



**Faculty of Engineering & Technology  
Electrical & Computer Engineering Department**

**Computer Organization And Microprocessor**

**Assembly Assignment**

---

**Prepared by:**

Mohammad Abu Shams                    1200549

**Instructor:** Dr. Ahmad Afaneh

Section: 3

Date: 22-6-2022

## The Code:

**proj2.s**

```

1 ; Mohammad Naeem Abu Shams.
2 ; 1200549.
3 ; ORGA Assignment.
4
5 AREA variables, DATA, READWRITE
6 SUM SPACE 3 ; Define an variable called SUM with size equal 3 byte.
7 SUM2 SPACE 3 ; Define a second variable called SUM2 with size equal 3 byte.
8 Result SPACE 10 ; Define a third variable called Result with size equal 10 byte.
9
10 AREA RESET, DATA, READONLY
11 EXPORT __Vectors
12
13 __Vectors DCD 0x20001000
14
15 DCD Reset_Handler
16
17 ALIGN
18
19 AREA mycode, CODE, READONLY
20 ENTRY
21 EXPORT Reset_Handler
22 Reset_Handler
23
24 MOV R0, #10 ; Store 10 decimal to the register R0.
25
26
27 LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
28 MOV R3, #0 ; Store 0 decimal to the register R3.
29
30 Array DCB 31, 74, 46, 100, 10, 160, 128, 245, 16, 55 ; Define an array with size equal 10 byte.
31
32
33 Loop : Loop to calculate the sum of numbers in the array;
34 LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
35 ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.

```

**Build Output**

```

Program Size: Code=144 RO-data=8 RW-data=16 ZI-data=0
".\Objects\proj2.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00

```

**proj2.s**

```

35 ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.
36 ADD R1, #1 ; Store 1 decimal to the register R1.
37
38 SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
39 BNE Loop ; Keep looping if Z = 0;
40
41 LDR R1, =SUM ; To bring the address of the SUM and stored it into a register R1.
42 STR R3, [R1] ; Stroe the value of R3 to SUM.
43
44 LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
45 MOV R0, #10 ; Store 10 decimal to the register R0.
46 MOV R4, #0 ;Store 0 decimal to the register R4.
47
48 loop2 ; Loop to find the sum of even numbers in the array.
49
50 LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
51
52 MOVS R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
53 BNE Skip; If Z=0, branch ( Skip the add ).
54 ADD R4, R4, R2 ; Add the number of register R2 to the register R4, and the result in the register R4.
55 Skip
56 ADD R1, #1 ; Increment the register R1 by 1.
57 SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
58
59 BNE loop2 ; Keep looping if Z = 0.
60
61 LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
62 STR R4, [R1] ; Stroe the value of R4 to SUM2.
63
64 LDR R1, =Array; To bring the address of the array and stored it into a register R1.
65 LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
66 MOV R0, #10 ; Store 10 decimal to the register R0.
67
68 Loop3 ; Loop to find the largest power of two for each numbers of array.
69

```

**Build Output**

```

Program Size: Code=144 RO-data=8 RW-data=16 ZI-data=0
".\Objects\proj2.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00

```

Project

Project: proj2

Target 1

Source Group 1

proj2.s

```

67
68 Loop3 ; Loop to find the largest power of two for each numbers of array.
69
70 LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
71 BL Pow ; Branch and link.
72 STRB R7, [R4] ; Store one byte to the result array.
73 ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
74
75 ADD R1, #1 ; Increment the register R1 by 1.
76 SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
77 BNE Loop3 ; Keep looping if Z = 0.
78
79 Pow ; Function to find the largest power of two.
80 MOV R5, #-1 ; Store -1 decimal to the register R5.
81 Loop4 : To find the number of zeros in the right.
82 ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
83 ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
84 MOV R2, R2, LSR #1 ; Shift right register R2 by 1 bit.
85 BEQ Loop4 ; Branch if Z=1;
86
87 MOV R7, #1 ; Store 1 decimal to the register R7.
88 MOV R7, R7, LSL R5 ; Logical shift register by R5.
89
90 BX LR ; Return.
91
92
93
94 END ; The end of programme.
95
96
97
98
99
100

