University of Dhaka

Department of Computer Science and Engineering

CSE-3113: Microprocessor and Assembly Language Lab

Lab Report 4

Submitted By:

Name: Mahdi Mohd. Hossain Noki

Roll No : 02

Submitted On :

February 27, 2023

Submitted To :

Upama Kabir

Md. Mustafizur Rahman

Mohammad Rafiqul Islam Rafi

1 Write an assembly language to perform all the logical operations (AND,OR,NOR,NAND,XOR,XNOR) on two variables.

1.1 16-bit Variables

This task was interesting since it restricted variable to 16bits.

```
var_one_16 DCDU 0xFFFCCFFF
var_two_16 DCDU 0x000FF000
logicalsmol

    LDRH r1, var_one_16
    LDRH r2, var_two_16

    AND r3, r1, r2                  ; r3 := r1 AND r2
    ORR r4, r1, r2                 ; r4 := r1 OR r2
    MVN r5, r4                         ; r5 := r1 NOR r2
    MVN r6, r3                         ; r6 := r1 NAND r2
    EOR r7, r1, r2                 ; r7 := r1 XOR r2
    MVN r8, r7                         ; r8 := r1 XNOR r2
```

LDR is a simple load instruction, but LDRH loads only Halfwords from a memory address. So the variables var_one_16 and var_two_16 - although were 32bit variables, only lower 16bits were loaded onto r1 and r2. The Following lines uses AND, ORR(OR), MVN(NOT) and EOR(XOR) instructions to perform the 16bit logical operations

1.2 32-bit variables

This task was similar to the last task, except LDR was used to load 32bit variables

```
logicalbig        ;Logical Operations for 32Bit numbers
    MOV r1, #0xFFFCCFFF
    MOV r2, #0x000FF000

    AND r3, r1, r2                  ; r3 := r1 AND r2
    ORR r4, r1, r2                 ; r4 := r1 OR r2
    MVN r5, r4                         ; r5 := r1 NOR r2
    MVN r6, r3                         ; r6 := r1 NAND r2
    EOR r7, r1, r2                 ; r7 := r1 XOR r2
    MVN r8, r7                         ; r8 := r1 XNOR r2
```

2 Write an assembly language to perform all the shift operations (LSR, ASR, LSL) on a 32-bit variable.

There are a total of 4 shifting operations

- ASR (Arithmetic Shift Right): Shifts the bits right and fills up the bits from left with the previous leftmost bit
- LSL (Logical Shift Left): Shifts the bits left and fills up from the right with 0s
- LSR (Logical Shift Right): Shifts the bits Right and fills up from the left with 0s
- ROR (ROtate Right): Shifts bits to the right and inserts the previous rightmost bit as the new leftmost bit

The code for the shofting operations are attached

```
shift
    MOV r0, #2147483648

    ASR r0, r0, #2
    LSR r0, r0, #1
    LSL r0, r0, #1
    ROR r0, r0, #31
```

3 Write an assembly language to perform all the arithmetic operations (Addition, Subtraction, Division and Multiplication) on two variables

3.1 Restricting input values to avoid overflow

variable values were kept small to avoid overflow

```
arithmaticsmol
    MOV r0, #3
    MOV r1, #2

    ADD r2, r1, r0 ; r2 = r1 + r0
    SUB r3, r0, r1 ; r3 = r0 - r1
    MUL r4, r0, r1 ; r4 = r0 * r1
    UDIV r5, r0, r1 ; r5 = r0 / r1
```

3.2 handling Overflow

Overflow , Negative and other flags were set to handle overflow

```
arithmaticbig
    MOV r0, #0xFFFFFFFF
    MOV r1, #0x00FF0000

    ADDS r2, r1, r0 ; r2 = r1 + r0
    SUBS r3, r0, r1 ; r3 = r0 - r1
    UMULL r5, r4, r0, r1 ; r5,r4 = r0 * r1
    UDIV r6, r0, r1 ; r6 = r0 / r1
```

4 Write an assembly language program to find the average of n numbers

```
arr DCDU 0xFF, 0xFF000000, 0xFF00, 0xFF0000
avginit        ;Average of N numbers
    LDR r1, =arr        ; declare r0 to hold address to arr
    MOV r2, #0              ; r2 is the index of array
    MOV r0, #0              ; sum r0 = 0
avg
    LDR r3, [r1, r2, LSL #2]        ;r3 = arr[i]
    ADD r0, r3                                  ;r0 = r0 + r3
    ADD r2, #1                                  ;r2++
    MOV r3, r2                                  ;r3 = r2
    SUBS r3, #4                                  ;r3 = r3 - 3
    BNE avg                                   ; return to avgb if r3 != 3
    UDIV r0, r2                                  ;r0 = r0 / 4
```

The avginit label initializes the registers and prepares them to find average for the values. memory address r1 is the base memory address for 4 variables initialised at the label arr. register r2 functions as an iterator to count upto 4. It access memory locations above r1 by multiplying itself and adding to r0. So arr[i] = r1 + r2 * 4 where r2 = i The addition of values is stored inside r0. Then 4 is subtracted from i to check if i == 4 or r2 - 4 == 0, which will set the Z flag. If the Z flag is set, register r0 is divided by 4 to find the average

5 Write an assembly language program to find the largest among n different numbers

The principal is same as the last section, except this time instead of adding numbers we store them on register r0 if they exceep the existing values in r0. We use unsigned comparison, where we find r0 - arr[i] and if the carry flag is set, it implies arr[i] ¿= r0. In this case we make r0 = arr[i], else we move forward without affecting r0

```
arr DCDU 0xFF, 0xFF000000, 0xFF00, 0xFF0000
largestinit        ;Largest of N numbers
```

```
    LDR r1, =arr        ; declare r0 to hold address to arr
    MOV r2, #0                ; r2 is the index of array
    MOV r0, #0                ; largest r0 = 0
largest
    LDR r3, [r1, r2, LSL #2]         ;r3 = arr[i]
    SUBS r3, r0                                        ;r3 = r3 - r0
    LDRCS r0, [r1, r2, LSL #2]        ;r0 = max(r0, arr[i])
    ADD r2, #1                                        ;i++
    MOV r3, r2                                        ;r3 = i
    SUBS r3, #4                                        ;r3 = r3 - 4
    BNE largest                                        ;if r3 != 4, return to largest
```

6 Write an assembly language program to find the average of n numbers using function call

To use a function call, we use branch and link BL instruction to store the current PC value in register r15 or LR. After executing the function, we return to the point where the function was called using branch information exchange BX, which swaps the current PC and r15 values

```
avginitf        ;Average of N numbers using function call
    LDR r1, =arr        ; declare r0 to hold address to arr
    MOV r2, #0                ; r2 is the index of array
    MOV r0, #0                ; sum r0 = 0
    BL avgf
    MOV r0, #0xCC00        ;testing return from function
avgf
    LDR r3, [r1, r2, LSL #2]        ;r3 = arr[i]
    ADD r0, r3                                        ;r0 = r0 + r3
    ADD r2, #1                                        ;r2++
    MOV r3, r2                                        ;r3 = r2
    SUBS r3, #4                                        ;r3 = r3 - 3
    BNE avgf                                        ;return to avgb if r3 != 3
    UDIV r0, r2                                        ;r0 = r0 / 4
    BX LR                                                ;return to function calling point
```