

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date:

Name: Nihal T M	SRN:PES2UG21CS333	Section: F
-----------------	-------------------	------------

Week# 4
1

Program Number:

Title of the Program

Write an ALP to add two 64 bit numbers loaded from memory and store the result in memory.

I. ARM Assembly Code

Code:

@ ALP program to add two 64-bit numbers

.data

a:.word 12312734,77823438

b:.word 24629982,56436738

c:.word 0,0

.text

ldr r1,=a

ldr r2,=b

ldr r3,=c

ldr r4,[r1],#4

ldr r5,[r2],#4

add r6,r4,r5

str r6,[r3],#4

ldr r4,[r1],#4

ldr r5,[r2],#4

add r6,r4,r5

str r6,[r3],#4

swi 0x11

.end

II. Output Screen Shot (One)

General Purpose | Floating | 1.s | 2.s | 3.s | 4.s | 5.s | 6.s |

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 :00000000
R1 :000011a4
R2 :000011ac
R3 :000011b4
R4 :04a37dce
R5 :035d2802
R6 :0800a5d0
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):0000102c

CPSR Register
Negative(N):0

```
@ ALP program to add two 64-bit numbers

.data
0000119C:          a:.word 12312734,77823438
000011A4:          b:.word 24629982,56436738
000011AC:          c:.word 0,0

.text
00001000:E59F1028   ldr r1,=a
00001004:E59F2028   ldr r2,=b
00001008:E59F3028   ldr r3,=c
0000100C:E4914004   ldr r4,[r1],#4
00001010:E4925004   ldr r5,[r2],#4
00001014:E0846005   add r6,r4,r5
00001018:E4836004   str r6,[r3],#4
0000101C:E4914004   ldr r4,[r1],#4
00001020:E4925004   ldr r5,[r2],#4
00001024:E0846005   add r6,r4,r5
00001028:E4836004   str r6,[r3],#4
0000102C:EF000011   swi 0x11
.end
```

OutputView | WatchView | MemoryView0

0000119c

0000119C	00BBE09E	04A37DCE	0177D2DE	035D2802	0233B37C	0800A5D0
000011D4	61762065	2065756C	6E207369	74616765	00657669	20656854

Week# 4
2

Program Number:

Title of the Program

Write an ALP to find 1's and 2's complement of a 32 bit number

I. ARM Assembly Code

Code:

@ ALP program to find 1's complement and 2's complement of a 32-bit number

.data

num:.word 32

.text

ldr r1,=num

ldr r2,[r1]

@ this is 1's complement code

mvn r3,r2

@ this is 2's complement code

rsb r5,r2,#0

swi 0x11

.end

II. Output Screen Shot (One)

General Purpose | Floating | 2.s | 3.s | 4.s | 5.s | 6.s |

Register	Value
R0	:00000000
R1	:00001160
R2	:00000020
R3	:ffffffdf
R4	:00000000
R5	:ffffffe0
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10(sl)	:00000000
R11(fp)	:00000000
R12(ip)	:00000000
R13(sp)	:00005400
R14(lr)	:00000000
R15(pc)	:00001010

```
@ ALP program to find 1's complement and 2's complement

.data
00001160:          num:.word 32

.text
00001000:E59F100C    ldr r1,num
00001004:E5912000    ldr r2,[r1]

@ this is 1's complement code
00001008:E1E03002    mvn r3,r2

@ this is 2's complement code
0000100C:E2625000    rsb r5,r2,#0

00001010:EF000011    swi 0x11

00001014:00001160    .end
```

Week# 4
 3

Program Number:

Title of the Program

Write an ALP to scan a 32 bit number if it is negative or positive

I. ARM Assembly Code

Code:

@ ALP program to scan a number and find if it is positive or negative

.data

num:.word 0

pos:.asciz "The value is positive"

neg:.asciz "The value is negative"

zer:.asciz "The value is zero"

.text

ldr r1,=num

ldr r2,[r1]

cmp r2,#0

beq zero

bpl positive

bmi negative

positive:

ldr r0,=pos

swi 0x02

b end

negative:

ldr r0,=neg

swi 0x02

b end

zero:

ldr r0,=zer

swi 0x02

b end

end: swi 0x11

.end

II. Output Screen Shot (One)

(i) when the value is positive:

The screenshot shows a debugger window with the following components:

- General Purpose Register Window:** Displays registers R0 through R15. R0 is 0000114c, R1 is 00001148, and R2 is 0000000a. The CPSR Register shows Negative(N):0.
- Assembly View:** Shows assembly code with addresses and instructions. The instruction at 00001010 is `bpl positive`, which is highlighted in blue. The instruction at 00001014 is `bmi negative`.
- Console:** Displays the output "The value is positive".

