

H M Mythreya

PES2UG20CS130

MPCA-Lab Week-1

Task 1: Write a program to check if a number stored in a register is even or odd. If even, store 00 in R0, else FF in R0.

```
.text
    mov r1, #13
    ands r2,r1,#1
    beq even
    mov r0,#0xFF
    B finish
even:
    mov r0, #0x00
finish:
    SWI 0x011

.end
```

The screenshot displays a debugger interface with two main panels. The left panel, titled 'RegistersView', shows the state of various registers. R0 contains the value 255, R1 contains 13, R2 contains 1, and R3 through R9 are all 0. R10 (s1) through R14 (lr) are also 0. R15 (pc) is 4120. Below the registers, the CPSR Register is shown with various flags: Negative (N) is 0, Zero (Z) is 0, Carry (C) is 0, Overflow (V) is 0, IRQ Disable is 1, FIQ Disable is 1, Thumb (T) is 0, and CPU Mode is System. The right panel, titled 'CodeView', shows the assembly code for 'even_odd.o'. The code starts with '.text' and includes instructions: 'mov r1, #13', 'ands r2,r1,#1', 'beq even', 'mov r0,#0xFF', and 'B finish'. The 'even:' label is followed by 'mov r0, #0x00'. The 'finish:' label is followed by 'SWI 0x011'. The code ends with '.end...'.

Register	Value
R0	255
R1	13
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10 (s1)	0
R11 (fp)	0
R12 (ip)	0
R13 (sp)	70656
R14 (lr)	0
R15 (pc)	4120

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

```
.text
00001000:E3A0100D    mov r1, #13
00001004:E2112001    ands r2,r1,#1
00001008:0A000001    beq even
0000100C:E3A000FF    mov r0,#0xFF
00001010:EA000000    B finish
even:
00001014:E3A00000    mov r0, #0x00
finish:
00001018:EF000011    SWI 0x011

.end...
```

2) Write a program to compare the value of R0 and R1, add if R0 = R1, else subtract

```
.text
    mov r0, #10
    mov r1, #5
    cmp r0,r1
    beq equ
    sub r2,r1,r0
    b finish
```

```
equ:
    add r2,r0,r1
```

```
finish:
    SWI 0x011
```

The screenshot displays two windows from a debugger interface:

- RegistersView:** Shows the state of ARM registers. The 'Signed Decimal' column is selected. R0 is 10, R1 is 5, R2 is -5, and R15 (PC) is 4132. The CPSR register shows various flags like Negative (N), Zero (Z), Carry (C), Overflow (V), IRQ Disable, FIQ Disable, Thumb (T), and CPU Mode (System).
- CodeView:** Shows the assembly code for the program. The current instruction is highlighted: `SWI 0x011...` at address `0000101C:EF000011`.

3) Based on the value of the number in R0, Write an ALP to store 1 in R1 if R0 is zero, Store 2 in R1 if R0 is positive, store 3 in R1 if R0 is negative.

```
.text
    mov r0,#0
    cmp r0,#0
    moveq r1,#1
    beq finish
    movmi r1,#3
    bmi finish
    mov r1,#2
```

```
finish:
    SWI 0x011
```

```
.end
```

The screenshot displays a debugger interface with two main panes: RegistersView on the left and CodeView on the right.

RegistersView: The 'General Purpose' tab is selected. The 'Signed Decimal' column is active. The registers are listed as follows:

Register	Value
R0	: 0
R1	: 1
R2	: 0
R3	: 0
R4	: 0
R5	: 0
R6	: 0
R7	: 0
R8	: 0
R9	: 0
R10 (s1)	: 0
R11 (fp)	: 0
R12 (ip)	: 0
R13 (sp)	: 70656
R14 (lr)	: 0
R15 (pc)	: 4124

Below the registers, the CPSR Register is shown with the following status:

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

The bottom of the RegistersView shows the memory address 0x600000df.

