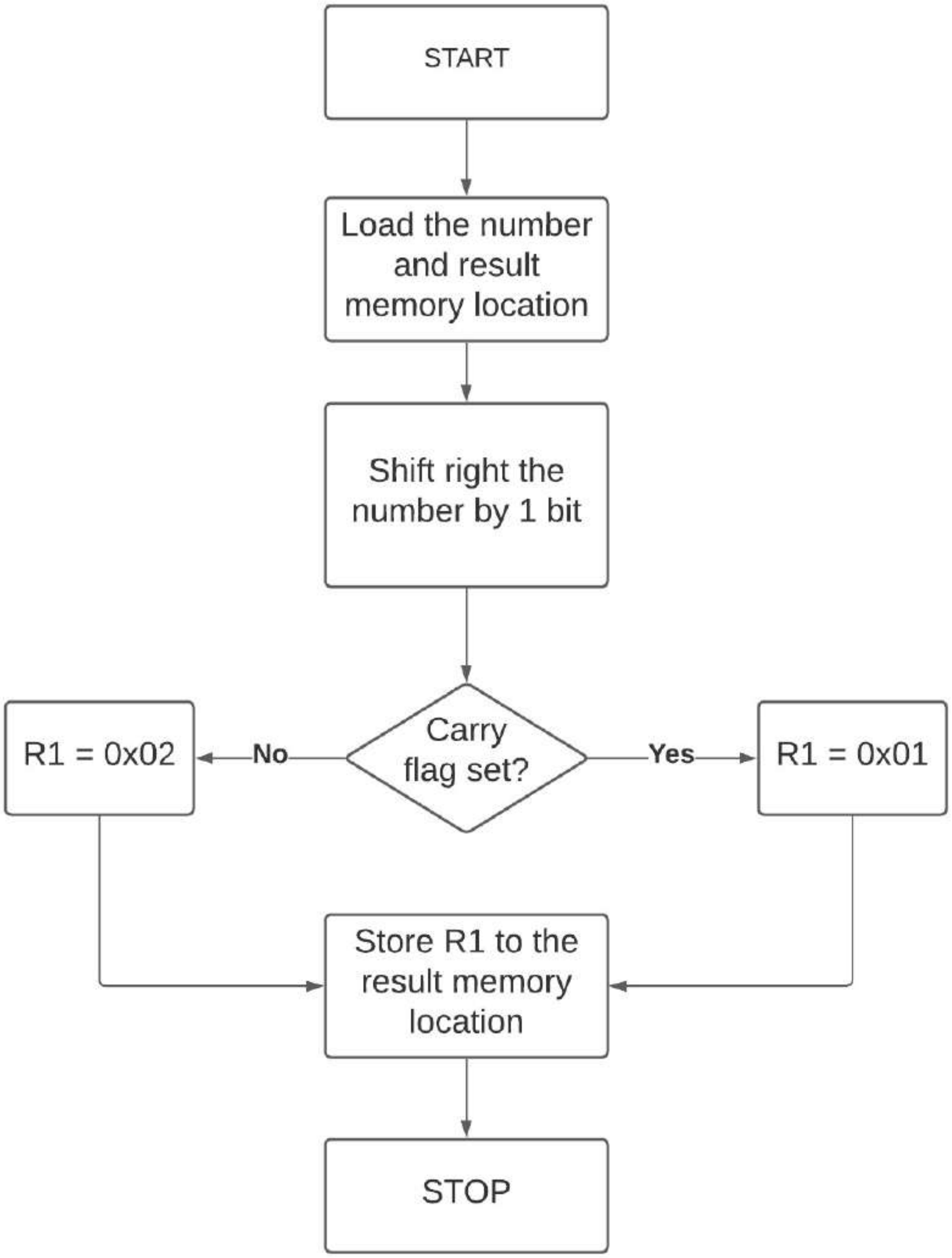
```
Address: 0x40000000
0x40000000: 08 CF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Real-Time Agent: Target Stopped    Simulation    t1: 0.00000825 sec    L:22 C:1    CAP NUM SCRL OVR R/W
```

The four consecutive bytes beginning at LIST taken in this program are 0x3C, 0x5F, 0x20 and 0x48. The output obtained by combining the low four bits of each of the four consecutive bytes (i.e., 0xCF08) would be stored at location 0x40000000.