(ii) when value is negative:

The screenshot shows a debugger window with the following components:

- General Purpose Register Window:** Displays registers R0 through R15. R0 is 00001162, R1 is 00001148, and R2 is ffffffff6. The CPSR Register shows Negative(N):1.
- Assembly View:** Shows assembly code with addresses and instructions. The instruction at 00001010 is `bpl positive`, and the instruction at 00001014 is `bmi negative`, which is highlighted in blue.
- Console:** Displays the output "The value is negative".

(iii) when the value is zero:

The screenshot displays a debugger window with several panels. At the top, there are tabs for 'General Purpose', 'Floating', and '3.s', '4.s', '5.s', '6.s'. Below these, a list of registers (R0 to R15) is shown with their current values in hexadecimal. R0 is 00001178, R1 is 00001148, R2 is 00000000, and R15 (PC) is 0000103c. The CPSR Register is also visible, with Negative(N) set to 0. The main assembly view shows code with addresses and instructions. The instruction at address 0000103c is 'swi 0x11', which is highlighted in blue. The console at the bottom shows the text 'The value is zero'.

Register	Value
R0	00001178
R1	00001148
R2	00000000
R3	00000000
R4	00000000
R5	00000000
R6	00000000
R7	00000000
R8	00000000
R9	00000000
R10(sl)	00000000
R11(fp)	00000000
R12(ip)	00000000
R13(sp)	00005400
R14(lr)	00000000
R15(pc)	0000103c

CPSR Register
Negative(N):0

```
00001008:E3520000    cmp r2,#0
0000100C:0A000007    beq zero
00001010:5A000000    bpl positive
00001014:4A000002    bmi negative

00001018:                positive:
00001018:E59F0024    ldr r0,=pos
0000101C:EF000002    swi 0x02
00001020:EA000005    b end

00001024:                negative:
00001024:E59F001C    ldr r0,=neg
00001028:EF000002    swi 0x02
0000102C:EA000002    b end

00001030:                zero:
00001030:E59F0014    ldr r0,=zer
00001034:EF000002    swi 0x02
00001038:EAffFFFF    b end

0000103C:EF000011    end: swi 0x11

.end
```

OutputView | WatchView | MemoryView0 |

Console | Stdin/Stdout/Stderr |

The value is zero

Week# 4
4

Program Number:

Title of the Program

Write an ALP to find the number of zeroes, positive and negative numbers in a given array

I. ARM Assembly Code

Code:

@ ALP to find the number of zeroes, positive numbers and negative numbers in the array

.data

arr:.word 0,5,-4,0,23,-4,32,66,-3,0

.text

ldr r0,=arr

mov r1,#11

mov r2,#0 @ holds number of zeroes

mov r3,#0 @ holds number of positive numbers

mov r4,#0 @ holds number of negative numbers

loop:

ldr r5,[r0],#4

```
subs r1,r1,#1
beq end
cmp r5,#0
beq zero
bpl positive
bmi negative
```

