

**MICROPROCESSOR AND COMPUTER ARCHITECTURE
LABORATORY**

UE19CS256

4TH SEMESTER, ACADEMIC YEAR 2020-21

Name: Atul Anurag	SRN: PES2UG19CS075	Section: B
-------------------	--------------------	------------

Date:27-01-2021

WEEK#1

Program Number: 1

Write an ALP using ARM instruction set to add and subtract two 32-bit numbers. Both numbers are in registers.

I.ARM Assembly Code for each program

Example case:

```
[10] ex1.s
1  .text
2  MOV r0, #0x0A
3  MOV r1, #0x14
4  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
5  SUB r2, r1, r0    @difference of r0 and r1 stored in r2
6  SWI 0x011
7  .end
8
```

Test Case:

```
[10] test1.s
1  .text
2  MOV r0, #0x0f
3  MOV r1, #0x0a
4  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
5  SUB r2, r0, r1    @difference of r0 and r1 stored in r2
6  SWI 0x011
7  .end
8
```

II. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases
(one example shown in class, one example of own choice)

Example case:

Addition:

The screenshot displays the ARM development environment with three main panes. The Register Window on the left shows the state of the processor registers. The CodeView pane in the center shows the assembly code being executed. The OutputView pane at the bottom shows the console output.

Register Window:

Register	Value
R0	:0000000a
R1	:00000014
R2	:0000001e
R3	:00000000
R4	:00000000
R5	:00000000
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10 (s1)	:00000000
R11 (fp)	:00000000
R12 (ip)	:00000000
R13 (sp)	:00011400
R14 (lr)	:00000000
R15 (pc)	:0000100e

CPSR Register:

Flag	Value
Negative (N)	:0
Zero (Z)	:0
Carry (C)	:0
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1
Thumb (T)	:0
CPU Mode	:System

CodeView:

```
ex1.o
.text
00001000:E3A0000A  MOV r0, #0x0A
00001004:E3A01014  MOV r1, #0x14
00001008:E0802001  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
0000100C:E0412000  SUB r2, r1, r0    @difference of r0 and r1 stored in r2
00001010:EF000011  SWI 0x011
.end
```

OutputView:

```
Console | stdin/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex1.s
```

Subtraction:

The screenshot displays the ARM development environment with three main panes. The Register Window on the left shows the state of the processor registers. The CodeView pane in the center shows the assembly code being executed. The OutputView pane at the bottom shows the console output.

Register Window:

Register	Value
R0	:0000000a
R1	:00000014
R2	:0000000a
R3	:00000000
R4	:00000000
R5	:00000000
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10 (s1)	:00000000
R11 (fp)	:00000000
R12 (ip)	:00000000
R13 (sp)	:00011400
R14 (lr)	:00000000
R15 (pc)	:00001010

CPSR Register:

Flag	Value
Negative (N)	:0
Zero (Z)	:0
Carry (C)	:0
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1
Thumb (T)	:0
CPU Mode	:System

CodeView:

```
ex1.o
.text
00001000:E3A0000A  MOV r0, #0x0A
00001004:E3A01014  MOV r1, #0x14
00001008:E0802001  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
0000100C:E0412000  SUB r2, r1, r0    @difference of r0 and r1 stored in r2
00001010:EF000011  SWI 0x011
.end
```

OutputView:

```
Console | stdin/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex1.s
```

Test Case: Addition:

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0000000f
R1 : 0000000a
R2 : 00000019
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 0000100a

CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x000000df

CodeView

test1.o

```
.text
00001000:E3A0000F  MOV r0, #0x0f
00001004:E3A0100A  MOV r1, #0x0a
00001008:E0802001  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
0000100C:E0402001  SUB r2, r0, r1    @difference of r0 and r1 stored in r2
00001010:EF000011  SWI 0x011
.end...
```

OutputView WatchView

Console stdin/stdout/stderr

Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test1.s

Subtraction:

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0000000f
R1 : 0000000a
R2 : 00000005
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00001010

CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x000000df

CodeView

test1.o

```
.text
00001000:E3A0000F  MOV r0, #0x0f
00001004:E3A0100A  MOV r1, #0x0a
00001008:E0802001  ADD r2, r0, r1    @sum of r0 and r1 stored in r2
0000100C:E0402001  SUB r2, r0, r1    @difference of r0 and r1 stored in r2
00001010:EF000011  SWI 0x011
.end...
```

