

NAME: SAGAR LADLA

ROLL: 2024HT01123

SUBJECT: EMBEDDED SYSTEM DESIGN

LAB ASSIGNMENT: 1

ANS 1:

- a) On reset ARM7TDMI processor's **state is 0** and **mode is Supervisor**.
- b) States taken to execute instructions:
- i) Arithmetic (ADR, ADD, SUB): 1
 - ii) Load (LDR): 3
 - iii) Store (STR): 2
- c) Number of states taken for BGE "(1) if taken and (2) not taken" are the same in both cases.
- i) Number of states taken by BGE if branch:
 - 1) Taken: 1
 - 2) Not taken: 1
- d) Performance of code-1 and code-2 for the following conditions:

Here we are taking a and b as first 2 values of DCD instruction, based on that I will be counting states:

Condition	Code-1-States	Code-2-States
a<b a=0x20 b=0x40	states=36 (if we execute `B stop` instruction) states=33 (if we do not execute `B stop` instruction)	states=33 (if we execute `B stop` instruction) states=30 (if we do not execute `B stop` instruction)
a>b a=0x40 b=0x20	states=32 (if we execute `B stop` instruction) states=29 (if we do not execute `B stop` instruction)	states=32 (if we execute `B stop` instruction) states=29 (if we do not execute `B stop` instruction)
a=b a=0x20 b=0x20	states=32 (if we execute `B stop` instruction) states=29 (if we do not execute `B stop` instruction)	states=32 (if we execute `B stop` instruction) states=29 (if we do not execute `B stop` instruction)

ANS 2:

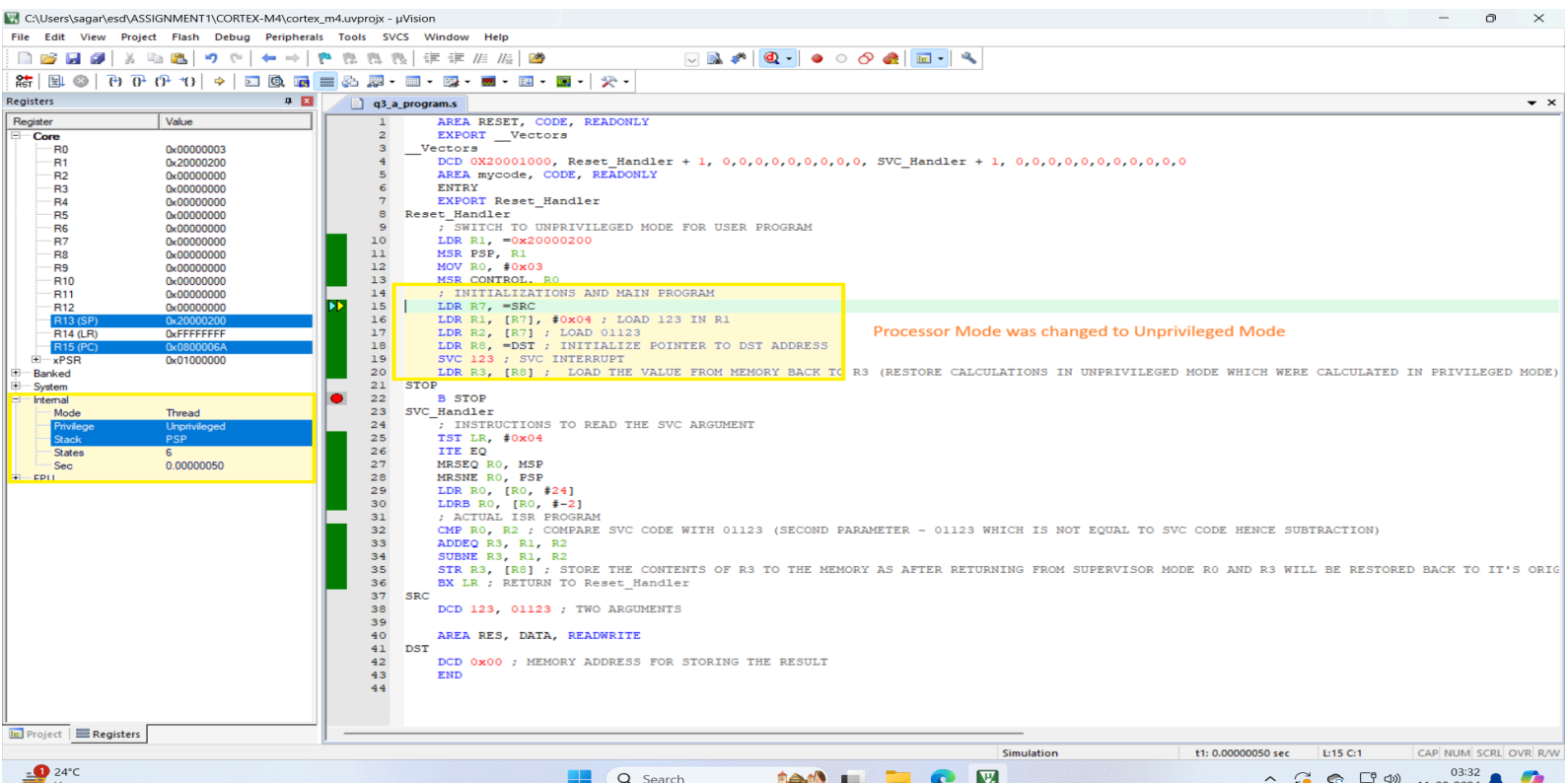
The screenshot displays the pVision IDE interface. On the left, the 'Registers' window shows the current state of registers R0 through R15, with R0 and R1 highlighted. The main window shows the disassembly of a program named 'q2_program.s'. The code is as follows:

```
1 AREA RESET, CODE, READONLY
2 ENTRY
3 START
4 LDR R6, =SRC
5 LDR R4, [R6]
6 BL PROG
7 MOV R5, #0x05 ; COUNT OF DIGITS
8 SUB R5, R5, R7 ; R5 - R7 TO CALCULATE LEADING 0s
9 ADD R0, R0, R5 ; ADD THE NUMBER OF LEADING 0s TO R0
10 STOP
11 B STOP
12 PROG
13 ADD R7, R7, #0x01 ; DIGIT COUNTER
14
15 CMP R4, #0x00
16 ADDEQ R0, R0, #0x01
17 MOVEQ PC, LR
18
19 CMP R4, #0x01
20 ADDEQ R1, R1, #0x01
21 MOVEQ PC, LR
22
23 MOV R2, #0x00
24 PUSH {LR} ; SAVING LR CONTEXT
25 BL DIV
26 CMP R5, #0x01
27 ADDEQ R1, R1, #0x01
28 CMP R5, #0x00
29 ADDEQ R0, R0, #0x01
30 POP {LR} ; RESTORING LR CONTEXT
31 B PROG
32 DIV
33 SUBS R4, R4, #0x0a
34 ADD R2, R2, #0x01
35 BFL DIV
36 SUB R2, R2, #0x01 ; QUOTIENT
37 ADD R5, R4, #0x0a ; REMAINDER
38 MOV R4, R2 ; R4 = R2
39 BX LR
40 SRC
41 DCD 00123 ; 5 DIGIT ROLL
42 END
```