zero:

```
add r2,r2,#1
b loop
```

positive:

```
add r3,r3,#1
b loop
```

negative:

```
add r4,r4,#1
b loop
```

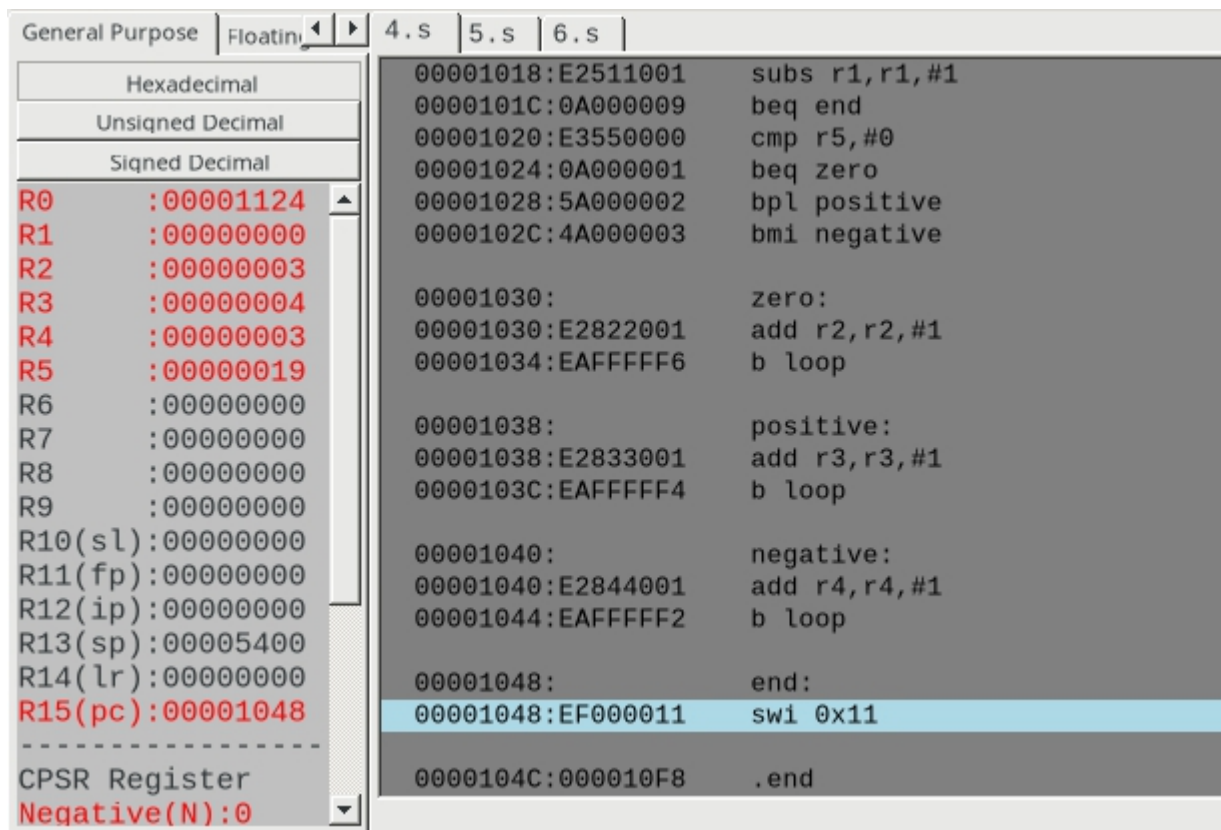
end:

```
swi 0x11
```

.end

II. Output Screen Shot (One)

here r2, r3, r4 holds the number of zeroes, positive numbers and negative numbers present in the arrys respectively:



The screenshot shows a debugger window with the following components:

- General Purpose** tab selected, with **Floatin** and **4.s** sub-tabs.
- Register List** on the left:
 - Hexadecimal, Unsigned Decimal, and Signed Decimal views are available.
 - Registers R0 through R15 are listed with their values in hexadecimal.
 - R0: 00001124
 - R1: 00000000
 - R2: 00000003
 - R3: 00000004
 - R4: 00000003
 - R5: 00000019
 - R6: 00000000
 - R7: 00000000
 - R8: 00000000
 - R9: 00000000
 - R10(sl): 00000000
 - R11(fp): 00000000
 - R12(ip): 00000000
 - R13(sp): 00005400
 - R14(lr): 00000000
 - R15(pc): 00001048
- CPSR Register** section at the bottom left, showing **Negative(N): 0**.
- Assembly Code** window on the right, showing instructions and their addresses:
 - 00001018:E2511001: subs r1,r1,#1
 - 0000101C:0A000009: beq end
 - 00001020:E3550000: cmp r5,#0
 - 00001024:0A000001: beq zero
 - 00001028:5A000002: bpl positive
 - 0000102C:4A000003: bmi negative
 - 00001030: zero:
 - 00001030:E2822001: add r2,r2,#1
 - 00001034:EAffFFFF6: b loop
 - 00001038: positive:
 - 00001038:E2833001: add r3,r3,#1
 - 0000103C:EAffFFFF4: b loop
 - 00001040: negative:
 - 00001040:E2844001: add r4,r4,#1
 - 00001044:EAffFFFF2: b loop
 - 00001048: end:
 - 00001048:EF000011: swi 0x11
 - 0000104C:000010F8: .end

Week# 4
5

Program Number:

Title of the Program

Write an ALP to count the number of 1's and 0's in a given 32 bit number.

I. ARM Assembly Code

Code:

@ ALP program to count the number of zeroes and ones in the binary number

.data

num:.word 25

.text

ldr r1,=num

ldr r2,[r1]

mov r3,#0 @ holds the number of zeroes in the no.

mov r4,#0 @ holds the number of ones in the no.

loop:and r5,r2,#1

cmp r2,#0

beq end

mov r2,r2,LSR #1

cmp r5,#0

beq zero

b one

```

zero:
    add r3,r3,#1
    b loop
one:
    add r4,r4,#1
    b loop
end:
    swi 0x11
.end

```

II. Output Screen Shot (One)

here r3 and r4 hold the number of zeroes and ones present in the binary number respectively.(excluding trailing left zeroes as they are not important)

The screenshot shows an ARM assembler simulator interface. On the left, a list of registers (R0-R15) and the CPSR register are displayed with their current values in hexadecimal. R15 (PC) is highlighted in red. The main area shows the assembly code being executed, with comments explaining the purpose of certain instructions. A 'Run' button is visible next to the code.