OutputView WatchView

Console stdin/stdout/stderr

Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test1.s

Output table for each program

EXAMPLE CASE: R0=10=Hex 0A R1=20=Hex 14 After Addition R2=30=Hex 1E After Subtraction R2 = 10 = Hex 0A			
R0	R1	Arithmetic Operation	Result
0x0A	0x14	ADD	R0=0x1E
0x0A	0x14	SUBTRACT	R0=0x0A
TEST CASE: R0 = 15 = Hex 0F R1 = 10 = Hex 0A After Addition R2 = 25 = Hex 19 After Subtraction R2 = 5 = Hex 05			
R0	R1	Arithmetic Operation	Result
0x0f	0x0a	ADD	R0=0x19
0x0f	0x0a	SUBTRACT	R0=0x05

Program Number: 2

Write an ALP to demonstrate logical operations. All operands are in registers.

I.ARM Assembly Code for each program

Example Case:

```
[10] ex2.s
1  .text
2  MOV r0, #0x05
3  MOV r1, #0x06
4  AND r2, r0, r1
5  ORR r3, r0, r1
6  EOR r4, r0, r1
7  MVN r5, r0
8  SWI 0x011
9  .end
10
```

Test Case:

```
[10] test2.s
1  .text
2  MOV r0, #0x0a
3  MOV r1, #0x0f
4  AND r2, r0, r1
5  ORR r3, r0, r1
6  EOR r4, r0, r1
7  MVN r5, r0
8  SWI 0x011
9  .end
10
```

II. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases
(one example shown in class, one example of own choice)

Example Case:

The screenshot displays the ARM debugger interface. The **RegistersView** window on the left shows the state of registers R0 through R15, CPSR, and CPU Mode. R15 (PC) is highlighted at 00011400. The **CodeView** window shows the assembly code for **ex2.o**, including instructions like MOV, AND, ORR, EOR, MVN, and SWI. The **OutputView** window at the bottom shows the execution log, including the loading of the assembly file, execution starting, control transfer warnings at location 113FC, and execution ending with an instruction count of 16640 and an elapsed time of 00:00:00.0552254.

```
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 00000005
R1 : 00000006
R2 : 00000004
R3 : 00000007
R4 : 00000003
R5 : ffffffff
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400
-----
CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System
0x000000df

CodeView
ex2.o
.text
00001000:E3A00005 MOV r0, #0x05
00001004:E3A01006 MOV r1, #0x06
00001008:E0002001 AND r2, r0, r1
0000100C:E1803001 ORR r3, r0, r1
00001010:E0204001 EOR r4, r0, r1
00001014:E1E05000 MVN r5, r0
00001018:EF000011 SWI 0x011
.end...

OutputView
Console | stdout/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex2.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16640 Elapsed Time:00:00:00.0552254
Instructions per second:301310
```

Test Case:

The screenshot displays the ARM debugger interface for a test case. The **RegistersView** window shows registers R0 through R15, CPSR, and CPU Mode. R15 (PC) is highlighted at 00011400. The **CodeView** window shows the assembly code for **test2.o**, including instructions like MOV, AND, ORR, EOR, MVN, and SWI. The **OutputView** window at the bottom shows the execution log, including the loading of the assembly file, execution starting, control transfer warnings at location 113FC, and execution ending with an instruction count of 16640 and an elapsed time of 00:00:00.0576941.

```
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 0000000a
R1 : 0000000f
R2 : 0000000a
R3 : 0000000f
R4 : 00000005
R5 : ffffffff5
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400
-----
CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System
0x000000df

CodeView
test2.o
.text
00001000:E3A0000A MOV r0, #0x0a
00001004:E3A0100F MOV r1, #0x0f
00001008:E0002001 AND r2, r0, r1
0000100C:E1803001 ORR r3, r0, r1
00001010:E0204001 EOR r4, r0, r1
00001014:E1E05000 MVN r5, r0
00001018:EF000011 SWI 0x011
.end

OutputView
Console | stdout/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test2.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16640 Elapsed Time:00:00:00.0576941
Instructions per second:288417
```