CodeView: The 'p3.o' file is loaded. The assembly code is displayed in the following format:

Address	Disassembly
00001000:E3A00000	mov r0,#0
00001004:E3500000	cmp r0,#0
00001008:03A01001	moveq r1,#1
0000100C:0A000002	beq finish
00001010:43A01003	movmi r1,#3
00001014:4A000000	bmi finish
00001018:E3A01002	mov r1,#2
finish:	
0000101C:EF000011	SWI 0x011
.end...	

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

```
.text
    mov r0,#0
    cmp r0,#0
    moveq r1,#1
    beq finish
    movmi r1,#3
    bmi finish
    mov r1,#2
```

```
finish:
    SWI 0x011
```

```
.end
```

The screenshot displays two windows from an ARM development tool. The **RegistersView** window on the left shows the state of the ARM registers. The **CodeView** window on the right shows the assembly code being executed.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0 : 123
- R1 : 456
- R2 : 579
- R3 : 333
- R4 : 0
- R5 : 0
- R6 : 0
- R7 : 0
- R8 : 0
- R9 : 0
- R10 (s1) : 0
- R11 (fp) : 0
- R12 (ip) : 0
- R13 (sp) : 70656
- R14 (lr) : 0
- R15 (pc) : 4116
-
- 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:

- add_sub.o
- .text
- 00001000:E3A0007B mov r0, #123
- 00001004:E3A01F72 mov r1, #456
- 00001008:E0802001 add r2,r0,r1
- 0000100C:E0413000 sub r3,r1,r0
- 00001010:EF000011 SWI 0x011
- .end...

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

```
.text
    mov r0,#0
    mov r1,#1
    and r2,r1,r0
    orr r3,r1,r0
.end
```

The screenshot displays a debugger interface with two main panels: **RegistersView** on the left and **CodeView** on the right.

RegistersView: The 'General Purpose' tab is active. It shows a list of registers (R0-R15) and the CPSR Register. The values are as follows:

Register	Value
R0	0
R1	1
R2	0
R3	1
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10 (s1)	0
R11 (fp)	0
R12 (ip)	0
R13 (sp)	70656
R14 (lr)	0
R15 (pc)	4116

Below the registers, the CPSR Register status is shown:

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

The bottom of the register view shows the memory address **0x000000df**.

CodeView: The file **logical.o** is loaded. It shows the assembly code for the **.text** section:

```
00001000:E3A00000    mov r0,#0
00001004:E3A01001    mov r1,#1
00001008:E0012000    and r2,r1,r0
0000100C:E1813000    ORR r3,r1,r0
.end...
```

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

```
.text
mov r1,#5
mov r2,#7
mov r3,#3
mov r4,#11
mov r5,#9
add r6,r1,r2
add r7,r3,r4
add r8,r6,r7
add r8,r8,r5
.end
```

The screenshot displays a debugger interface with two main panels: **RegistersView** and **CodeView**.

RegistersView: This panel shows the state of the processor's registers. It has tabs for **General Purpose** and **Floating Point**. Under the **General Purpose** tab, there are sub-tabs for **Hexadecimal**, **Unsigned Decimal**, and **Signed Decimal**. The **Signed Decimal** tab is selected. The registers are listed as follows:

Register	Value
R0	: 0
R1	: 5
R2	: 7
R3	: 3
R4	: 11
R5	: 9
R6	: 12
R7	: 14
R8	: 35
R9	: 0
R10 (s1)	: 0
R11 (fp)	: 0
R12 (ip)	: 0
R13 (sp)	: 70656
R14 (lr)	: 0
R15 (pc)	: 4136

Below the registers, the **CPSR Register** is shown with the following status:

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

The current PC value is **0x000000df**.

CodeView: This panel shows the assembly code for the file **add5.o**. The code is as follows:

```
.text
00001000:E3A01005    mov r1,#5
00001004:E3A02007    mov r2,#7
00001008:E3A03003    mov r3,#3
0000100C:E3A0400B    mov r4,#11
00001010:E3A05009    mov r5,#9
00001014:E0816002    add r6,r1,r2
00001018:E0837004    add r7,r3,r4
0000101C:E0868007    add r8,r6,r7
00001020:E0888005    add r8,r8,r5
.end...
```