Register	Value
R0	:00000000
R1	:000010a8
R2	:00000000
R3	:00000002
R4	:00000003
R5	:00000000
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10(sl)	:00000000
R11(fp)	:00000000
R12(ip)	:00000000
R13(sp)	:00005400
R14(lr)	:00000000
R15(pc)	:0000103c

```

00001008:E3A03000  mov r3,#0      @ holds the number of zeroes in the number
0000100C:E3A04000  mov r4,#0      @ holds the number of ones in the number
00001010:          loop:
00001010:E2025001  and r5,r2,#1
00001014:E3520000  cmp r2,#0
00001018:0A000007  beq end
0000101C:E1A020A2  mov r2,r2,LSR #1
00001020:E3550000  cmp r5,#0
00001024:0A000000  beq zero
00001028:EA000001  b one

0000102C:          zero:
0000102C:E2833001  add r3,r3,#1
00001030:EAF0FF6   b loop

00001034:          one:
00001034:E2844001  add r4,r4,#1
00001038:EAF0FF6   b loop

0000103C:          end:
0000103C:EF000011  swi 0x11

00001040:000010A8  .end

```

Week# 4
6

Program Number:

Title of the Program

Write an ALP to check the given number has odd or even number of 1's and display the result. (Even Parity and Odd Parity)

I. ARM Assembly Code

Code:

@ ALP program to check if a given number has odd or even number of ones in binary

.data

num:.word 25

even:.asciz "Even Parity"

odd:.asciz "Odd Parity"

.text

ldr r1,=num

ldr r2,[r1]

mov r3,#0 @ number of ones in the binary format
of the given number

loop:

```
and r5,r2,#1
cmp r2,#0
beq print
mov r2,r2,LSR #1
cmp r5,#1
beq ones
b loop
```

ones:

```
add r3,r3,#1
b loop
```

print:

```
ands r6,r3,#1
beq is_even
b is_odd
```

is_odd:

```
ldr r0,=odd
swi 0x02
b end
```

is_even:

ldr r0,=even

swi 0x02

swi 0x11

end:

swi 0x11

.end

II. Output Screen Shot (One)

(i) when it is odd parity:

```
@ ALP program to check if a given number has odd or even number of ones

.data
00001064:          num:.word 25
00001068:          even:.asciz "Even Parity"
00001074:          odd:.asciz "Odd Parity"

.text
00001000:E59F1050  ldr r1,=num
00001004:E5912000  ldr r2,[r1]
00001008:E3A03000  mov r3,#0          @ number of ones in the binary
0000100C:          loop:
0000100C:E2025001  and r5,r2,#1
00001010:E3520000  cmp r2,#0
00001014:0A000005  beq print
00001018:E1A020A2  mov r2,r2,LSR #1
0000101C:E3550001  cmp r5,#1
00001020:0A000000  beq ones
00001024:EAF00000  b loop

00001028:          ones:
00001028:E2833001  add r3,r3,#1
0000102C:EAF00000  b loop

R0      :00001074
R1      :00001064
R2      :00000000
R3      :00000003
R4      :00000000
R5      :00000000
R6      :00000001
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):00001054
-----
CPSR Register
Negative(N):0

OutputView | WatchView | MemoryView0 |
Console Stdin/Stdout/Stderr
Odd Parity
```

(ii) when it is even parity:

General Purpose Floating 6.S

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 : 00001068
R1 : 00001064
R2 : 00000000
R3 : 00000002
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10(sl): 00000000
R11(fp): 00000000
R12(ip): 00000000
R13(sp): 00005400
R14(lr): 00000000
R15(pc): 00001050

CPSR Register
Negative(N): 0

```
@ ALP program to check if a given number has odd or even number of ones  
  
.data  
00001064:      num:.word 24  
00001068:      even:.asciz "Even Parity"  
00001074:      odd:.asciz "Odd Parity"  
  
.text  
00001000:E59F1050  ldr r1,=num  
00001004:E5912000  ldr r2,[r1]  
00001008:E3A03000  mov r3,#0          @ number of ones in the binary  
0000100C:          loop:  
0000100C:E2025001  and r5,r2,#1  
00001010:E3520000  cmp r2,#0  
00001014:0A000005  beq print  
00001018:E1A020A2  mov r2,r2,LSR #1  
0000101C:E3550001  cmp r5,#1  
00001020:0A000000  beq ones  
00001024:EAF00000  b loop  
  
00001028:          ones:  
00001028:E2833001  add r3,r3,#1  
0000102C:EAF00000  b loop
```

OutputView | WatchView | MemoryView0

Console Stdin/Stdout/Stderr

Even Parity

Disclaimer:

The programs and output submitted is duly written, verified and executed by me.

I have not copied from any of my peers nor from the external resource such as internet.

If found plagiarized, I will abide with the disciplinary action of the University.

Signature:

Name: Nihal T M

SRN: PES2UG21CS333

Section: F

Date: 11/02/2023