A comment at the bottom of the disassembly window states: 'Two 0's and One 1' We got the same value in R0 = 2 and R1 = 1. The status bar at the bottom shows 'Real-Time Agent: Target Stopped' and 'Simulation' mode.

We have achieved the division (modulus) via repeated subtraction and broke the loop once negative flag set to get the remainder (modulus)

After RESET, setting program execution to unprivileged mode.



CASE 1: WHEN SVC ARGUMENT AND ANY ONE PARAMETER MATCHES WITH BITS ID (ADDITION)

The screenshot displays the Keil uVision IDE interface during simulation.

Registers Window:

Register	Value
R0	0x0000007B
R1	0x0000007B
R2	0x00000463
R3	0x000004DE
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x08000A4
R8	0x20000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(SP)	0x20001000
R14(LR)	0xFFFFFFFF
R15(PC)	0x0000005E
xPSR	0x6100000B

SVC Handler Assembly Code (q3_a_program.s):

```
1 AREA RESET, CODE, READONLY  
2 EXPORT __Vectors  
  
3 _Vectors  
4 DCD 0X20001000, Reset_Handler + 1, 0, 0, 0, 0, 0, 0, 0, SVC_Handler + 1, 0, 0, 0, 0, 0, 0, 0, 0  
5 AREA mycode, CODE, READONLY  
6 ENTRY  
7 EXPORT Reset_Handler  
  
8 Reset_Handler  
; SWITCH TO UNPRIVILEGED MODE FOR USER PROGRAM  
9  
10 LDR R1, =0x20000200  
11 MSR PSP, R1  
12 MOV RO, #0x03  
13 MSR CONTROL, RO  
14 ; INITIALIZATIONS AND MAIN PROGRAM  
15 LDR RT, =SRC  
16 LDR R1, [RT], #0x04 ; LOAD 123 IN R1  
17 LDR R2, [RT] ; LOAD 01123  
18 LDR RS, =DST ; INITIALIZE POINTER TO DST ADDRESS  
19 SVC 123 ; SVC INTERRUPT  
20 LDR R3, [RS] ; LOAD THE VALUE FROM MEMORY BACK TO R3 (RESTORE CALCULATIONS IN UNPRIVILEGED MODE WHICH WERE CALCULATED IN PRIVILEGED MODE)  
  
21 STOP  
22 B STOP  
23 SVC_Handler  
; INSTRUCTIONS TO READ THE SVC ARGUMENT  
24  
25 TST LR, #0x04  
26 ITE EQ  
27 MRSEQ RO, MSP  
28 MRSNE RO, PSP  
29 LDR RO, [RO, #24]  
30 LDREB RO, [RO, #-2]  
31 ; ACTUAL ISR PROGRAM  
32 CMP RO, R1 : COMPARE SVC CODE WITH 123 (FIRST PARAMETER - 123 WHICH IS EQUAL TO SVC CODE HENCE ADDITION)  
33 ADDEQ R3, R1, R2  
34 SUBNE R3, R1, R2  
35 STR R3, [RS] ; STORE THE CONTENTS OF R3 TO THE MEMORY AS AFTER RETURNING FROM SUPERVISOR MODE RO AND R3 WILL BE RESTORED BACK TO IT'S ORIG  
36 BX LR ; RETURN TO Reset_Handler  
  
37 SRC  
38 DCD 123, 01123 ; TWO ARGUMENTS  
39  
40 AREA RES, DATA, READWRITE  
41 DST  
42 DCD 0x00 ; MEMORY ADDRESS FOR STORING THE RESULT  
43 END  
44
```