3.(a) Flowchart/Logic Explanation:

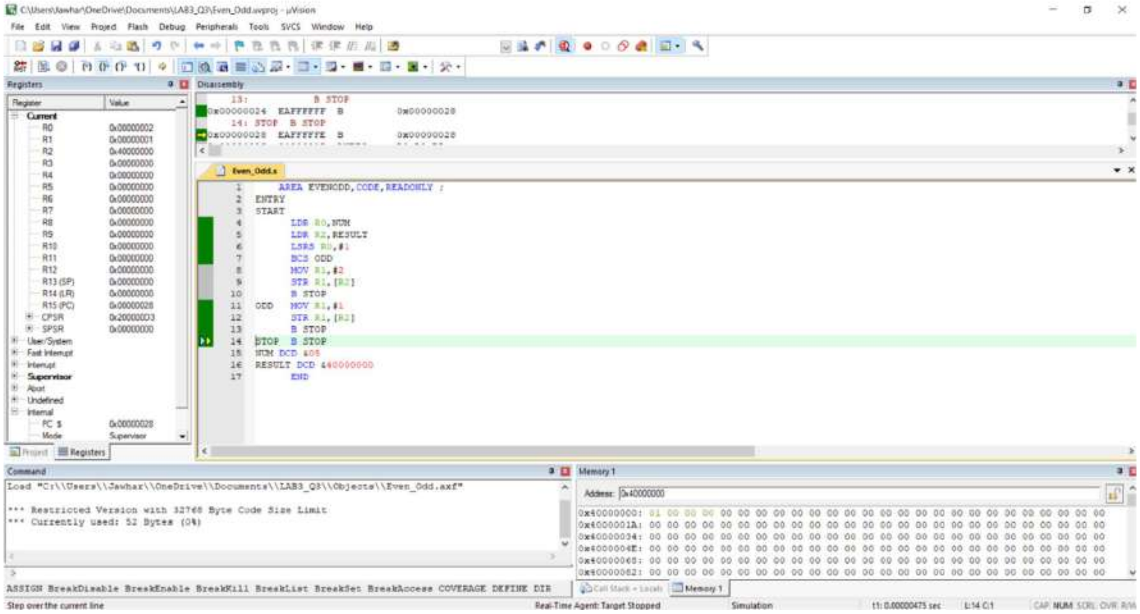
2 is loaded into R1 if the number is even and 1 is loaded into R1 if the number is odd.


START
Load the number and result memory location
Shift right the number by 1 bit
Carry flag set?
No
R1 = 0x02
Yes
R1 = 0x01
Store R1 to the result memory location
STOP

3.(b) Code:

```
         AREA EVENODD,CODE,READONLY ;
ENTRY
START
     LDR R0,NUM
          LDR R2,RESULT
          LSRS R0,#1
          BCS ODD
          MOV R1,#2
          STR R1,[R2]
          B STOP
ODD   MOV R1,#1
          STR R1,[R2]
          B STOP
STOP  B STOP
NUM DCD &05
RESULT DCD &40000000
          END
```

- When the number is odd:


C:\Users\Jawhar\OneDrive\Documents\LAB3_Q3\Even_Odd.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Even_Odd.s
Project
Registers
Command
Memory 1
Call Stack + Locals

```
Register        Value
Current
  R0            0x00000002
  R1            0x00000001
  R2            0x40000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x00000028
  CPSR          0x20000003
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC $          0x00000028
  Mode          Supervisor
```

```
    13:             B STOP
0x00000024  EAFFFFFF  B          0x00000028
    14: STOP  B STOP
0x00000028  EAFFFFFE  B          0x00000028
```

```
        AREA EVENODD,CODE,READONLY ;
ENTRY
START
        LDR R0,NUM
        LDR R2,RESULT
        LSRS R0,#1
        BCS ODD
        MOV R1,#2
        STR R1,[R2]
        B STOP
ODD     MOV R1,#1
        STR R1,[R2]
        B STOP
STOP  B STOP
NUM DCD &05
RESULT DCD &40000000
        END
```

