

LAB3

Aniket Sambher

Reg no-190905466

Roll no-58

SOLVED EXAMPLE

AREA RESET, DATA, READONLY

EXPORT __Vectors

__Vectors

DCD 0x1001000

DCD Reset_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset_Handler

Reset_Handler

LDR R1,=VALUE1

LDR R2,=VALUE2

UMULL R3,R4,R2,R1

LDR R2,=RESULT

STR R4,[R2]

ADD R2, #4

STR R3,[R2]

STOP

B STOP

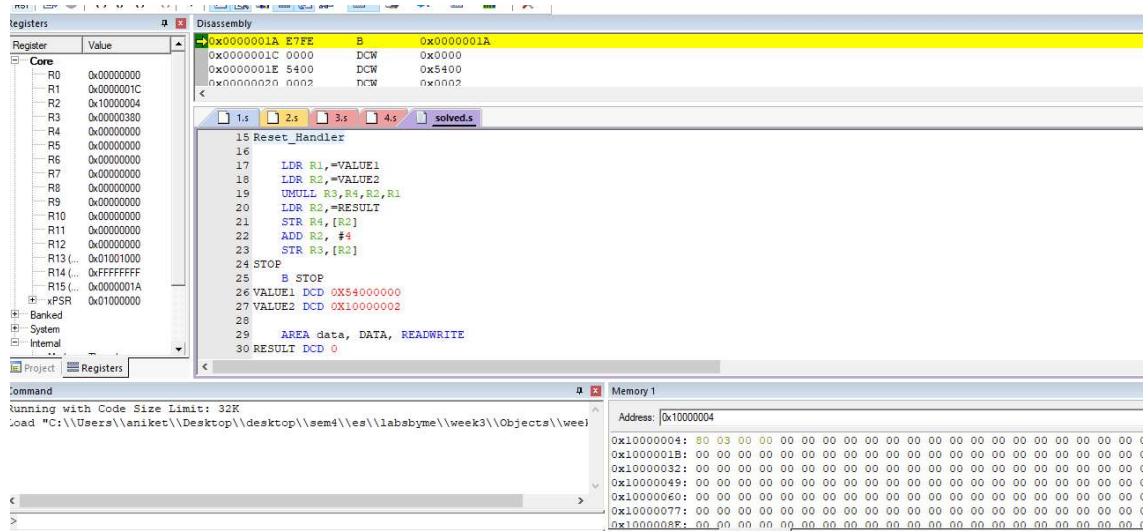
VALUE1 DCD 0X54000000

VALUE2 DCD 0X10000002

AREA data, DATA, READWRITE

RESULT DCD 0

END



Excercises

1. AREA RESET, DATA, READONLY

EXPORT __Vectors

__Vectors

DCD 0x1001000

DCD Reset_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset_Handler

Reset_Handler

LDR R3,=N1

LDR R4,=N2

LDR R5,=result

LDR R0,[R3] ;divident

LDR R1,[R4] ;divisor

MOV R2,#0 ; quotient

L1 CMP R0,R1 ;Compare R0 with R1 to see if less than 10

BLO STORE ;if R0 < R1 jump to finish

SUB R0,R0,R1 ;R0 = R0 - R1 (division by subtraction)

ADD R2,R2,#1 ;R2 = R2 + 1 (quotient is incremented)

B L1 ;

STORE STR R2,[R5,#4]

STR R0,[R5]

END

The screenshot displays the Immunity Debugger interface. The top pane shows the assembly code for a function, with the following instructions visible:

```

15  LDR R3,=N1
16  LDR R4,=N2
17  LDR R5,=result
18  LDR R0,[R3];divident
19
20  LDR R1,[R4];divisor
21  MOV R2,#0; quotient
22  LI CMP R0,R1;Compare R0 with R1 to see if less than 10
23  BLO STORE;if R0 < R1; jump to finish
24  SUB R0,R0,R1;R0 = R0 - R1 (division by subtraction)
25  ADD R2,R2,#1;R2 = R2 + 1 (quotient is incremented)
26  B LI;
27 STORE STR R2,[R5,#4]
28  STR R0,[R5]
29
30 STOP

```

The bottom pane shows the memory dump for address 0x10000000, displaying a series of zeroed-out memory locations.

At the bottom of the window, there is a status bar with the text: "ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE".

2.

