

# Assignment 1

Q1

a) Direct addressing addition

The screenshot shows a debugger interface with two main panels. The left panel, titled 'Registers', displays a list of registers and their current values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Execution. Below this, the 'Disassembly' tab is active, showing the assembly code for 'add.asm'.

Register	Value
R0	0x00000000
R1	0x00000002
R2	0x00000045
R3	0x00000047
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000

Source	Count	Execution
Event Start/Stop Group A		
Event Start/Stop Group B		

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3      LDR R1, VALUE1
4      LDR R2, VALUE2
5      ADD R3, R2, R1
6      SWI &11
7      AREA PROGRAM, DATA, READONLY
8  VALUE1 DCD &00000002
9  VALUE2 DCD &00000045
10  END
```

**Result Stored in R3**

CODE:

```
AREA PROGRAM, CODE, READONLY
```

```
ENTRY
```

```
LDR R1, VALUE1
```

```
LDR R2, VALUE2
```

```
ADD R3, R2, R1
```

```
SWI &11
```

```
AREA PROGRAM, DATA, READONLY
```

```
VALUE1 DCD &00000002
```

```
VALUE2 DCD &00000045
```

```
END
```

## Direct addressing Subtraction

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their current values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Execution Timing. Below this, the 'Disassembly' tab is active, showing the assembly code for 'add.asm'.

Register	Value
<b>Current</b>	
R0	0x00000000
R1	0x00000002
R2	0x00000045
R3	0x00000043
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000010
R15 (PC)	0x00000008
CPSR	0x000000D3
SPSR	0x000000D3
User/System	
Fast Interrupt	
Interrupt	
Supervisor	

Source	Count	Execution Timing
Event Start/Stop Group A		
Event Start/Stop Group B		

Disassembly

add.asm

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R1, VALUE1
4 LDR R2, VALUE2
5 SUB R3, R2, R1
6 SWI &11
7 AREA PROGRAM, DATA, READONLY
8 VALUE1 DCD &00000002
9 VALUE2 DCD &00000045
10 END
```

### Result Stored in R3

CODE:

```
AREA PROGRAM, CODE, READONLY
```

```
ENTRY
```

```
LDR R1, VALUE1
```

```
LDR R2, VALUE2
```

```
SUB R3, R2, R1
```

```
SWI &11
```

```
AREA PROGRAM, DATA, READONLY
```

```
VALUE1 DCD &00000002
```

```
VALUE2 DCD &00000045
```

```
END
```

## b) Indirect Addressing Addition

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers with their current values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Exec. Below this, there are tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing the assembly code for 'INDIRECT.asm'. The code includes an entry point, several load instructions, an addition instruction, a software interrupt, and data definitions.

Register	Value
R0	0x00000000
R1	0x00000500
R2	0x00000504
R3	0x00000060
R4	0x00000020
R5	0x00000080
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000

Source	Count	Exec
Event Start/Stop Group A		
Event Start/Stop Group B		

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3      LDR R1, VALUE1
4      LDR R2, VALUE2
5      LDR R3, [R1]
6      LDR R4, [R2]
7      ADD R5, R3, R4
8      SWI &11
9      AREA PROGRAM, DATA, READONLY
10 VALUE1 DCD &00000500
11 VALUE2 DCD &00000504
12      END
13
```

Result stored in R5

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
LDR R2, VALUE2
LDR R3, [R1]
LDR R4, [R2]
ADD R5, R3, R4
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000500
VALUE2 DCD &00000504
END
```

## Indirect Addressing subtraction

The screenshot shows a debugger interface with two main panels. The left panel, titled 'Registers', displays a list of registers and their current values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Exec. Below this, there are tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing the assembly code for 'INDIRECT.asm' and 'test\_shift.asm'. The assembly code is as follows:

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R1, VALUE1
4 LDR R2, VALUE2
5 LDR R3, [R1]
6 LDR R4, [R2]
7 SUB R5, R3, R4
8 SWI &11
9 AREA PROGRAM, DATA, READONLY
10 VALUE1 DCD &00000500
11 VALUE2 DCD &00000504
12 END
13
```

The 'Registers' panel shows the following values:

Register	Value
R0	0x00000000
R1	0x00000500
R2	0x00000504
R3	0x00000060
R4	0x00000020
R5	0x00000040
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000018
R15 (PC)	0x00000008
CPSR	0x000000D3
SPSR	0x000000D3

Result stored in R5

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
LDR R2, VALUE2
LDR R3, [R1]
LDR R4, [R2]
SUB R5, R3, R4
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000500
VALUE2 DCD &00000504
END
```