Annotations:

- "Parameters passed in R1 and R2": Points to lines 16-17 where R1 and R2 are loaded from memory at address 0x20000200.
- "Comparing R0 == R1; Then Addition as R0 == R1": Points to lines 25-26 (TST LR) and 32 (CMP).
- "Reading SVC argument and storing in R0": Points to lines 27-30 which load values from the stack into R0.
- "Ox7B (R1) + Ox463 (R2) = Ox04DE (R3)": Points to line 33 (ADDEQ), explaining how the result in R3 is derived from R1 and R2.

Status Bar: Simulation | t1: 0.00000325 sec | L:36 C:1 | CAP NUM SCRL OVR RAW | Step one line | 24°C Haze | Search | 11-09-2024

CASE 2: WHEN SVC ARGUMENT AND ANY ONE PARAMETER NOT MATCHES WITH BITS ID (SUBTRACTION)

The screenshot displays the Keil uVision IDE interface during a simulation. The main window shows the assembly source file `q3_a_program.s`. The code defines a reset handler that switches to unprivileged mode, initializes registers, and processes an SVC interrupt by comparing R0 and R2, subtracting R2 from R0, and storing the result in R3.

Registers Window:

Register	Value
R0	0x0000007B
R1	0x0000007B
R2	0x00000463
R3	0xFFFFFC18
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x080000A4
R8	0x20000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800009E
xPSR	0x8100000B

Assembly Code Snippets:

```

1 AREA RESET, CODE, READONLY
2 EXPORT __Vectors
3 __Vectors
4 DCD 0X20001000, Reset_Handler + 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5 AREA mycode, CODE, READONLY
6 ENTRY
7 EXPORT Reset_Handler
8 Reset_Handler
9 ; SWITCH TO UNPRIVILEGED MODE FOR USER PROGRAM
10 LDR R1, =0x20000200
11 MSR PSP, R1
12 MOV R0, #0x03
13 MSR CONTROL, R0
14 ; INITIALIZATIONS AND MAIN PROGRAM
15 LDR R7, =SRC
16 LDR R1, [R7], #0x04 ; LOAD 123 IN R1
17 LDR R2, [R7] ; LOAD 01123
18 LDR R0, =DST ; INITIALIZE POINTER TO DST ADDRESS
19 SVC 123 ; SVC INTERRUPT
20 LDR R3, [R8] ; LOAD THE VALUE FROM MEMORY BACK TO R3 (RESTORE CALCULATIONS IN UNPRIVILEGED MODE WHICH WERE CALCULATED IN PRIVILEGED MODE)
21 STOP
22 B STOP
23 SVC_Handler
24 ; INSTRUCTIONS TO READ THE SVC ARGUMENT
25 TST LR, #0x04
26 ITE EQ
27 MRSEQ R0, MSP
28 MRSNE R0, PSP
29 LDR R0, [R0, #24]
30 LDRB R0, [R0, #-2]
31 ; ACTUAL ISR PROGRAM
32 CMP R0, R2 ; COMPARE SVC CODE WITH 01123 (SECOND PARAMETER - 01123 WHICH IS NOT EQUAL TO SVC CODE HENCE SUBTRACTION)
33 ADDEQ R3, R1, R2
34 SUBNE R3, R1, R2
35 STR R3, [R8] ; STORE THE CONTENTS OF R3 TO THE MEMORY AS AFTER RETURNING FROM SUPERVISOR MODE R0 AND R3 WILL BE RESTORED BACK TO IT'S ORIG
36 BX LR ; RETURN TO Reset_Handler
37 SRC
38 DCD 123, 01123 ; TWO ARGUMENTS
39
40 AREA RES, DATA, READWRITE
41 DST
42 DCD 0x00 ; MEMORY ADDRESS FOR STORING THE RESULT
43 END
  
```

Annotations:

- "Loading arguments in R0 and R1" points to lines 16-17.
- "Reading SVC argument and storing in R0" points to lines 29-30.
- "Comparing R0 and R2; Then Subtraction as R0 != R2" points to lines 32-34.
- "R0 = 123 (SVC argument)" and "R1 = 00123 (Parameter R2)" are noted near the comparison.
- "0x7B (R0) - 0x463 (R2) = 0xFFFFFC13 (R3) (Negative result)" points to the subtraction result in R3.

Status Bar: Simulation, tt: 0.00000325 sec, L:36 C:1, CAP: NUM SCRL OVR/ RAW, Step one line, 24°C Haze, 03:37 11-09-2024.