```

Project Templates

Build Output

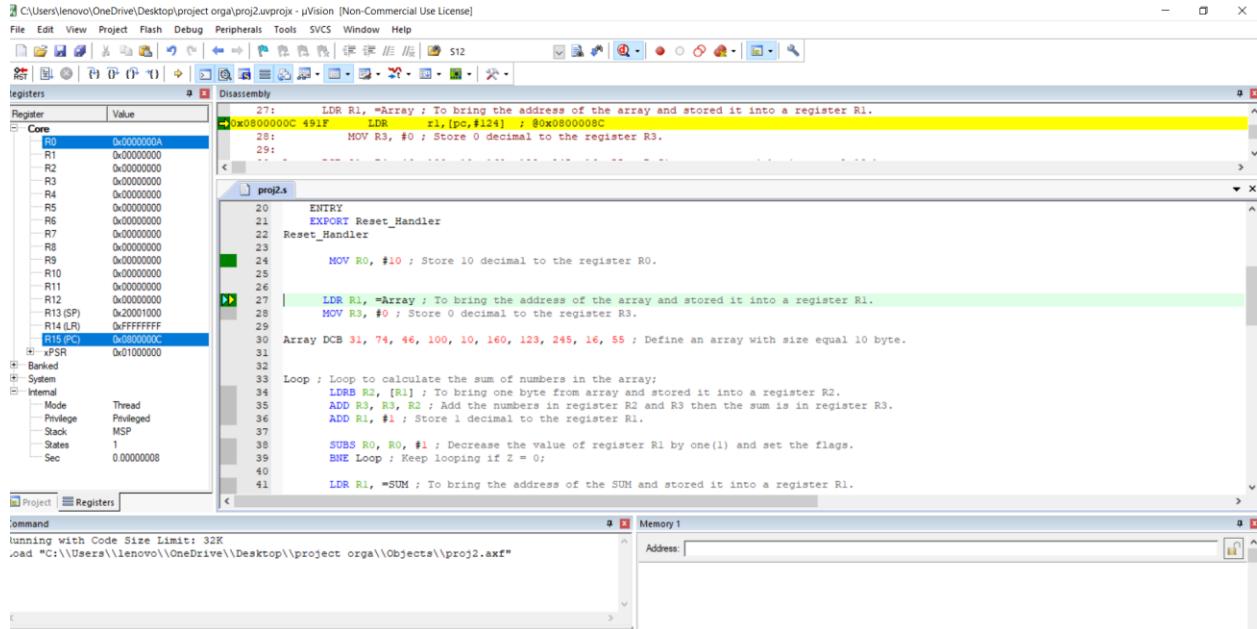
Program Size: Code=144 RO-data=8 RW-data=16 ZI-data=0  
 ".\Objects\proj2.axf" - 0 Error(s), 0 Warning(s).

Build Time Elapsed: 00:00:00

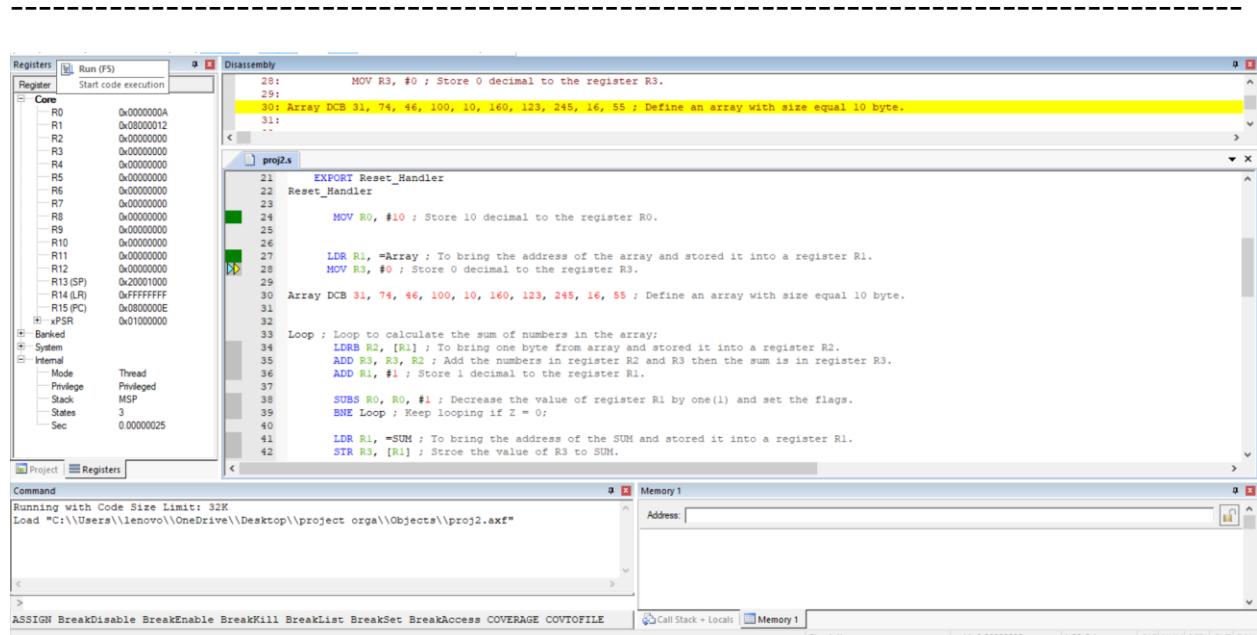
Simulation | L84 C40 | CAP NUM SCR OVR R/W

a) Declare an array of at least 10 8-bit unsigned integer numbers in the memory with initial values.

31, 74, 46, 100, 10, 160, 123, 245, 16, 55



Store 10 decimal to the register R0.



Store the address of the array into a register R1.

The screenshot shows a debugger interface with several windows:

- Registers**: Shows CPU register values. Notable values include R0 (0x0000000A), R1 (0x00000012), R2 (0x00000000), R3 (0x00000000), R4 (0x00000000), R5 (0x00000000), R6 (0x00000000), R7 (0x00000000), R8 (0x00000000), R9 (0x00000000), R10 (0x00000000), R11 (0x00000000), R12 (0x00000000), R13 (SP) (0x20001000), R14 (LR) (0xFFFFFFF), R15 (PC) (0x0800001C), and xPSR (0x01000000).
- Disassembly**: Shows assembly code for the project. A specific instruction at address 0x0800001C (LDRB R2, [R1]) is highlighted in yellow.
- proj2.s**: Shows the assembly source code for the project. It includes instructions to load array addresses, add array elements, calculate the sum of even numbers, and store the result.
- Memory**: Shows a memory dump starting at address 0x08000012. The first 10 bytes of the array are underlined in blue: **1F 4A 2E 64 0A A0 7B F5 10**.