### c) Barrel shifter addition

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Executi. Below this, there are tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing the assembly code for 'lest\_shift.asm'. The code includes an area for program code, an entry point, a load instruction for R1, an add instruction for R3 (shifting R1 left by 2 bits), a software interrupt instruction (SWI &11), and an area for program data. The assembly code is as follows:

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R1, VALUE1
4 ADD R3, R1, R1, LSL #2
5 SWI &11
6 AREA PROGRAM, DATA, READONLY
7 VALUE1 DCD &00000002
8 END
9
```

Result stored in R3

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
ADD R3, R1, R1, LSL #2
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000002
END
```

## Barrel Shifter Subtraction

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their current values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Execution. Below this, there are tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing the assembly code for 'lest\_shift.asm'. The code includes an entry point, a load instruction for R1, a subtraction instruction for R3, a software interrupt instruction, and a data definition for VALUE1.

Register	Value
R0	0x00000000
R1	0x00000002
R2	0x00000000
R3	0xFFFFFFFF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x0000000C
R15 (PC)	0x00000008
CPSR	0x000000D3
SPSR	0x000000D3
User/System	
Fast Interrupt	
Interrupt	

Source	Count	Execution
Event Start/Stop Group A		
Event Start/Stop Group B		

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3      LDR R1, VALUE1
4      SUB R3, R1, R1, LSL #2
5      SWI &11
6      AREA PROGRAM, DATA, READONLY
7  VALUE1 DCD &00000002
8      END
9
```

### Result stored in R3

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
SUB R3, R1, R1, LSL #2
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000002
END
```



## Q2) Left Shift

The screenshot displays a debugger window with two main panes. The left pane, titled 'Registers', shows a list of registers from R0 to R15, along with CPSR and SPSR. Register R2 is highlighted with a blue background and shows a value of 0x00000114. Register R15 (PC) is also highlighted and shows 0x00000008. The right pane, titled 'Event Statistics', has tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing assembly code from a file named 'add.asm'. The code includes an AREA declaration for PROGRAM CODE, an ENTRY point, a load instruction (LDR R1, VALUE1), a left shift instruction (MOV R2, R1, LSL #2) which is highlighted in green, a software interrupt instruction (SWI &11), another AREA declaration for PROGRAM DATA, a data definition (VALUE1 DCD &00000045), and an END instruction. The 'Event Statistics' tab shows a table with columns for Source, Count, and Execution Timing, with two entries for 'Event Start/Stop Group A' and 'Event Start/Stop Group B'.

Register	Value
R0	0x00000000
R1	0x00000045
R2	0x00000114
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000008
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

Source	Count	Execution Timing
Event Start/Stop Group A		
Event Start/Stop Group B		

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R1, VALUE1
4 MOV R2, R1, LSL #2
5 SWI &11
6 AREA PROGRAM, DATA, READONLY
7 VALUE1 DCD &00000045
8 END
```

### Result Stored in R2

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
MOV R2, R1, LSL #2
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000045
END
```

## Right Shift

The screenshot shows a debugger interface with two main panels. The left panel, titled 'Registers', displays a list of registers and their values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Execution Time. Below this, the 'Disassembly' tab is active, showing the assembly code for a file named 'add.asm'.

**Registers Panel:**

Register	Value
<b>Current</b>	
R0	0x00000000
R1	0x00000045
<b>R2</b>	<b>0x00000011</b>
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000008</b>
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	

**Event Statistics Panel:**

Source	Count	Execution Time
Event Start/Stop Group A		
Event Start/Stop Group B		

**Disassembly Panel (add.asm):**

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  LDR R1, VALUE1
4  MOV R2, R1, LSR #2
5  SWI &11
6  AREA PROGRAM, DATA, READONLY
7  VALUE1 DCD &00000045
8  END
```

## Result Stored in R2

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R1, VALUE1
MOV R2, R1, LSR #2
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000045
END
```



### Q3) Even Odd

The screenshot shows an ARM assembly debugger interface. On the left, a 'Register' window lists registers R0 through R15, CPSR, and SPSR. R0 has a value of 0x00000005, R1 has 0x00000001, and R14 (LR) has 0x0000000C. The SPSR register is highlighted with a value of 0x000000D3. On the right, the 'Source' window shows assembly code for 'mul.asm'. The code includes an entry point, a load instruction for R0, an AND instruction for R1, a software interrupt instruction (SWI &11), and a data definition for VALUE1. The instruction 'SWI &11' is highlighted in green, and a green arrow points to it. Above the source window, there are tabs for 'Disassembly' and 'Event Statistics', and a table with columns 'Source', 'Count', and 'Exec'.

