EE2016 : Microprocessor Theory and Lab

Lab Experiment # 6

ARM Assembly 2 - Computations

Batch 35

Sarathchandra Koppera EE21B073

Vikas Gollapalli EE21B051

AIM:

To

(a) Learn advanced ARM instructions, conditional execution etc.

(b) Go through example programs in Welsh.

(c) Write assembly language programs for the given set of problems at the end of this document.

Tasks to be done:

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

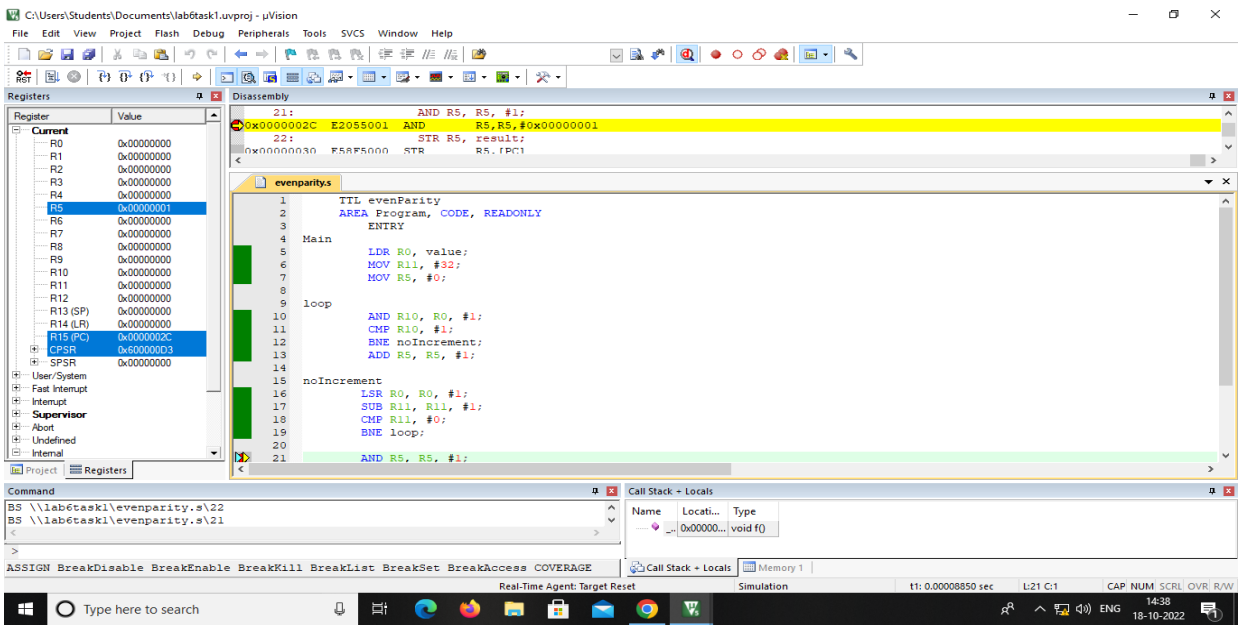
3. Given a sequence of 32-bit words (sequentially arranged) of length 8 (32 bytes or 256 bits), identify and track special bit patterns of 01111110 in the sequence (if at all appears in the sequence). [This special bit sequence is called framing bits, which corresponds to HDLC protocol]. Note that this 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).

Task-1

Code:

```
      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
```

Screenshot:



Explanation:

The code helps us to decide on the even parity. It gives us number of 1's in the given input and then reports it as the result and with that we can give the given input a even parity of 1 or 0.

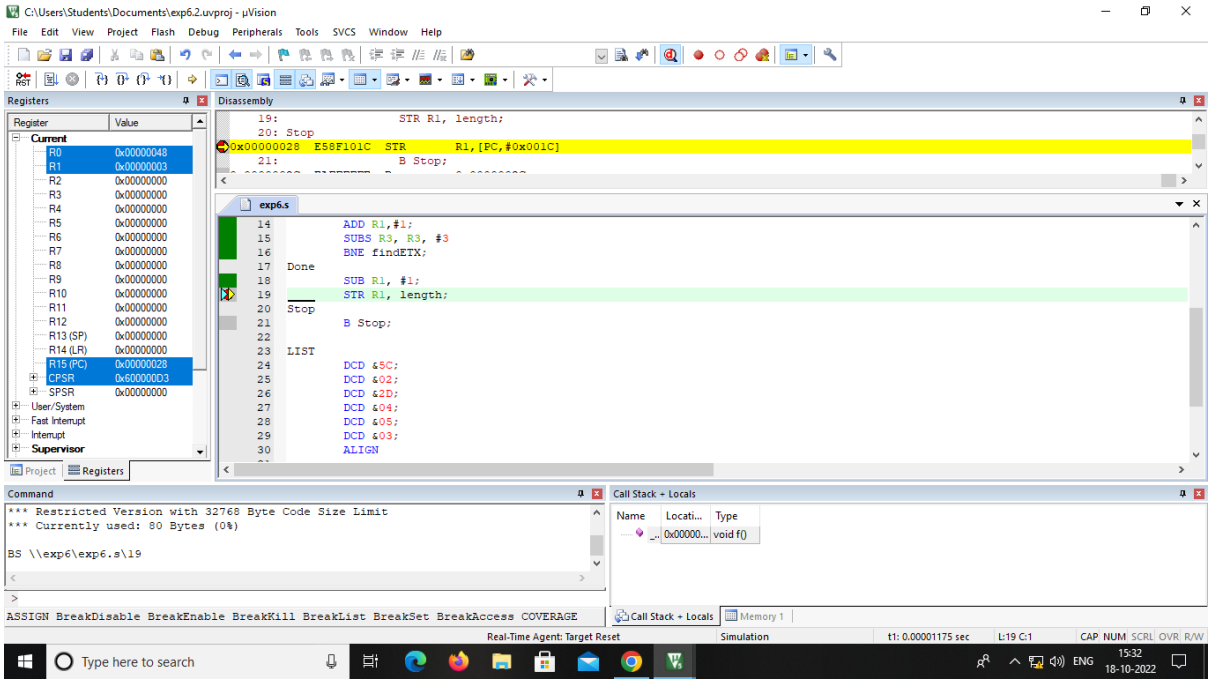
The code takes the value and analyses bit-wise. It implements AND with 1 and if the output is 1 then input is 1 and if output is 0 the input is 0. The code calculates all such ones and adds them together and displays it as the final output in R5.