The first 10 number in this address from left to right in the first line which underlined with blue color it's the array values.

---

**The first part is done**



b) Find the sum of all elements of the array and store it in the memory, e.g. variable SUM.

```

Registers
Register Value
Core
R0 0x0000000A
R1 0x00000012
R2 0x0000001F
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20001000
R14 (LR) 0xFFFFFFF
R15 (PC) 0x0000001E
xPSR 0x10000000

Disassembly
34: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
35: ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.
36: ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.
37: ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.

proj2.s
29 MOV R3, $0 ; Store 0 decimal to the register R3.
30 Array DCB 31, 74, 46, 100, 10, 160, 123, 245, 16, 55 ; Define an array with size equal 10 byte.
31
32
33 Loop : Loop to calculate the sum of numbers in the array;
34 LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
35 ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.
36 ADD R1, #1 ; Store 1 decimal to the register R1.
37
38 SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
39 BNE Loop ; Keep looping if Z = 0;
40
41 LDR R1, =SUM ; To bring the address of the SUM and stored it into a register R1.
42 STR R3, [R1] ; Stroe the value of R3 to SUM.
43
44 LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
45 MOV R0, #10 ; Store 10 decimal to the register R0.
46 MOV R4, $0 ;Store 0 decimal to the register R4.
47
48 loop2 : Loop to find the sum of even numbers in the array.
49
50

Memory 1
Address: 
```

To bring one byte from array and stored it into a register R2.

```

Registers
Register Value
Core
R0 0x0000000A
R1 0x00000012
R2 0x0000001F
R3 0x0000001F
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20001000
R14 (LR) 0xFFFFFFF
R15 (PC) 0x00000020
xPSR 0x10000000

Disassembly
36: ADD R1, #1 ; Store 1 decimal to the register R1.
37:
38: ADD R1, #1 ; Store 1 decimal to the register R1.
39: SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
40:
41: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
42: ADD R3, R3, R2 ; Add the numbers in register R2 and R3 then the sum is in register R3.
43: ADD R1, #1 ; Store 1 decimal to the register R1.
44:
45: SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
46: BNE Loop ; Keep looping if Z = 0;
47:
48: LDR R1, =SUM ; To bring the address of the SUM and stored it into a register R1.
49: STR R3, [R1] ; Stroe the value of R3 to SUM.
50:
51: LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
52: MOV R0, #10 ; Store 10 decimal to the register R0.
53: MOV R4, $0 ;Store 0 decimal to the register R4.
54:
55: loop2 : Loop to find the sum of even numbers in the array.
56:
57: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.

Memory 1
Address: 
```

Add the numbers value in the register R2 with register R3.

The screenshot shows the IAR Embedded Workbench interface with the following components:

- Registers** window: Shows the state of various registers. The R1 register is highlighted in blue, indicating it is the current selection.
- Disassembly** window: Displays the assembly code for the program. The instruction at address 0x00000024 is highlighted in yellow, and its assembly representation is shown: SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
- Code View** window: Shows the C/C++ source code for the project. It includes comments explaining the purpose of each section of code, such as calculating the sum of numbers in an array and finding the sum of even numbers.
- Memory Dump** window: Allows viewing memory contents at specific addresses.
- Command Line** window: Displays the command being run: "arm-none-eabi-gdb -ex r -ex file proj2.axf".

The register R1 indicate to the next index.

The screenshot shows the QEMU debugger interface with the following details:

- Registers:** A table showing CPU registers (R0-R14, CPSR) with their current values.
- Disassembly:** The assembly code for the program, highlighting specific instructions like BNE and LDR.
- proj2.s:** The source code file being debugged, showing the assembly generated from the C code.
- Memory Dump:** A window showing memory contents at address 0x00000000.
- Command Line:** The terminal window at the bottom showing the build command and its output.

Decrease the value of register R1 by one.

The screenshot shows a debugger interface with the following components:

- Registers** window: Shows the state of various registers. The R15 (PC) register is highlighted with a blue selection bar, containing the value 0x0000001C.
- Disassembly** window: Displays the assembly code for the program. The current instruction at address 0x0000001C is highlighted in yellow. The assembly code includes instructions like LDRB, ADD, and SUBS, which are part of a loop to calculate the sum of an array.
- Memory 1** window: Shows a memory dump starting at address 0x00000012. The dump displays binary data and its corresponding hex values.
- Command** window: Shows the command line with "Running with Code Size Limit: 32K" and "Load C:\Users\lenovo\Desktop\project orga\Objects\proj2.axf".

The loop repeated 10 times, and we adds the 10 numbers of the array.

---

This screenshot is similar to the one above, but the R15 (PC) register now contains the value 0x0000002A, indicating the program has moved to the next instruction in the loop.

The sum of array's values which it's stored to register R3 is **35C**

Stored the address of the SUM into a register R1.

