LAB REPORT– EXPERIMENT 6

ARM Assembly 2 - Computations in ARM

BY

AJOE GEORGE (EE21B009)

AND

MANDLA SIVA MANOJ (EE21B083)

GROUP 3

AIM:

To (a) learn advanced ARM instructions, conditional execution etc (b) go through example programs in Welsh and (c) write assembly language programs for the given set of problems at the end of this document.

PROBLEM STATEMENT:

Solve the following engineering problems using ARM through assembly programs:
1. Given a 32-bit number, generate even parity bit for that (32-bit) word.

2. 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

3. 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)

EQUIPMENTS REQUIRED:

The list of equipments, components required are:
1. A PC with Window OS
2. KEIL Microvision V5 IDE for ARM

PROCEDURE:

Since it is a simulation experiment, we dont need hardware. It is enough if we have a PC loaded with Keil software.
1. Go through Welsh thoroughly. Do all the home work - meaning browse ARM architecture, go on till example program 7.1(a). Demo the example program in KEIL for yourselves.
2. Write the assembly programs for the above problems (one at a time).
3. Enter the above program in KEIL software, edit and compile / assemble.
4. Run it in the 'debug' mode to see whats happening to the registers.

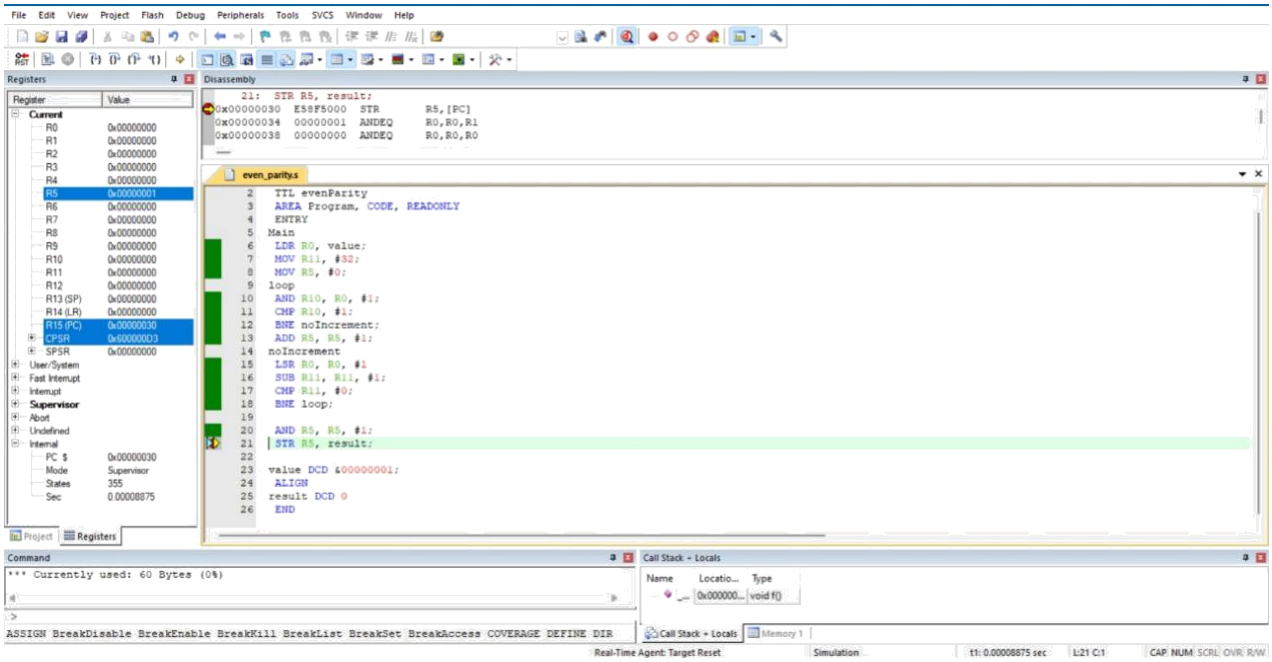
TASK:

1.EVEN PARITY:

i) INPUT WITH ODD NUMBER OF 1'S:

```
; Even Parity
 TTL evenParity
 AREA Program, CODE,
 READONLY ENTRY
Main
 LDR R0, value;
MOV R11, #32;
MOV R5, #0;
loop
 AND R10, R0, #1;
CMP R10, #1; BNE
noIncrement; ADD
R5, R5, #1;
noIncrement LSR
R0, R0, #1 SUB
R11, R11, #1; CMP
R11, #0; BNE loop;


 AND R5, R5, #1;
 STR R5, result;

value DCD &00000001;
 ALIGN
result DCD 0
 END
```