III. Output table for each program

EXAMPLE CASE:				
R0	R1	Logical Operation	Instruction	Result
0x05	0x06	AND	AND	R2=0x04
0x05	0x06	OR	ORR	R3=0x07
0x05	0x06	EX-OR	EOR	R4=0x03
0x05		NOT	MVN	R5=0xffffffffa

TEST CASE:				
R0	R1	Logical Operation	Instruction	Result
0x0a	0x0f	AND	AND	R2=0x0a
0x0a	0x0f	OR	ORR	R3=0x0f
0x0a	0x0f	EX-OR	EOR	R4=0x05
0x0a		NOT	MVN	R5=0xffffffff5

Program Number: 3

Write an ALP to add 5 numbers where values are present in registers.

I.ARM Assembly Code for each program.

Example Case:

```
[10] ex3.s
1  .text
2  MOV r0, #0x05
3  MOV r1, #0x06
4  MOV r2, #0x07
5  MOV r3, #0x06
6  MOV r4, #0x15
7  ADD r5, r0, r1
8  ADD r6, r5, r2
9  ADD r7, r6, r3
10 ADD r8, r7, r4
11 SWI 0x011
12 .end
13
```

Test Case:

```
[10] test3.s
1  .text
2  MOV r0, #0x01
3  MOV r1, #0x02
4  MOV r2, #0x03
5  MOV r3, #0x04
6  MOV r4, #0x05
7  ADD r5, r0, r1
8  ADD r6, r5, r2
9  ADD r7, r6, r3
10 ADD r8, r7, r4
11 SWI 0x011
12 .end
13
```

II. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases
(one example shown in class, one example of own choice)

Example Case:

RegistersView

General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	

R0 : 00000005
R1 : 00000006
R2 : 00000007
R3 : 00000006
R4 : 00000015
R5 : 0000000b
R6 : 00000012
R7 : 00000018
R8 : 0000002d
R9 : 00000000
R10 (sl) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400

CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x000000df

CodeView

ex3.o

```
.text
00001000:E3A00005 MOV r0, #0x05
00001004:E3A01006 MOV r1, #0x06
00001008:E3A02007 MOV r2, #0x07
0000100C:E3A03006 MOV r3, #0x06
00001010:E3A04015 MOV r4, #0x15
00001014:E0805001 ADD r5, r0, r1
00001018:E0856002 ADD r6, r5, r2
0000101C:E0867003 ADD r7, r6, r3
00001020:E0878004 ADD r8, r7, r4
00001024:EF000011 SWI 0x011
.end
```

OutputView

Console stdout/stdout/stderr

```
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex3.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16640 Elapsed Time:00:00:00.0552024
Instructions per second:301426
```

Test Case:

RegistersView

General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	

R0 : 00000001
R1 : 00000002
R2 : 00000003
R3 : 00000004
R4 : 00000005
R5 : 00000003
R6 : 00000006
R7 : 0000000a
R8 : 0000000f
R9 : 00000000
R10 (sl) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400

CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x000000df

CodeView

test3.o

```
.text
00001000:E3A00001 MOV r0, #0x01
00001004:E3A01002 MOV r1, #0x02
00001008:E3A02003 MOV r2, #0x03
0000100C:E3A03004 MOV r3, #0x04
00001010:E3A04005 MOV r4, #0x05
00001014:E0805001 ADD r5, r0, r1
00001018:E0856002 ADD r6, r5, r2
0000101C:E0867003 ADD r7, r6, r3
00001020:E0878004 ADD r8, r7, r4
00001024:EF000011 SWI 0x011
.end
```

OutputView

Console stdout/stdout/stderr

```
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test3.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16640 Elapsed Time:00:00:00.0500042
Instructions per second:332772
```

III. Output table for each program

EXAMPLE CASE:		
R0		0x05
R1		0x06
R2		0x07
R3		0x06
R4		0x15
R5	R0+R1	0x0b
R6	R5+R2	0x12
R7	R6+R3	0x18
R8	R7+R4	0x27

TEST CASE:		
R0		0x01
R1		0x02
R2		0x03
R3		0x04
R4		0x05
R5	R0+R1	0x03
R6	R5+R2	0x06
R7	R6+R3	0x0a
R8	R7+R4	0x0f

Program Number: 4

Write an ALP using ARM instruction set to check if a number stored in a register is even or odd. If even, store 00 in R0, else store FF in R0