The screenshot shows the QEMU debugger interface with the following panes:

- Registers**: Shows the CPU registers (R0-R13, SP, LR, PC, xPSR) and their values.
- Disassembly**: Displays the assembly code for the program, highlighting specific instructions like LDR R1, =Array; MOV R0, #10; SUBS R0, R0, #1; etc.
- Registers**: Shows the CPU registers again, with the PC register highlighted.
- Command**: Displays the command line: "Running with Code Size Limit: 32K" and "Load \"C:\\Users\\lenovo\\OneDrive\\Desktop\\project orga\\Objects\\proj2.axf\"".
- Memory**: Shows a memory dump starting at address 0x20000000, displaying hex values for memory locations 0x20000000 through 0x2000000F.

We stored the value of register R3 to SUM.

$$34+74+46+100+10+160+123+245+16+55=860$$

860 in decimal = 35C in hexadecimal which appear above.

**Underlined with blue color.**

## The second part is done

c) find the sum of the even numbers in this array and store it in the memory, e.g. variable EVEN

```

Registers
Disassembly
proj2.s
        LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
        MOV R0, #10 ; Store 10 decimal to the register R0.
        MOV R4, #0x0000000A
        SUBS R0, R0, #1 ; Decrease the value of register R0 by one(1) and set the flags.
        BNE loop1 ; Keep looping if Z = 0;
        LDR R1, =SUM ; To bring the address of the SUM and stored it into a register R1.
        STR R3, [R1] ; Stroe the value of R3 to SUM.
        LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
        MOV R0, #10 ; Store 10 decimal to the register R0.
        MOV R4, #0 ; Store 0 decimal to the register R4.
loop2 : Loop to find the sum of even numbers in the array.
        LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
        MOVS R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
        BNE Skip; If Z=0, branch ( Skip the add).
        ADD R4, R4, R2; Add the number of register R2 to the register R4, and the result in the register R4.
        ADD R1, #1 ; Increment the register R1 by 1.
        SUBS R0, R0, #1 ; Decrease the value of register R0 by one(1) and set the flags.
        BNE loop2 ; Keep looping if Z = 0.
Skip
        ADD R1, #1 ; Increment the register R1 by 1.
        SUBS R0, R0, #1 ; Decrease the value of register R0 by one(1) and set the flags.
        BNE loop2 ; Keep looping if Z = 0.

Memory 1
Address: [ ]
```

Store the address of the array in a register R1.

```

Registers
Disassembly
proj2.s
        MOVS R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
        BNE Skip; If Z=0, branch ( Skip the add).
        MOV R0, #10 ; Store 10 decimal to the register R0.
        MOV R4, #0 ; Store 0 decimal to the register R4.
loop2 : Loop to find the sum of even numbers in the array.
        LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
        MOVS R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
        BNE Skip; If Z=0, branch ( Skip the add).
        ADD R4, R4, R2; Add the number of register R2 to the register R4, and the result in the register R4.
        ADD R1, #1 ; Increment the register R1 by 1.
        SUBS R0, R0, #1 ; Decrease the value of register R0 by one(1) and set the flags.
        BNE loop2 ; Keep looping if Z = 0.
Skip
        ADD R1, #1 ; Increment the register R1 by 1.
        SUBS R0, R0, #1 ; Decrease the value of register R0 by one(1) and set the flags.
        BNE loop2 ; Keep looping if Z = 0.

Memory 1
Address: [ ]
```

To bring one byte from array and stored it into a register R2.

The screenshot shows the IAR Embedded Workbench interface with several windows open:

- Registers**: Shows the state of various registers (R0-R15, PC, RSR) with their current values.
- Disassembly**: Displays assembly code with comments explaining the logic. The highlighted line is `BNE 0x0800003A D100 BNE 0x0800003E`.
- Code**: Shows the C source code for the project, including comments explaining the assembly instructions.
- Command**: A terminal window at the bottom left.
- Memory**: A memory dump window at the bottom right.

Shift left register R2 by 31 bit, then stored it into a register R3, then check if the number don't equal zero, and skip if it doesn't equal zero.

The screenshot shows the Keil MDK-ARM interface with three main windows:

- Registers** pane: Shows the state of various registers. The R1 register is highlighted in blue, indicating it is the current selection.
- Disassembly** pane: Displays assembly code for the current program. The instruction at address 0x08000042 is highlighted in yellow, and the following instruction at address 0x08000043 is highlighted in green. The assembly code includes comments explaining the purpose of each instruction.
- Stack** pane: Shows the stack dump for the current task. It lists memory addresses from 0x00000000 to 0x0000000F, their current values, and the stack pointer value.

The register R1 indicate to the next index.

Registers

| Register | Value      |
|----------|------------|
| R0       | 0x00000009 |
| R1       | 0x00000013 |
| R2       | 0x0000001F |
| R3       | 0x00000000 |
| R4       | 0x00000000 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0xFFFFFFFF |
| R15 (PC) | 0x00000044 |
| xPSR     | 0x21000000 |

Disassembly

```

59:     BNE loop2 ; Keep looping if Z = 0.
60:
61:     LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
62:
63:     .....
64:
65:     MOV R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
66:     BNE Skip; If Z=0, branch ( Skip the add ).
67:     ADD R4, R4, R2; Add the number of register R2 to the register R4, and the result in the register R4.
68:     Skip
69:     ADD R1, #1 ; Increment the register R1 by 1.
70:     SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
71:
72:     BNE loop2 ; Keep looping if Z = 0.
73:
74:     LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
75:     STR R4, [R1] ; Stroe the value of R4 to SUM2.
76:
77:     .....
78:     LDR R1, =Array; To bring the address of the array and stored it into a register R1.
79:     LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
80:     MOV R0, #10 ; Store 10 decimal to the register R0.
81:
82:     Loop3 ; Loop to find the largest power of two for each numbers of array.
83:
84:     LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
85:     BL Pow ; Branch and link.
86:     STRB R7, [R4] ; Store one byte to the result array.
87:     ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.