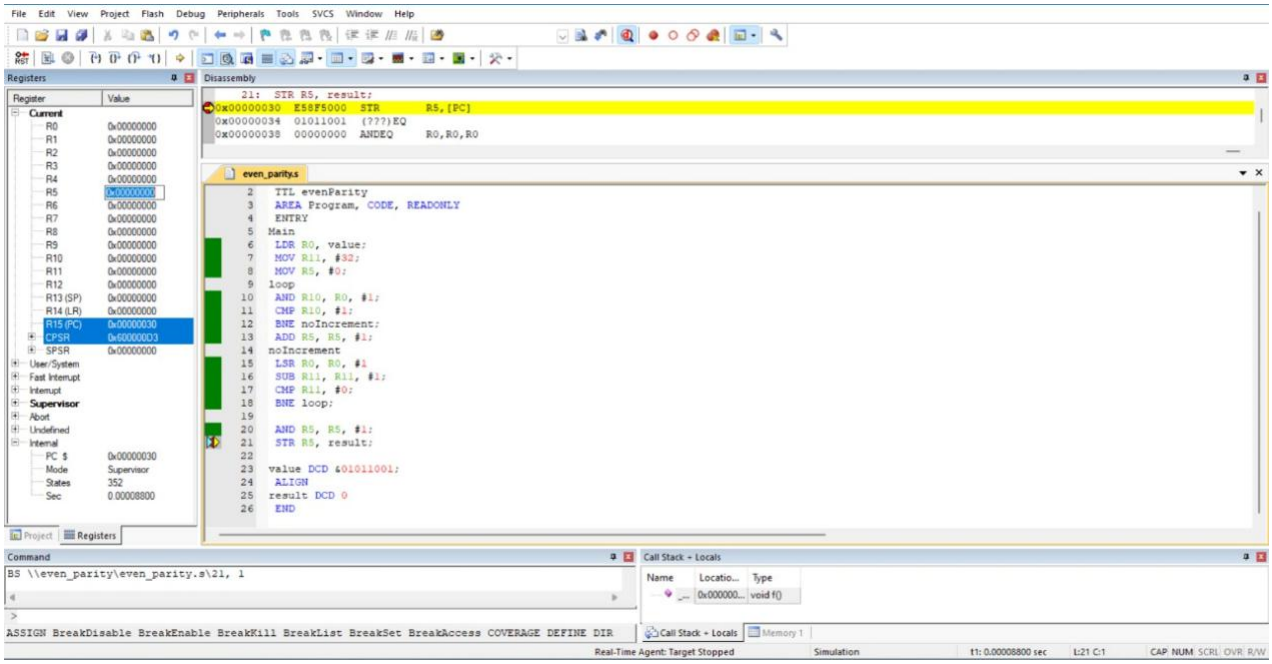
OUTPUT 1:



ii)INPUT WITH EVEN NUMBER OF 1'S:

```
; Even Parity
   TTL evenParity
   AREA Program, CODE, READONLY
   ENTRY
Main
 LDR R0, value;
 MOV R11, #32;
 MOV R5, #0;
loop
 AND R10, R0, #1;
 CMP R10, #1;
 BNE noIncrement;
 ADD R5, R5, #1;
noIncrement
 LSR R0, R0, #1
 SUB R11, R11, #1;
 CMP R11, #0;
 BNE loop;

 AND R5, R5, #1;
 STR R5, result;

value DCD &01011001;
 ALIGN
result DCD 0
 END
```

OUTPUT 2:

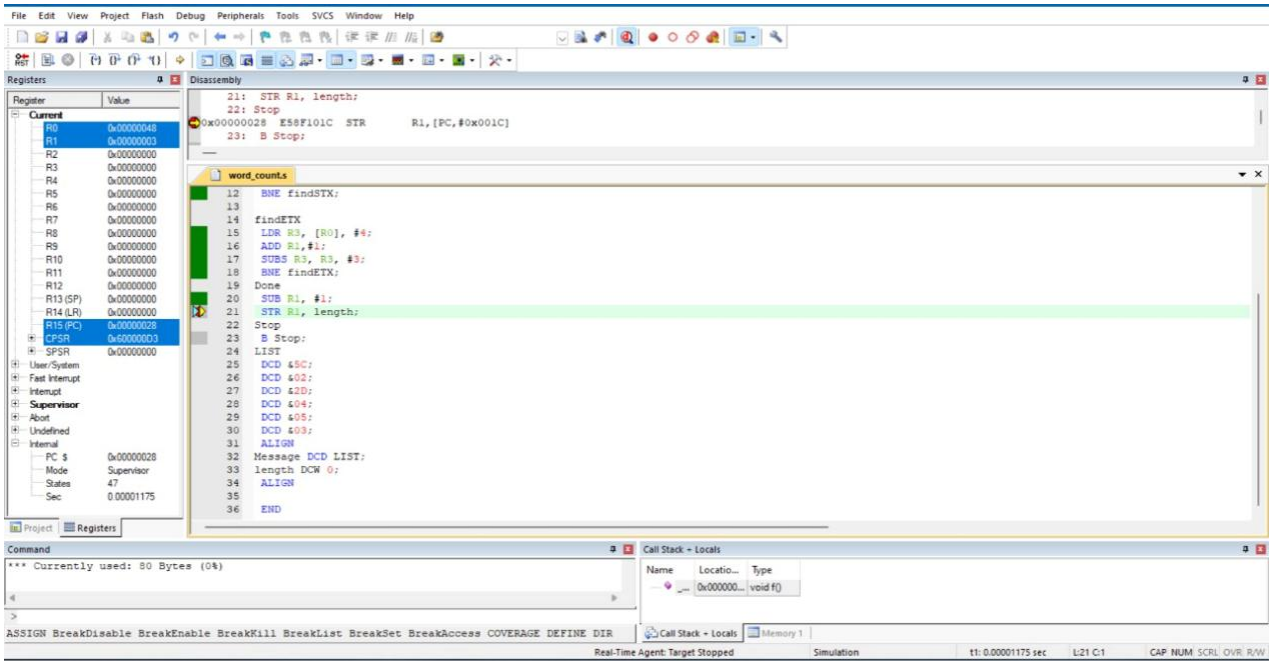


2.LENGTH OF ASCII MESSEGE:

```
; Word Count
 TTL wordcount
 AREA Program, CODE,
 READONLY ENTRY
Main
 LDR R0, Message;
 EOR R1,R1,R1;

findSTX
 LDR R3, [R0], #4;
 SUBS R3, R3, #2;
 BNE findSTX;

findETX
 LDR R3, [R0], #4;
 ADD R1,#1;
 SUBS R3, R3, #3;
 BNE findETX;
Done
 SUB R1, #1;
 STR R1, length;
Stop
 B Stop;
```

```
LIST
 DCD &5C;
 DCD &02;
 DCD &2D;
 DCD &04;
 DCD &05;
 DCD &03;
 ALIGN
Message DCD LIST;
length DCW 0;
 ALIGN

 END
```

OUTPUT:

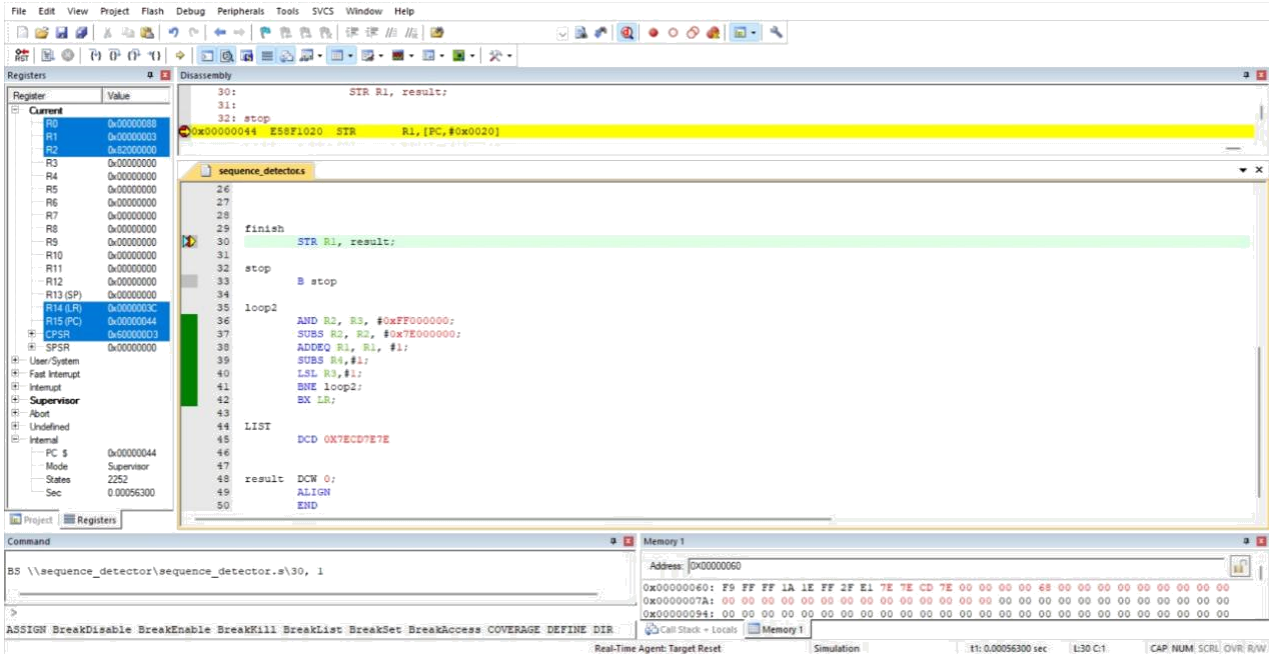


3.SEQUENCE DETECTOR:

```
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;

LIST
                DCD 0X7ECD7E7E


result DCW 0;
                ALIGN
                END
```

OUTPUT:



EXPLANATIONS:

1.EVEN PARITY:

Here if the input contains odd number of 1s then the output even parity is 1.

And if the input contains even number of 1s then the output even parity is 0.

2.LENGTH OF ASCII MESSEGE:

Word count between STX(0x02) and ETX(0x03) is stored in R1.

3.SEQUENCE DETECTOR:

Our input sequence has 7E repeated 3 times so output 3 is stored R1.