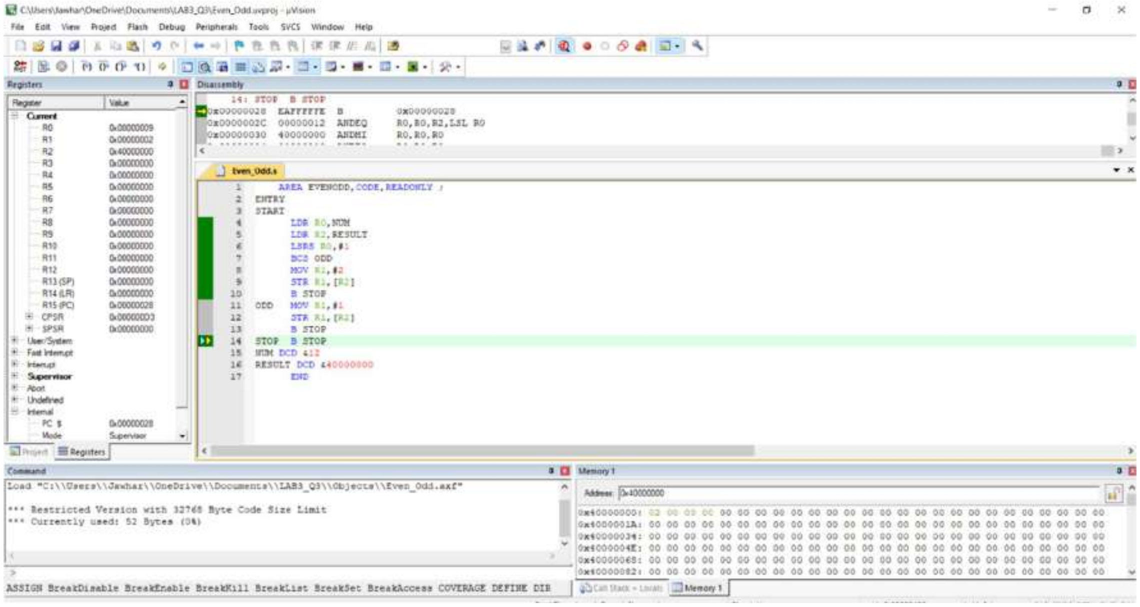
```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB3_Q3\\Objects\\Even_Odd.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 52 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

```
Address: 0x40000000
0x40000000: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Step over the current line    Real-Time Agent: Target Stopped    Simulation    t1: 0.00000475 sec    L:14 C:1    CAP NUM SCRL OVR R/W
```

- When the number is even:


C:\Users\Jawhar\OneDrive\Documents\LAB3_Q3\Even_Odd.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Even_Odd.s
Project
Registers
Command
Memory 1
Call Stack + Locals

```
Register        Value
Current
  R0            0x00000009
  R1            0x00000002
  R2            0x40000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x00000028
  CPSR          0x000000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC $          0x00000028
  Mode          Supervisor
```

```
    14: STOP  B STOP
0x00000028  EAFFFFFE  B          0x00000028
0x0000002C  00000012  ANDEQ      R0,R0,R2,LSL R0
0x00000030  40000000  ANDMI      R0,R0,R0
```

```
        AREA EVENODD,CODE,READONLY ;
ENTRY
START
        LDR R0,NUM
        LDR R2,RESULT
        LSRS R0,#1
        BCS ODD
        MOV R1,#2
        STR R1,[R2]
        B STOP
ODD     MOV R1,#1
        STR R1,[R2]
        B STOP
STOP  B STOP
NUM DCD &12
RESULT DCD &40000000
        END
```

```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB3_Q3\\Objects\\Even_Odd.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 52 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

```
Address: 0x40000000
0x40000000: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

INFERENCES/LEARNINGS FROM THE EXPERIMENT:

- Learnt about the ARM Architecture: various processor modes, flags and General-Purpose Registers (GPRs) in ARM.
- Learnt about the ARM ISA, in particular understood the various instructions used for computations in ARM.
- Installed Kiel software and configured it to use it as a simulator.
- Understood the example programs by debugging them in Kiel software.
- Learnt how to transfer the contents in memory to GPRs and vice-versa in ARM.
- Learnt about branching, arithmetic and logical instructions and used some of them to write the codes for the questions given in the handout.