```

Project Registers

Decrease the value of register R1 by one.

---

Registers

| Register | Value      |
|----------|------------|
| R0       | 0x00000009 |
| R1       | 0x00000013 |
| R2       | 0x0000001F |
| R3       | 0x00000000 |
| R4       | 0x00000000 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0xFFFFFFFF |
| R15 (PC) | 0x00000036 |
| xPSR     | 0x21000000 |

Disassembly

```

50:     LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
51:
52:     MOV R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
53:     BNE Skip; If Z=0, branch ( Skip the add ).
54:
55:     LDR R1, =Array ; To bring the address of the array and stored it into a register R1.
56:     MOV R0, #10 ; Store 10 decimal to the register R0.
57:     MOV R4, #0 ;Store 0 decimal to the register R4.
58:
59:     loop2 ; Loop to find the sum of even numbers in the array.
60:
61:     LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
62:
63:     MOV R3, R2, LSL #31 ; Shift left register R2 by 31 bit, then set the flags. If Z=0, then the number is odd, else if the Z=1, then the number is even.
64:     BNE Skip; If Z=0, branch ( Skip the add ).
65:     ADD R4, R4, R2; Add the number of register R2 to the register R4, and the result in the register R4.
66:     Skip
67:     ADD R1, #1 ; Increment the register R1 by 1.
68:     SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
69:
70:     BNE loop2 ; Keep looping if Z = 0.
71:
72:     LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
73:     STR R4, [R1] ; Stroe the value of R4 to SUM2.
74:
75:     .....
76:     LDR R1, =Array; To bring the address of the array and stored it into a register R1.

```

Project Registers

Command Memory

Repeated the loop 10 times.

The screenshot shows the QEMU debugger interface. The Registers window on the left displays various CPU registers with their addresses and values. The Disassembly window in the center shows assembly code for project2.s, with specific instructions highlighted in blue. The Memory 1 window at the bottom right shows a dump of memory starting at address 0x20000003.

```

Registers
Disassembly
proj2.s
Memory 1

```

**Registers Window:**

| Register | Value      |
|----------|------------|
| R0       | 0x00000000 |
| R1       | 0x00000003 |
| R2       | 0x00000037 |
| R3       | 0x00000000 |
| R4       | 0x00000196 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0xFFFFFFF  |
| R15 (PC) | 0x080000AA |
| vPSR     | 0x10000000 |

**Disassembly Window:**

```

64:    LDR R1, =Array; To bring the address of the array and stored it into a register R1.
65:    LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
66:    LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
67:    LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
68:    MOV R0, #10 ; Store 10 decimal to the register R0.

Loop3 : Loop to find the largest power of two for each numbers of array.
69:    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
70:    BNE loop2 ; Keep looping if Z = 0.

LDRB R2, [R1] ;To bring one byte from array and stored it into a register R2.
71:    BL Pow ; Branch and link.
72:    STRB R7, [R4] ; Store one byte to the result array.
73:    ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.

ADD R1, #1 ; Increment the register R1 by 1.
75:    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
76:    BNE Loop3 ; Keep looping if Z = 0.

Loop3 : Loop to find the largest power of two for each numbers of array.
77:    ADD R1, #1 ; Increment the register R1 by 1.
78:    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
79:    BNE Loop3 ; Keep looping if Z = 0.

Pow : Function to find the largest power of two.
80:    MOV R5, #-1 ; Store -1 decimal to the register R5.

```

**Memory 1 Window:**

| Address    | Value       |
|------------|-------------|
| 0x20000003 | 06 01 00 00 |
| 0x20000004 | 00 00 00 00 |
| 0x20000037 | 00 00 00 00 |
| 0x20000051 | 00 00 00 00 |
| 0x2000006B | 00 00 00 00 |
| 0x20000085 | 00 00 00 00 |

Stored the sum of even numbers in a SUM2

Even numbers:  $74+46+100+10+160+16= 406$  in decimal which equal 196 in hexadecimal and this appeared above, underlined with blue color.

---

The screenshot shows the QEMU debugger interface. The Registers window on the left displays various CPU registers with their addresses and values. The Disassembly window in the center shows assembly code for project2.s, with specific instructions highlighted in blue. The Memory 1 window at the bottom right shows a dump of memory starting at address 0x20000003.

```

Registers
Disassembly
proj2.s
Memory 1

```

**Registers Window:**

| Register | Value      |
|----------|------------|
| R0       | 0x00000000 |
| R1       | 0x00000012 |
| R2       | 0x00000037 |
| R3       | 0x00000000 |
| R4       | 0x20000006 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0xFFFFFFF  |
| R15 (PC) | 0x080000AE |
| vPSR     | 0x10000000 |