I.ARM Assembly Code for each program

Example Code:

Even:

```
[10] ex4.s
1  .text
2  MOV r1, #0x06
3  ANDS r2, r1, #0x01
4
5  BEQ L1      @checks if r1 is even
6  B L2        @else
7
8  L1:         @this block is executed if r1 is even
9  MOV r0, #0x00
10 SWI 0x011
11 B end       @ends the program if L1 is executed
12
13 L2:         @this block is executed if r1 is odd
14 MOV r0, #0xFF
15 SWI 0x011
16
17 end:
18 .end
19
```

Odd:

```
[10] ex4.s
1  .text
2  MOV r1, #0x05
3  ANDS r2, r1, #0x01
4
5  BEQ L1      @checks if r1 is even
6  B L2        @else
7
8  L1:         @this block is executed if r1 is even
9  MOV r0, #0x00
10 SWI 0x011
11 B end       @ends the program if L1 is executed
12
13 L2:         @this block is executed if r1 is odd
14 MOV r0, #0xFF
15 SWI 0x011
16
17 end:
18 .end
19
```

Test Case: Even:

```
[10] test4.s
1  .text
2  MOV r1, #0x0a
3  ANDS r2, r1, #0x01
4
5  BEQ L1          @checks if r1 is even
6  B L2           @else
7
8  L1:             @this block is executed if r1 is even
9  MOV r0, #0x00
10 SWI 0x011
11 B end          @ends the program if L1 is executed
12
13 L2:             @this block is executed if r1 is odd
14 MOV r0, #0xFF
15 SWI 0x011
16
17 end:
18 .end
19
```

Odd:

```
[10] test4.s
1  .text
2  MOV r1, #0x0f
3  ANDS r2, r1, #0x01
4
5  BEQ L1          @checks if r1 is even
6  B L2           @else
7
8  L1:             @this block is executed if r1 is even
9  MOV r0, #0x00
10 SWI 0x011
11 B end          @ends the program if L1 is executed
12
13 L2:             @this block is executed if r1 is odd
14 MOV r0, #0xFF
15 SWI 0x011
16
17 end:
18 .end
19
```

II. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases
(one example shown in class, one example of own choice)

Example Case:

Even:

The screenshot shows the ARM debugger interface. The Register Window on the left displays the state of registers R0 through R15 and the CPSR. R15 (PC) is 00011400. The Code View on the right shows the assembly code for 'ex4.o'. The code includes instructions for MOV, ANDS, BEQ, B, L1, L2, MOV, and SWI. The Output View at the bottom shows the execution log, including the loading of the assembly file, execution starting, and ending with a count of 16637 instructions and an elapsed time of 00:00:00.0468928.

```
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 00000000
R1 : 00000006
R2 : 00000000
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1): 00000000
R11 (fp): 00000000
R12 (ip): 00000000
R13 (sp): 00011400
R14 (lr): 00000000
R15 (pc): 00011400
-----
CPSR Register
Negative (N): 0
Zero (Z): 1
Carry (C): 0
Overflow (V): 0
IRQ Disable: 1
FIQ Disable: 1
Thumb (T): 0
CPU Mode: System
0x400000df

CodeView
ex4.o
.text
00001000:E3A01006 MOV r1, #0x06
00001004:E2112001 ANDS r2, r1, #0x01
00001008:0A000000 BEQ L1 @checks if r1 is even
0000100C:EA000002 B L2 @else
00001010:E3A00000 L1: @this block is executed if r1 is even
00001014:EF000011 MOV r0, #0x00
00001018:EA000001 SWI 0x011 @ends the program if L1 is executed
0000101C:E3A000FF B end @this block is executed if r1 is odd
00001020:EF000011 L2: MOV r0, #0xFF
end: SWI 0x011
.end

OutputView WatchView
Console | stdin/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex4.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16637 Elapsed Time:00:00:00.0468928
Instructions per second:354787
```

Odd:

The screenshot shows the ARM debugger interface for the 'Odd' test case. The Register Window on the left displays the state of registers R0 through R15 and the CPSR. R15 (PC) is 00011400. The Code View on the right shows the assembly code for 'ex4.o'. The code includes instructions for MOV, ANDS, BEQ, B, L1, L2, MOV, and SWI. The Output View at the bottom shows the execution log, including the loading of the assembly file, execution starting, and ending with a count of 16637 instructions and an elapsed time of 00:00:00.0534033.

```
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 000000ff
R1 : 00000005
R2 : 00000001
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1): 00000000
R11 (fp): 00000000
R12 (ip): 00000000
R13 (sp): 00011400
R14 (lr): 00000000
R15 (pc): 00011400
-----
CPSR Register
Negative (N): 0
Zero (Z): 0
Carry (C): 0
Overflow (V): 0
IRQ Disable: 1
FIQ Disable: 1
Thumb (T): 0
CPU Mode: System
0x000000df

CodeView
ex4.o
.text
00001000:E3A01005 MOV r1, #0x05
00001004:E2112001 ANDS r2, r1, #0x01
00001008:0A000000 BEQ L1 @checks if r1 is even
0000100C:EA000002 B L2 @else
00001010:E3A00000 L1: @this block is executed if r1 is even
00001014:EF000011 MOV r0, #0x00
00001018:EA000001 SWI 0x011 @ends the program if L1 is executed
0000101C:E3A000FF B end @this block is executed if r1 is odd
00001020:EF000011 L2: MOV r0, #0xFF
end: SWI 0x011
.end

OutputView WatchView
Console | stdin/stdout/stderr
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\ex4.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16637 Elapsed Time:00:00:00.0534033
Instructions per second:311535
```

Test Case: Even:

RegistersView | **CodeView**

General Purpose | Floating Point

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 : 00000000
R1 : 0000000a
R2 : 00000000
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400

CPSR Register
Negative (N) : 0
Zero (Z) : 1
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x400000df

test4.o

```
.text
00001000:E3A0100A MOV r1, #0x0a
00001004:E2112001 ANDS r2, r1, #0x01
00001008:0A000000 BEQ L1      @checks if r1 is even
0000100C:EA000002 B L2      @else
                                L1:      @this block is executed if r1 is even
00001010:E3A00000 MOV r0, #0x00
00001014:EF000011 SWI 0x011
00001018:EA000001 B end      @ends the program if L1 is executed
                                L2:      @this block is executed if r1 is odd
0000101C:E3A000FF MOV r0, #0xFF
00001020:EF000011 SWI 0x011
                                end:
                                .end
```

OutputView | **WatchView**

Console | stdin/stdout/stderr

Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test4.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16637 Elapsed Time:00:00:00.0534014
Instructions per second:311546

Odd:

RegistersView | **CodeView**

General Purpose | Floating Point

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 : 000000ff
R1 : 0000000f
R2 : 00000001
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00000000
R15 (pc) : 00011400

CPSR Register
Negative (N) : 0
Zero (Z) : 0
Carry (C) : 0
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : System

0x000000df

test4.o

```
.text
00001000:E3A0100F MOV r1, #0x0f
00001004:E2112001 ANDS r2, r1, #0x01
00001008:0A000000 BEQ L1      @checks if r1 is even
0000100C:EA000002 B L2      @else
                                L1:      @this block is executed if r1 is even
00001010:E3A00000 MOV r0, #0x00
00001014:EF000011 SWI 0x011
00001018:EA000001 B end      @ends the program if L1 is executed
                                L2:      @this block is executed if r1 is odd
0000101C:E3A000FF MOV r0, #0xFF
00001020:EF000011 SWI 0x011
                                end:
                                .end
```

OutputView | **WatchView**

Console | stdin/stdout/stderr

Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Lab1 Student copy\test4.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16637 Elapsed Time:00:00:00.0534032
Instructions per second:311535

III. Output table for each program

EXAMPLE CASE			
CASE 1	R1		0x06
	R2	After AND operation	0x00
	R0	(EVEN)	0x00
CASE 2	R1		0x05
	R2	After AND operation	0x01
	R0	(ODD)	0xFF

TEST CASE			
CASE 1	R1		0x0A
	R2	After AND operation	0x00
	R0	(EVEN)	0x00
CASE 2	R1		0x0F
	R2	After AND operation	0x01
	R0	(ODD)	0xFF

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: *Atul Anurag*

Name: Atul Anurag

SRN: PES2UG19CS075

Section: B

Date: 27-01-2021