AREA RESET, DATA, READONLY

EXPORT __Vectors

__Vectors

DCD 0x1001000

DCD Reset_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset_Handler

Reset_Handler

LDR R0,=N1

LDR R1,=result

LDR R3,[R0]

MOV R4,#0

L1 CMP R3,#0

BEQ STORE

ADD R4,R3

SUB R3,#1

B L1;

STORE STR R4,[R1]

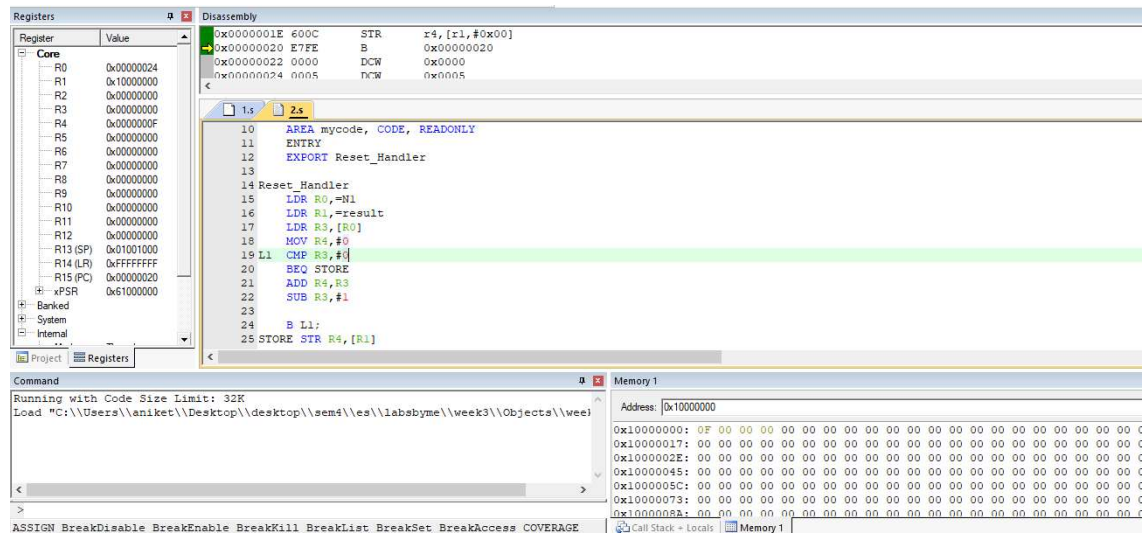
STOP

B STOP

N1 DCD 5

result DCD 0

END



3.

```
AREA RESET,DATA,READONLY
```

```
EXPORT __Vectors
```

__Vectors

```
DCD 0X10001000
```

```
DCD Reset_Handler
```

```
AREA mycode,CODE,READONLY
```

```
ENTRY
```

```
EXPORT Reset_Handler
```

Reset_Handler

```
LDR R0, =N1
```

```
LDR R1, =N2
```

```
LDR R2, [R0]
```

```
LDR R3, [R1]
```

GCDT

```
CMP R2, R3;three cases greater ,lower,equal
```

```
SUBLT R3, R3, R2 ;lower
```

```
SUBGT R2, R2, R3;greater
```

```
BNE GCDT;equal
```

STRGCD

```
LDR R4, =GCD
```

```
STR R2, [R4]
```

LCMT

```
LDR R5, [R0]
```

```
LDR R6, [R1]
MOV R4, #0
MUL R7, R5, R6
```

DIVISION

```
SUB R7, R7, R2
ADD R4, R4, #1
CMP R7, R2
BGE DIVISION
```

STRLCM

```
LDR R8, =LCM
STR R4, [R8]
```

STOP B STOP

```
N1 DCD 10
```

```
N2 DCD 15
```

```
AREA mydata,DATA,READWRITE
```

```
GCD DCD 0
```

```
LCM DCD 0
```

```
END
```


Register	Value
Core	
R0	0x00000044
R1	0x00000048
R2	0x00000005
R3	0x00000005
R4	0x0000001E
R5	0x0000000A
R6	0x0000000F
R7	0x00000000
R8	0x10000004
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13	0x10001000
R14	0xFFFFFFFF
R15	0x00000040
xPSR	0x81000000
Banked	
System	
Internal	

0x00000040 E7FE B 0x00000040

0x00000042 0000 DCW 0x0000

0x00000044 000A DCW 0x000A

0x00000046 0000 DCW 0x0000

1.s2.s3.s

30 LDR R6, [R1]

31 MOV R4, #0

32 MUL R7, R5, R6

33

34 DIVISION

35 SUB R7, R7, R2

36 ADD R4, R4, #1

37 CMP R7, R2

38 BGE DIVISION

39

40 STRLCM

41 LDR R8, =LCM

42 STR R4, [R8]

43

44 STOP B STOP

45 N1 DCD 10

Command

Running with Code Size Limit: 32K

Load "C:\Users\aniket\Desktop\desktop\sem4\es\labsbyme\week3\Objects\wee

Memory 1

Address: 0x10000004

0x10000004: 1E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000049: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000077: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000008F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

4.