**Disassembly Window:**

```

66:    MOV R0, #10 ; Store 10 decimal to the register R0.
67:    LDR R1, =Array; To bring the address of the array and stored it into a register R1.
68:    LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
69:    LDR R1, =SUM2 ; To bring the address of the SUM2 and stored it into a register R1.
70:    LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.
71:    MOV R0, #10 ; Store 10 decimal to the register R0.

Loop3 : Loop to find the largest power of two for each numbers of array.
72:    BNE loop2 ; Keep looping if Z = 0.

LDRB R2, [R1] ;To bring one byte from array and stored it into a register R2.
73:    BL Pow ; Branch and link.
74:    STRB R7, [R4] ; Store one byte to the result array.
75:    ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.

ADD R1, #1 ; Increment the register R1 by 1.
76:    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
77:    BNE Loop3 ; Keep looping if Z = 0.

Loop3 : Loop to find the largest power of two for each numbers of array.
78:    ADD R1, #1 ; Increment the register R1 by 1.
79:    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
80:    BNE Loop3 ; Keep looping if Z = 0.

Pow : Function to find the largest power of two.
81:    MOV R5, #-1 ; Store -1 decimal to the register R5.

```

**Memory 1 Window:**

| Address    | Value       |
|------------|-------------|
| 0x20000003 | 06 01 00 00 |
| 0x20000004 | 00 00 00 00 |
| 0x20000037 | 00 00 00 00 |
| 0x20000051 | 00 00 00 00 |
| 0x2000006B | 00 00 00 00 |
| 0x20000085 | 00 00 00 00 |

Stored the address of the array into a register R1, and stored the address of the result into a register R4.

**The third Part is done**



d) Find the largest power of 2 divisor that divides into a number exactly for each element in the array and store it in another array in the memory. You have to use a procedure (function), POW2, which takes an integer as an input parameter and return its largest power of 2. For example, POW(52) would return 4, where POW(56) would return 8, and so on.

**Hint:** You can find the largest power of 2 dividing into a number exactly by finding the rightmost bit of the number. For example,  $(52)_{10} = (110100)_2$  has its rightmost bit in the 4's place, so the largest power of 2 divisor is 4;  $(56)_{10} = (111000)_2$  has the rightmost bit in the 8's place, so its largest power of 2 divisor is 8

The screenshot shows a debugger interface with several panes:

- Registers** pane: Shows the state of various registers. The R2 register is highlighted in blue.
- Disassembly** pane: Displays assembly code with comments explaining its purpose. The current instruction at address 0x08000054 is highlighted in yellow.
- Assembly** pane: Shows the assembly code for the project, with labels like Loop3 and Pow3.
- Command** and **Memory** panes: Located at the bottom of the interface.

**Registers** pane content:

| Register        | Value             |
|-----------------|-------------------|
| R0              | 0x000000A         |
| R1              | 0x08000012        |
| <b>R2</b>       | <b>0x0000001F</b> |
| R3              | 0x00000000        |
| R4              | 0x20000006        |
| R5              | 0x00000000        |
| R6              | 0x00000000        |
| R7              | 0x00000000        |
| R8              | 0x00000000        |
| R9              | 0x00000000        |
| R10             | 0x00000000        |
| R11             | 0x00000000        |
| R12             | 0x00000000        |
| R13 (SP)        | 0x20001000        |
| R14 (LR)        | 0xFFFFFFF         |
| <b>R15 (PC)</b> | <b>0x08000054</b> |
| xPSR            | 0x10000000        |

**Disassembly** pane content (selected instruction highlighted in yellow):

```
70: LDRB R2, [R1] ;To bring one byte from array and stored it into a register R2.  
    0x08000052 780A LDRB r2,[r1, #0x00]  
71: BL Pow ; Branch and link.  
    0x08000054 F000F807 BL.W 0x08000066
```

**Assembly** pane content:

```
proj2.s  
64: LDR R1, =Array; To bring the address of the array and stored it into a register R1.  
65: LDR R4, =Result ; To bring the address of the Result and stored it into a register R4.  
66: MOV R0, #10 ; Store 10 decimal to the register R0.  
67:  
68: Loop3 ; Loop to find the largest power of two for each numbers of array.  
69:  
70: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.  
71: BL Pow ; Branch and link.  
72: STRB R7, [R4] ; Store one byte to the result array.  
73: ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.  
74:  
75: ADD R1, #1 ; Increment the register R1 by 1.  
76: SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.  
77: BNE Loop3 ; Keep looping if Z = 0.  
78:  
79: Pow ; Function to find the largest power of two.  
80: MOV R5, #-1 ; Store -1 decimal to the register R5.  
81: Loop4 ; To find the number of zeros in the right.  
82: ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.  
83: ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.  
84: MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.  
85: BEQ Loop4 ; Branch if Z=1;
```

To bring one byte from array and stored it into a register R2.

## Branch and link.

The screenshot shows a debugger interface with several windows:

- Registers**: Shows the state of various registers. R5 is highlighted and has a value of **0xFFFFFFFF**.
- Disassembly**: Shows the assembly code for the program. A specific instruction at address **0x0800006A** is highlighted in yellow: **ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.**
- proj2.s**: The source code file, showing the assembly language implementation of the program.
- Memory**: A window showing memory dump or registers.

Stored -1 decimal to the register R5.

---

The screenshot shows a debugger interface with several windows:

- Registers**: Shows the state of various registers. R6 is highlighted and has a value of **0x00000001**.
- Disassembly**: Shows the assembly code for the program. A specific instruction at address **0x08000072** is highlighted in yellow: **MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.**
- proj2.s**: The source code file, showing the assembly language implementation of the program.
- Memory**: A window showing memory dump or registers.

