Experiment 6: ARM Assembly 2 - Computations

Santhosh S P ee21b119

Problem 1

Problem statement

Given a 32-bit number, generate an even parity bit for that (32-bit) word.

Flowchart

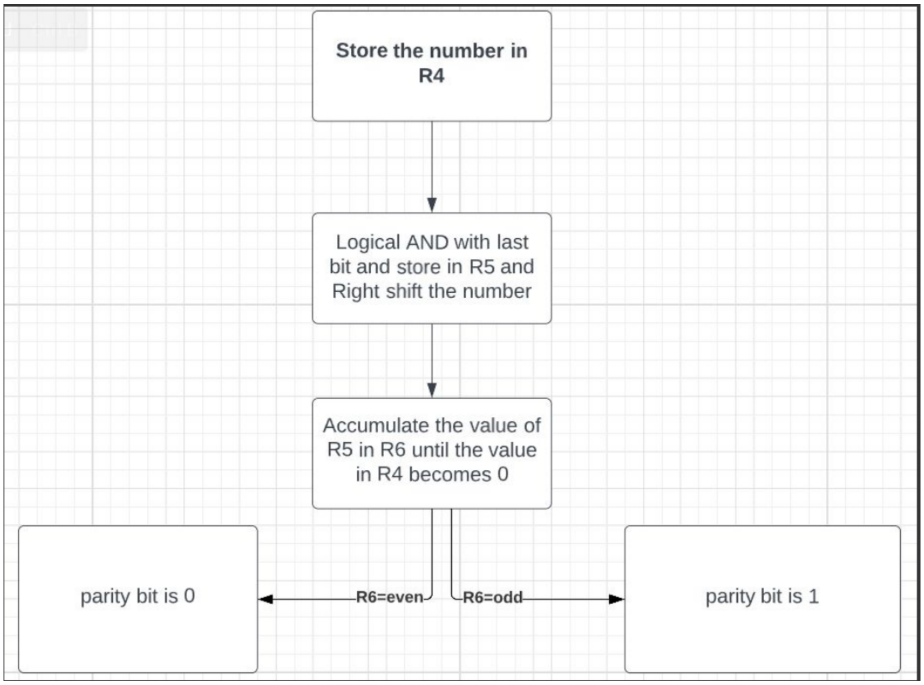

Store the number in R4
Logical AND with last bit and store in R5 and Right shift the number
Accumulate the value of R5 in R6 until the value in R4 becomes 0
R6=even
R6=odd
parity bit is 0
parity bit is 1

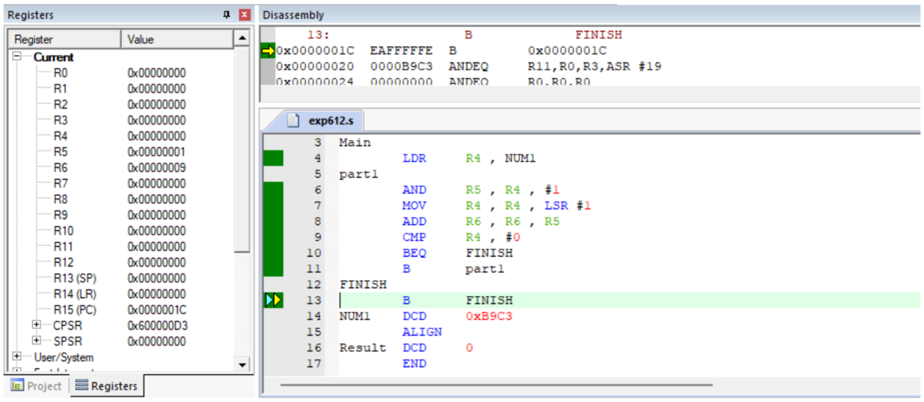
Figure 1: Flowchart for the problem

Code

```
        AREA  Program,CODE,READONLY
        ENTRY
```

```
Main
        LDR  R4 , NUM1
part1
        AND  R5 , R4 , #1
        MOV  R4 , R4 , LSR #1
        ADD  R6 , R6 , R5
        CMP  R4 , #0
        BEQ  FINISH
        B  part1
FINISH
        B  FINISH
NUM1    DCD  0xAB
        ALIGN
Result  DCD     0
        END
```

Output



Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000001
R6	0x00000009
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000001C
CPSR	0x600000D3
SPSR	0x00000000
User/System	

```
    13:                   B              FINISH
0x0000001C  EAFFFFFE  B          0x0000001C
0x00000020  0000B9C3  ANDEQ      R11,R0,R3,ASR #19
0x00000024  00000000  ANDEQ      R0,R0,R0
```

```
Main
        LDR     R4 , NUM1
part1
        AND     R5 , R4 , #1
        MOV     R4 , R4 , LSR #1
        ADD     R6 , R6 , R5
        CMP     R4 , #0
        BEQ     FINISH
        B       part1
FINISH
        B       FINISH
NUM1    DCD     0xB9C3
        ALIGN
Result  DCD     0
        END
```

Figure 2: The 32-bit input string is 0xB9C3 which is same as 0b1011100111000011. Register R6 keeps track of the number of '1's in the string (9 in this case). Register R5 contains the parity bit (1 in this case).