Register	Value
R0	0x00000005
R1	0x00000001
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x0000000C
R15 (PC)	0x00000008
CPSR	0x000000D3
SPSR	0x000000D3

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, VALUE1
4 AND R1, R0, #1
5 SWI &11
6 AREA PROGRAM, DATA, READONLY
7 VALUE1 DCD &00000005
8 END
```

Result is stored in R1

If R1 is 1 number is odd

If R1 is 0 number is even

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, VALUE1
AND R1, R0, #1
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000005
END
```

#### Q4) Multiplication using repeated addition

The screenshot shows a debugger interface with two main panes. The left pane displays the register window with the following values:

Register	Value
R0	0x00000005
R1	0x0000000F
R2	0x00000003
R3	0x00000003
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000024
R15 (PC)	0x0000001C
CPSR	0x600000D3
SPSR	0x600000D3

The right pane shows the assembly code for the program, with the 'mul.asm' file selected. The code is as follows:

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, VALUE1
4 LDR R2, VALUE2
5 MOV R1, R0
6 MOV R3, #1
7 LOOP
8 ADD R3, #1
9 ADD R1, R0
10 CMP R2, R3
11 BNE LOOP
12 SWI &11
13 AREA PROGRAM, DATA, READONLY
14 VALUE1 DCD &00000005
15 VALUE2 DCD &00000003
16 END
```

#### Result Stored in R1

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, VALUE1
LDR R2, VALUE2
MOV R1, R0
MOV R3, #1
LOOP
ADD R3, #1
ADD R1, R0
CMP R2, R3
BNE LOOP
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000005
VALUE2 DCD &00000003
END
```

## Q5) Multiplication Table

The screenshot displays an ARM assembly editor with three main panels. The top-left panel, titled 'Registers', shows a list of registers from R0 to R15 (PC) and CPSR/SPSR, with their current values. R15 (PC) is highlighted with a blue bar. The top-right panel shows the assembly code for 'mul.asm'. The code starts with 'AREA PROGRAM, CODE, READONLY', followed by 'ENTRY'. The main code block includes 'LDR R0, NUM', 'MOV R1, R0', 'MOV R2, #1', and 'MOV R3, 0x40000000'. A loop labeled 'LOOP' contains 'STR R1, [R3], #4', 'ADD R2, #1', 'ADD R1, R0', 'CMP R2, #10', and 'BNE LOOP'. The code ends with 'SWI &11', 'AREA PROGRAM, DATA, READONLY', 'NUM DCD &00000003', and 'END'. The bottom panel, titled 'Memory 1', shows the memory address '0x40000000' and a hex dump of the memory contents, which are mostly zeros.

Register	Value
R0	0x00000003
R1	0x0000001E
R2	0x0000000A
R3	0x40000024
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0x600000D3
SPSR	0x00000000

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, NUM
4 MOV R1, R0
5 MOV R2, #1
6 MOV R3, 0x40000000
7 LOOP
8 STR R1, [R3], #4
9 ADD R2, #1
10 ADD R1, R0
11 CMP R2, #10
12 BNE LOOP
13 SWI &11
14 AREA PROGRAM, DATA, READONLY
15 NUM DCD &00000003
16 END
17
```

Memory 1

Address: 0x40000000

0x40000000: 03 00 00 00 06 00 00 00 09 00 00 00 0C 00 00 00 0F 00 00 00 12 00 00 00

0x40000018: 15 00 00 00 18 00 00 00 1B 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x40000030: 00

0x40000048: 00

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, NUM
MOV R1, R0
MOV R2, #1
MOV R3, 0x40000000
LOOP
STR R1, [R3], #4
ADD R2, #1
ADD R1, R0
CMP R2, #10
BNE LOOP
SWI &11
AREA PROGRAM, DATA, READONLY
NUM DCD &00000003
END
```

## Q6) Division using subtraction

The screenshot shows a debugger interface with two main panels. The left panel, titled 'Registers', displays a list of registers and their values. The right panel, titled 'Event Statistics', shows a table with columns for Source, Count, and Execution Time. Below this, there are tabs for 'Disassembly' and 'Event Statistics'. The 'Disassembly' tab is active, showing assembly code for a file named 'mul.asm'. The code includes an entry point, a loop, and a switch instruction. The registers panel shows the following values:

Register	Value
R0	0x00000008
R1	0x00000002
R2	0x00000003
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000024
R15 (PC)	0x00000020
CPSR	0x800000D3
SPSR	0x800000D3
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	

The assembly code in the 'mul.asm' file is as follows:

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, VALUE1
4 LDR R2, VALUE2
5 MOV R1, R0
6 MOV R3, #0
7 LOOP
8 ADD R3, #1
9 SUB R1, R2
10 CMP R1, R2
11 BGE LOOP
12 SWI &11
13 AREA PROGRAM, DATA, READONLY
14 VALUE1 DCD &00000008
15 VALUE2 DCD &00000003
16 END
```

Result stored in R3

And remainder in R1

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, VALUE1
LDR R2, VALUE2
MOV R1, R0
MOV R3, #0
LOOP
ADD R3, #1
SUB R1, R2
CMP R1, R2
BGE LOOP
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000008
VALUE2 DCD &00000003
END
```