And between the register R2 and 1.

The screenshot shows a debugger interface with two main panes. The left pane is titled 'Registers' and lists the following register values:

| Register        | Value             |
|-----------------|-------------------|
| R0              | 0x0000000A        |
| R1              | 0x08000012        |
| <b>R2</b>       | <b>0x0000000F</b> |
| R3              | 0x00000000        |
| R4              | 0x20000006        |
| R5              | 0x00000000        |
| R6              | 0x00000001        |
| R7              | 0x00000000        |
| R8              | 0x00000000        |
| R9              | 0x00000000        |
| R10             | 0x00000000        |
| R11             | 0x00000000        |
| R12             | 0x00000000        |
| R13 (SP)        | 0x20001000        |
| R14 (LR)        | 0x08000059        |
| <b>R15 (PC)</b> | <b>0x08000076</b> |
| xPSR            | 0x21000000        |

The right pane is titled 'Disassembly' and shows the assembly code for 'proj2.s':

```

    85: BEQ Loop4 ; Branch if Z=1;
    86:
    87: 0x08000076 D0F8 BEQ    0x0800006A
    87: MOV R7, #1 ; Store 1 decimal to the register R7.

    78      Pow : Function to find the largest power of two.
    79      MOV R5, #-1 ; Store -1 decimal to the register R5.
    80      Loop4 : To find the number of zeros in the right.
    81          ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
    82          ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
    83          MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
    84          BEQ Loop4 ; Branch if Z=1;

    87      MOV R7, #1 ; Store 1 decimal to the register R7.
    88      MOV R7, R7, LSL R5 ; Logical shift register by R5.

    89      BX LR ; Return.

    90      END : The end of programme.

```

Shift right register R2 by 1 bit. ( left in the code is a mistake in writing )

And repeated the loop to become the first index equal 1.

---

The screenshot shows a debugger interface with two main panes. The left pane is titled 'Registers' and lists the following register values:

| Register        | Value             |
|-----------------|-------------------|
| R0              | 0x0000000A        |
| R1              | 0x08000012        |
| R2              | 0x0000000F        |
| R3              | 0x00000000        |
| R4              | 0x20000006        |
| R5              | 0x00000000        |
| R6              | 0x00000001        |
| R7              | 0x00000000        |
| R8              | 0x00000000        |
| R9              | 0x00000000        |
| R10             | 0x00000000        |
| R11             | 0x00000000        |
| R12             | 0x00000000        |
| R13 (SP)        | 0x20001000        |
| R14 (LR)        | 0x08000059        |
| <b>R15 (PC)</b> | <b>0x08000080</b> |
| xPSR            | 0x21000000        |

The right pane is titled 'Disassembly' and shows the assembly code for 'proj2.s':

```

    90: BX LR ; Return.

    0x08000080 4770 BX lr
    0x08000082 0000 DCW 0x0000
    0x08000084 0000 DCW 0x0000
    .... .... .... ....

    78      Pow : Function to find the largest power of two.
    79      MOV R5, #-1 ; Store -1 decimal to the register R5.
    80      Loop4 : To find the number of zeros in the right.
    81          ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
    82          ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
    83          MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
    84          BEQ Loop4 ; Branch if Z=1;

    87      MOV R7, #1 ; Store 1 decimal to the register R7.
    88      MOV R7, R7, LSL R5 ; Logical shift register by R5.

    89      BX LR ; Return.

    90      END : The end of programme.

```

Shift left register R7 by the register R5 value's, then return.

Registers

| Register | Value      |
|----------|------------|
| R0       | 0x0000000A |
| R1       | 0x00000012 |
| R2       | 0x0000000F |
| R3       | 0x00000000 |
| R4       | 0x20000006 |
| R5       | 0x00000000 |
| R6       | 0x00000001 |
| R7       | 0x00000001 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0x0800059  |
| R15 (PC) | 0x080005A  |
| xPSR     | 0x21000000 |

Disassembly

```

73: ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
74: ADD r4,r4,#0x01
75: ADD R1, #1 ; Increment the register R1 by 1.

66: MOV R0, #10 ; Store 10 decimal to the register R0.
67:
68: Loop3 ; Loop to find the largest power of two for each numbers of array.
69:
70: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
71: BL Pow ; Branch and link.
72: STRB R7, [R4] ; Store one byte to the result array.
73: ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
74:
75: ADD R1, #1 ; Increment the register R1 by 1.
76: SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
77: BNE Loop3 ; Keep looping if Z = 0.

79: Pow ; Function to find the largest power of two.
80: MOV R5, #-1 ; Store -1 decimal to the register R5.
81: Loop4 ; To find the number of zeros in the right.
82: ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
83: ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
84: MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
85: BEQ Loop4 ; Branch if Z=1;

87: MOV R7, #1 ; Store 1 decimal to the register R7.

```

Stored the value of register R7 into a Result.