Task-2

Code:

```
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 findSTX;
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;
```

Screenshot:



Explanation:

The code detects the number of characters between the fixed STX(start of the text) and ETX(end of the text). In the given code the STX is 0x02 and ETX is 0x03. The code gives the number of characters between 0x02 and 0x03 in the given list. The value stores in register R1. For the given list result is 2 i.e there are two words between 02 and 03 they are 04 and 05.

Task-3

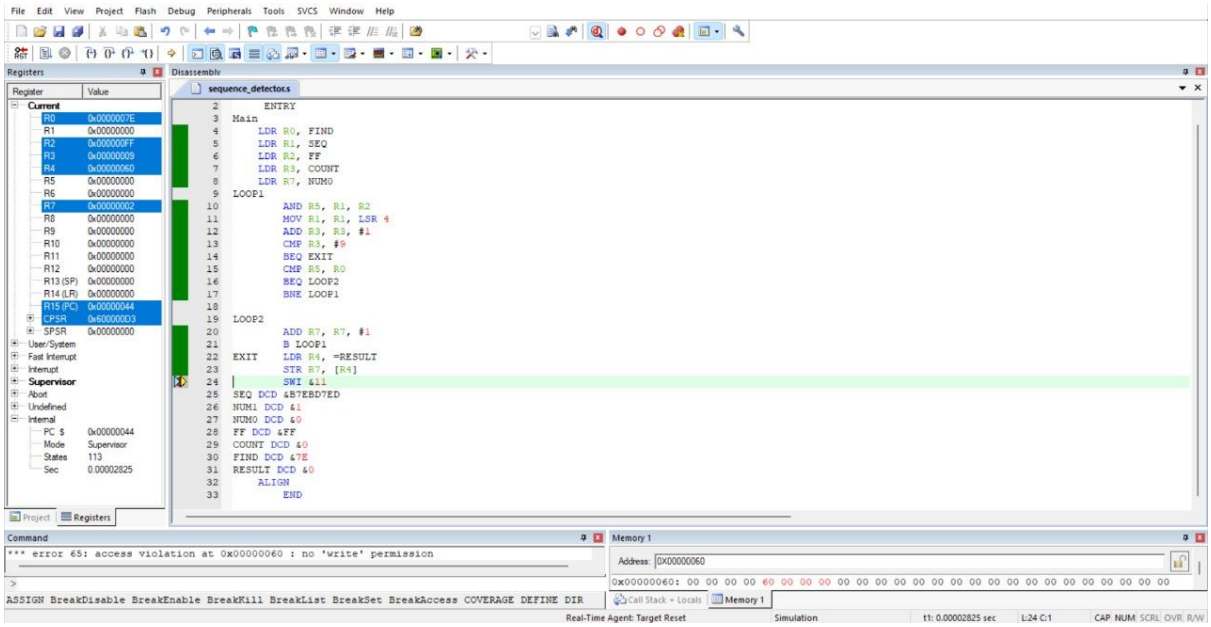
Code:

```
AREA Program, CODE, READONLY
     ENTRY
Main
    LDR R0, FIND
    LDR R1, SEQ
    LDR R2, FF
    LDR R3, COUNT
    LDR R7, NUM0
LOOP1
        AND R5, R1, R2
        MOV R1, R1, LSR 4
        ADD R3, R3, #1
        CMP R3, #9
        BEQ EXIT
        CMP R5, R0
        BEQ LOOP2
        BNE LOOP1

LOOP2
        ADD R7, R7, #1
        B LOOP1
EXIT    LDR R4, =RESULT
        STR R7, [R4]
        SWI &11
SEQ DCD &7EA7EB7E
NUM1 DCD &1
NUM0 DCD &0
FF DCD &FF
COUNT DCD &0
FIND DCD &7E
RESULT DCD &0
    ALIGN
        END
```

Explanation:

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. 01111110 is equivalent to 7E in hexadecimal. The code checks whether there are any 7E in the given input. We are comparing every 2 bits from the right end with 7E in a sequence, and if it is equal we are adding 1 to R7. At last this gives the total number of 7E(01111110) in the input number.

Screenshot:



Learning Outcomes:

1. By doing this experiment it helped us to get familiar with the ARM assembly language and its commands and how to work with it.
2. Got familiar with using Kiel software
3. It helped in thinking about the problem in different ways so that correct output could be obtained.