### Q7) Greatest and smallest of 2 numbers

The screenshot shows an ARM assembly debugger interface. On the left, the 'Registers' window displays the current state of registers. R0 through R12 have values of 0x00000003 or 0x00000009. R13 (SP) is 0x00000000. R14 (LR) is 0x0000002C. R15 (PC) is 0x00000008. CPSR and SPSR are 0x800000D3. On the right, the 'Event Statistics' window shows the assembly code being executed. The code is in the 'mul.asm' file and includes instructions for loading values, comparing them, and branching based on the result. The current instruction being executed is 'CMP R0, R1'.

Register	Value
R0	0x00000003
R1	0x00000009
R2	0x00000009
R3	0x00000003
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x0000002C
R15 (PC)	0x00000008
CPSR	0x800000D3
SPSR	0x800000D3

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, VALUE1
4 LDR R1, VALUE2
5 CMP R0, R1
6 BGT GREATER
7 BLT LESSER
8
9 GREATER
10 MOV R2, R0
11 MOV R3, R1
12 SWI &11
13
14 LESSER
15 MOV R2, R1
16 MOV R3, R0
17 SWI &11
18 AREA PROGRAM, DATA, READONLY
19 VALUE1 DCD &00000003
20 VALUE2 DCD &00000009
21 END
```

**GREATER IS STORED IN R2 AND LESSER IN R3**

```
AREA PROGRAM, CODE, READONLY
```

```
ENTRY
```

```
LDR R0, VALUE1
```

```
LDR R1, VALUE2
```

```
CMP R0, R1
```

```
BGT GREATER
```

```
BLT LESSER
```

```
GREATER
```

```
MOV R2, R0
```

```
MOV R3, R1
```

```
SWI &11
```

```
LESSER
```

```
MOV R2, R1
```

```
MOV R3, R0
```

```
SWI &11
```

```
AREA PROGRAM, DATA, READONLY
```

```
VALUE1 DCD &00000003
```

```
VALUE2 DCD &00000009
```

```
END
```



### Q8) 1's and 2's complement of a number

**Registers**

Register	Value
<b>Current</b>	
R0	0x00000007
R1	0xFFFFFFFF8
R2	0xFFFFFFFF9
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000010
<b>R15 (PC)</b>	<b>0x00000008</b>
CPSR	0x000000D3
SPSR	0x000000D3
User/System	

**Event Statistics**

Source	Count	Execution Timing
Event Start/Stop Group A		
Event Start/Stop Group B		

**Disassembly** | **Event Statistics**

INDIRECT.asm | comp.asm | **mul.asm**

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  LDR R0, VALUE1
4  MVN R1, R0
5  ADD R2, R1, #1
6  SWI &11
7  AREA PROGRAM, DATA, READONLY
8  VALUE1 DCD &00000007
9  END
```

**1's complement is stored in R1**

**2's complement is stored in R2**

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, VALUE1
MVN R1, R0
ADD R2, R1, #1
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000007
END
```



### Q9) Factorial of a number

The screenshot shows a debugger interface with two main panels. The left panel, titled 'Registers', lists registers R0 through R15 (PC), CPSR, and SPSR. R15 (PC) is highlighted with a value of 0x00000018. The right panel, titled 'Event Statistics', shows a table with columns 'Source', 'Count', and 'Execut'. Below this, there are tabs for 'INDIRECT.asm', 'comp.asm', and 'mul.asm'. The 'mul.asm' tab is active, displaying assembly code. The code includes a loop for calculating a factorial, with the instruction 'SWI &11' highlighted in green. A yellow arrow points to the instruction 'SWI &11' in the assembly code.

Register	Value
R0	0x00000001
R1	0x00000018
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x600000D3
SPSR	0x00000000

Source	Count	Execut
Event Start/Stop Group A		
Event Start/Stop Group B		

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 LDR R0, VALUE1
4 MOV R1, R0
5 LOOP
6 SUB R0, #1
7 MUL R1, R0, R1
8 CMP R0, #1
9 BNE LOOP
10 SWI &11
11 AREA PROGRAM, DATA, READONLY
12 VALUE1 DCD &00000004
13 END
```

Result is stored in R1

```
AREA PROGRAM, CODE, READONLY
ENTRY
LDR R0, VALUE1
MOV R1, R0
LOOP
SUB R0, #1
MUL R1, R0, R1
CMP R0, #1
BNE LOOP
SWI &11
AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000004
END
```