Registers

| Register | Value      |
|----------|------------|
| R0       | 0x00000009 |
| R1       | 0x00000013 |
| R2       | 0x0000000F |
| R3       | 0x00000000 |
| R4       | 0x20000007 |
| R5       | 0x00000000 |
| R6       | 0x00000001 |
| R7       | 0x00000001 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x20001000 |
| R14 (LR) | 0x0800059  |
| R15 (PC) | 0x0800064  |
| xPSR     | 0x21000000 |

Disassembly

```

77: BNE Loop3 ; Keep looping if Z = 0.
78:
79: Pow ; Function to find the largest power of two.
80: DIF5 BNE 0x0800052
81:
82: LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
83: BL Pow ; Branch and link.
84: STRB R7, [R4] ; Store one byte to the result array.
85: ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
86:
87: ADD R1, #1 ; Increment the register R1 by 1.
88: SUBS R0, R0, [Increment] ; Decrease the value of register R1 by one(1) and set the flags.
89: BNE Loop3 ; Keep looping if Z = 0.

89: Pow ; Function to find the largest power of two.
90: MOV R5, #-1 ; Store -1 decimal to the register R5.
91: Loop4 ; To find the number of zeros in the right.
92: ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
93: ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
94: MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
95: BEQ Loop4 ; Branch if Z=1;

97: MOV R7, #1 ; Store 1 decimal to the register R7.
98: MOV R7, R7, LSL R5 ; Logical shift register by R5.
99: BX LR ; Return.

```

The register R1 and R4 indicates to the next index.

The screenshot shows the IAR Embedded Workbench interface. The top window displays the assembly code for 'proj2.s'. The code includes a loop for finding powers of two and a 'Pow' function. The registers window shows various CPU registers with their addresses and values. The memory dump window shows the memory starting at address 0x20000006, which contains binary data representing the results of the calculations.

```

Registers
Disassembly
proj2.s
    BNE Loop3 ; Keep looping if Z = 0.
    Pow : Function to find the largest power of two.
    ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
    ADD R1, #1 ; Increment the register R1 by 1.
    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
    BNE Loop3 ; Keep looping if Z = 0.

    Pow : Function to find the largest power of two.
    MOV R5, #-1 ; Store -1 decimal to the register R5.
    Loop4 : To find the number of zeros in the right.
    ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
    ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
    MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
    BEQ Loop4 ; Branch if Z=1;
    MOV R7, #1 ; Store 1 decimal to the register R7.
    MOV R7, R7, LSL R5 ; Logical shift register by R5.
    BX LR ; Return.

    END : The end of programme.

Memory 1
Address: 0x20000006
0x20000006: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000003A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000006E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Stored the Result to the new array, and repeated the loop 10 times to include all numbers.

This screenshot shows the same IAR Embedded Workbench interface as the previous one, but with modifications to the assembly code. The 'LDRB' instruction at line 70 has been added to load a byte from memory into R2. The 'BL' instruction at line 71 has been added to call the 'Pow' function. The rest of the code remains the same, including the loop and the final memory dump.

```

Registers
Disassembly
proj2.s
    BNE Loop3 ; Keep looping if Z = 0.
    Pow : Function to find the largest power of two.
    ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.
    ADD R1, #1 ; Increment the register R1 by 1.
    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
    BNE Loop3 ; Keep looping if Z = 0.

    Pow : Function to find the largest power of two.
    LDRB R2, [R1] ; To bring one byte from array and stored it into a register R2.
    BL Pow ; Branch and link.
    STRB R7, [R4] ; Store one byte to the result array.
    ADD R4, R4, #1 ; Add one to the register R4, and the result in the register R4.

    ADD R1, #1 ; Increment the register R1 by 1.
    SUBS R0, R0, #1 ; Decrease the value of register R1 by one(1) and set the flags.
    BNE Loop3 ; Keep looping if Z = 0.

    Pow : Function to find the largest power of two.
    MOV R5, #-1 ; Store -1 decimal to the register R5.
    Loop4 : To find the number of zeros in the right.
    ADD R5, R5, #1 ; Add one to the register R5, and the result in the register R5.
    ANDS R6, R2, #0x1 ; And between the register R2 and 1 and save the result in R6, then set the flag.
    MOV R2, R2, LSR #1 ; Shift left register R2 by 1 bit.
    BEQ Loop4 ; Branch if Z=1;
    MOV R7, #1 ; Store 1 decimal to the register R7.
    MOV R7, R7, LSL R5 ; Logical shift register by R5.
    BX LR ; Return.

    END : The end of programme.

Memory 1
Address: 0x20000006
0x20000006: 01 02 02 04 02 20 01 01 10 01 00 00 00 00 00 00
0x20000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000003A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x2000006E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

The outputs stored in the variable Result, underlined with blue color.

- 1-  $\text{POW}(31)(11111) = 2^0 = 01$
- 2-  $\text{POW}(74)(1001010) = 2^1 = 02$
- 3-  $\text{POW}(46)(101110) = 2^1 = 02$
- 4-  $\text{POW}(100)(1100100) = 2^2 = 04$
- 5-  $\text{POW}(10)(1010) = 2^1 = 02$
- 6-  $\text{POW}(160)(10100000) = (32 \text{ in decimal}) = 20 \text{ in hexadecimal.}$
- 7-  $\text{POW}(123)(1111011) = 2^0 = 01$
- 8-  $\text{POW}(245)(11110101) = 2^0 = 01$
- 9-  $\text{POW}(16)(10000) = (16 \text{ in decimal}) = 10 \text{ in hexadecimal.}$
- 10-  $\text{POW}(55)(110111) = 2^0 = 01$

**The fourth part is done** 

**The assignment is done**

**Thank you Dr. Ahmad Afaneh**