Problem 2

Problem statement

Determine the length of an ASCII message. All characters are 7-bit ASCII with MSB = 0. The string of characters in which the message is embedded has a starting address which is contained in the START variable. The message itself starts with an ASCII STX (Start of Text) character (0x02) and ends with ETX (End of Text) character (0x03). Save the length of the message, the number of characters between the STX and the ETX markers (but not including the markers) in the LENGTH variable.

Flowchart

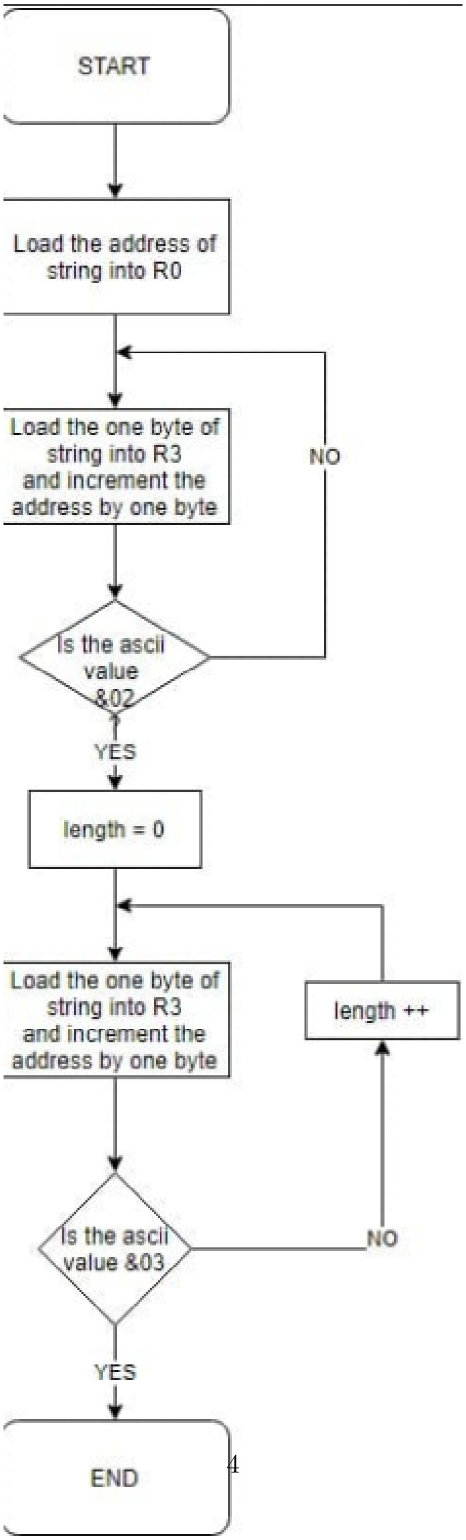
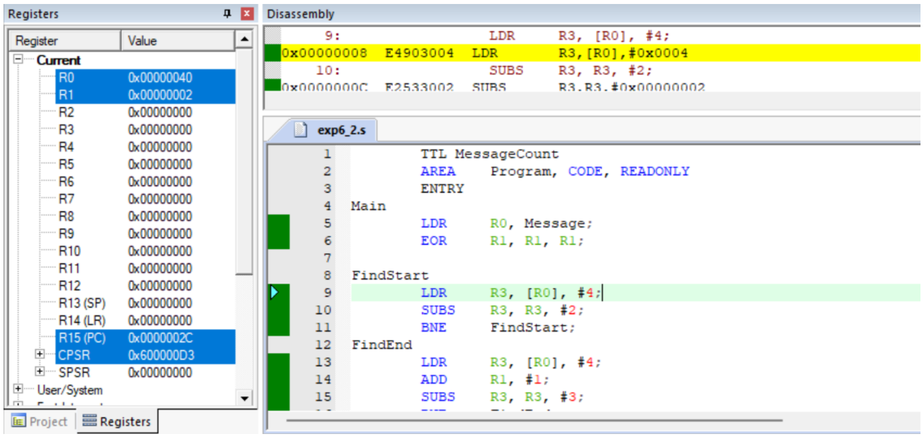

START
Load the address of string into R0
Load the one byte of string into R3 and increment the address by one byte
NO
Is the ascii value &02
YES
length = 0
Load the one byte of string into R3 and increment the address by one byte
length ++
Is the ascii value &03
NO
YES
END

Figure 3: Flowchart for the problem

Code

```
        TTL     MessageCount
        AREA    Program, CODE, READONLY
        ENTRY
Main
        LDR     R0, Message;
        EOR     R1, R1, R1;

FindStart
        LDR     R3, [R0], #4;
        SUBS    R3, R3, #2;
        BNE     FindStart;
FindEnd
        LDR     R3, [R0], #4;
        ADD     R1, #1;
        SUBS    R3, R3, #3;
        BNE     FindEnd;
Done
        SUB     R1, #1;
        STR     R1, LENGTH;
Stop
        B       Stop;
InputList
        DCD     &02;
        DCD     &12;
        DCD     &54;
        DCD     &03;
        DCD     &99;
        DCD     &FE;
        ALIGN

Message DCD     InputList;

LENGTH  DCW      0;
        ALIGN
        END
```