AREA RESET, DATA, READONLY

EXPORT __Vectors

__Vectors

DCD 0x10001000

DCD Reset_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset_Handler

Reset_Handler

LDR R6, =DST

MOV R2, #0

LDR R0, =SRC

LDR R1, [R0]

UP CMP R1, #0XA;we have two cases

BCC STORE;case 1 the hexadecimal digit<10

SUB R1, #0XA;case 2 hexadecimal digit>10 so we find the ones digit and the tens digit

ADD R2, #01

B UP

STORE

ADD R1, #0X30

STRB R1, [R6], #1

MOV R1, R2

MOV R2, #0

CMP R1, #0XA

BCS UP

ADD R1, #0X30

STRB R1, [R6]

STOP B STOP

SRC DCD 0X9

AREA mydata, DATA, READWRITE

DST DCD 0

END

The screenshot displays a disassembler interface with three main panes:

- Registers:** A table showing the state of various registers. R0 through R15 are listed with values ranging from 0x00000000 to 0xFFFFFFFF. Special registers like xPSR, Banked, System, and Internal are also shown.
- Disassembly:** A list of assembly instructions. The current instruction is highlighted in green: `AREA mycode, CODE, READONLY`. Other instructions include `EXPORT __Vectors`, `DCD 0x10001000`, `DCD Reset_Handler`, `ALIGN`, `ENTRY`, `EXPORT Reset_Handler`, `Reset_Handler`, `LDR R6, =DST`, `MOV R2, #0`, `LDR R0, =SRC`, `LDR R1, [R0]`, `CMF R1, #0xA; we have two cases`, and `BCC STORE; case 1 the hexadecimal digit < 10`.
- Memory:** A view of memory at address 0x10000001, showing a sequence of bytes (00 00 00 00) repeated across multiple lines.

At the bottom, a command window shows the execution status: "Running with Code Size Limit: 32K" and "Load *C:\\Users\\laniket\\Desktop\\desktop\\sem4\\es\\labs\\week3\\Objects\\wee".

5.

AREA RESET, DATA, READONLY

EXPORT __Vectors

__Vectors

DCD 0x10001000

DCD Reset_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset_Handler

Reset_Handler

LDR R0, =SRC

LDR R2, =DST

MOV R5, #2

UP LDRB R1, [R0], #1

LDRB R4, [R0], #1

LSL R1, #4

ADD R3, R1, R4

STRB R3, [R2], #1

SUBS R5, #1

BNE UP

STOP

B STOP

SRC DCB 1,2

AREA mydata, DATA, READWRITE

DST DCB 0

END

The screenshot shows the Keil MDK-ARM IDE interface. The main window is the Disassembly window, which displays the assembly code for the ARM7TDMI-S processor. The code is organized into sections: Core, Banked, System, and Internal. The Core section contains the main assembly code, which includes instructions like LDR, LDRB, MOV, LDRB, LSL, ADD, STRB, SUBS, BNE, STOP, SRC, AREA, DST, and END. The Banked section contains the Banked registers. The System section contains the System registers. The Internal section contains the Internal registers. The Disassembly window also shows the Code Size Limit, which is set to 32K. The Memory window shows the memory address 0x10000000.

Registers

Register	Value
R0	0x0000002E
R1	0x0000002A0
R2	0x10000002
R3	0x0000002A0
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (..)	0x10001000
R14 (..)	0xFFFFFFF
R15 (..)	0x00000028
PSR	0x61000000

Banked

System

Internal

Project

Registers

Disassembly

0x00000029 E7FE B 0x00000029

0x0000002A 0201 DCW 0x0201

0x0000002C 002A DCW 0x002A

0x0000002F 0000 DCW 0x0000

1.5 2.5 3.5 4.5 solved.s 5.5

12 LDR R0, =SRC

13 LDR R2, =DST

14 MOV R5, #2

15 UP LDRB R1, [R0], #1

16 LDRB R4, [R0], #1

17 LSL R1, #4

18 ADD R3, R1, R4

19 STRB R3, [R2], #1

20 SUBS R5, #1

21 BNE UP

22 STOP

23 B STOP

24 SRC DCB 1,2

25 AREA mydata, DATA, READWRITE

26 DST DCB 0

27 END

ommand

unning with Code Size Limit: 32K

oad "C:\\Users\\aniket\\Desktop\\desktop\\sem4\\es\\labsbyme\\week3\\Objects\\wee

Address: 0x10000000

0x10000000: 12 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000001: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000003: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000007: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000011: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000014: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000016: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000019: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x1000001F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000021: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000025: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000026: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000027: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x10000029: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x100000