Output



Registers

Register	Value
Current	
R0	0x00000040
R1	0x00000002
R2	0x00000000
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)	0x0000002C
CPSR	0x600000D3
SPSR	0x00000000
User/System	

Disassembly

```
     9:                     LDR      R3, [R0], #4;
0x00000008  E4903004  LDR      R3,[R0],#0x0004
    10:                     SUBS     R3, R3, #2;
0x0000000C  E2533002  SUBS     R3,R3,#0x00000002
```

exp6_2.s

```
         TTL MessageCount
         AREA     Program, CODE, READONLY
         ENTRY
Main
         LDR      R0, Message;
         EOR      R1, R1, R1;

FindStart
         LDR      R3, [R0], #4;
         SUBS     R3, R3, #2;
         BNE      FindStart;
FindEnd
         LDR      R3, [R0], #4;
         ADD      R1, #1;
         SUBS     R3, R3, #3;
```

Figure 4: The input string has the bytes 0x12 and 0x54 between the STX and ETX character. Register R1 tracks the number of bytes between the STX and RTX character, which is 2 in this case.

Problem 3

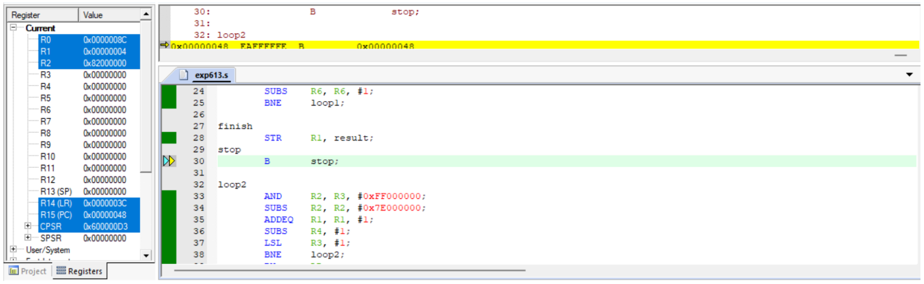
Problem statement

Given a sequence of 32-bit words (sequentially arranged) of length 8 (32 bytes or 256 bits), identify and track special bit patterns of 01111110 in the sequence (if at all appears in the sequence). [This special bit sequence is called “framing bits”, which corresponds to HDLC protocol]. Note that this special bit pattern may start at any bit, not neccessarily at byte boundaries. Framing bits, allow the digital receiver to identify the start of the frame (from the stream of bits received).

Code

```
        TTL sequenceDetector
        AREA Program, CODE, READONLY
        ENTRY
Main
        LDR     R0, List;
        EOR     R1, R1, R1;
        MOV     R3, #0xFF000000;
        MOV     R6, #8;

loop1
        LDR     R5, [R0];
        LSR     R5, #8;
        ADD     R3, R3, R5;
        MOV     R4, #24;
        BL      loop2

        LDR     R5, [R0], #4;
        AND     R5, R5, #0xFF;
        LSL     R5, #16;
        ADD     R3, R3, R5;
        MOV     R4, #8;
        BL      loop2;

        SUBS    R6, R6, #1;
        BNE     loop1;

finish
        STR     R1, result;
stop
        B       stop;

loop2
        AND     R2, R3, #0xFF000000;
        SUBS    R2, R2, #0x7E000000;
        ADDEQ   R1, R1, #1;
        SUBS    R4, #1;
        LSL     R3, #1;
        BNE     loop2;
BX      LR;

result DCW  0;
        ALIGN

start
        DCD     &00000000;
        DCD     &00000000;
        DCD     &000007E0;
        DCD     &00000007;
        DCD     &E0000000;
        DCD     &0000007E;
        DCD     &00000000;
        DCD     &0007E000;
        ALIGN

List    DCD start
        END
```

Output



Register	Value
Current	
R0	0x0000008C
R1	0x00000004
R2	0x82000000
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)	0x0000003C
R15 (PC)	0x00000048
CPSR	0x600000D3
SPSR	0x00000000
User/System	

```
    30:                 B              stop;
    31:
    32: loop2
0x00000048  EAFFFFFE  B          0x00000048
```

exp613.s

```
            SUBS    R6, R6, #1;
            BNE     loop1;

finish
            STR     R1, result;
stop
            B       stop;

loop2
            AND     R2, R3, #0xFF000000;
            SUBS    R2, R2, #0x7E000000;
            ADDEQ   R1, R1, #1;
            SUBS    R4, #1;
            LSL     R3, #1;
            BNE     loop2;
```

Figure 5: When the input 8 32-bit words are given as the input, the number of occurrences of the pattern ’01111110’ are counted and displayed in register R1 (4 in this case).