



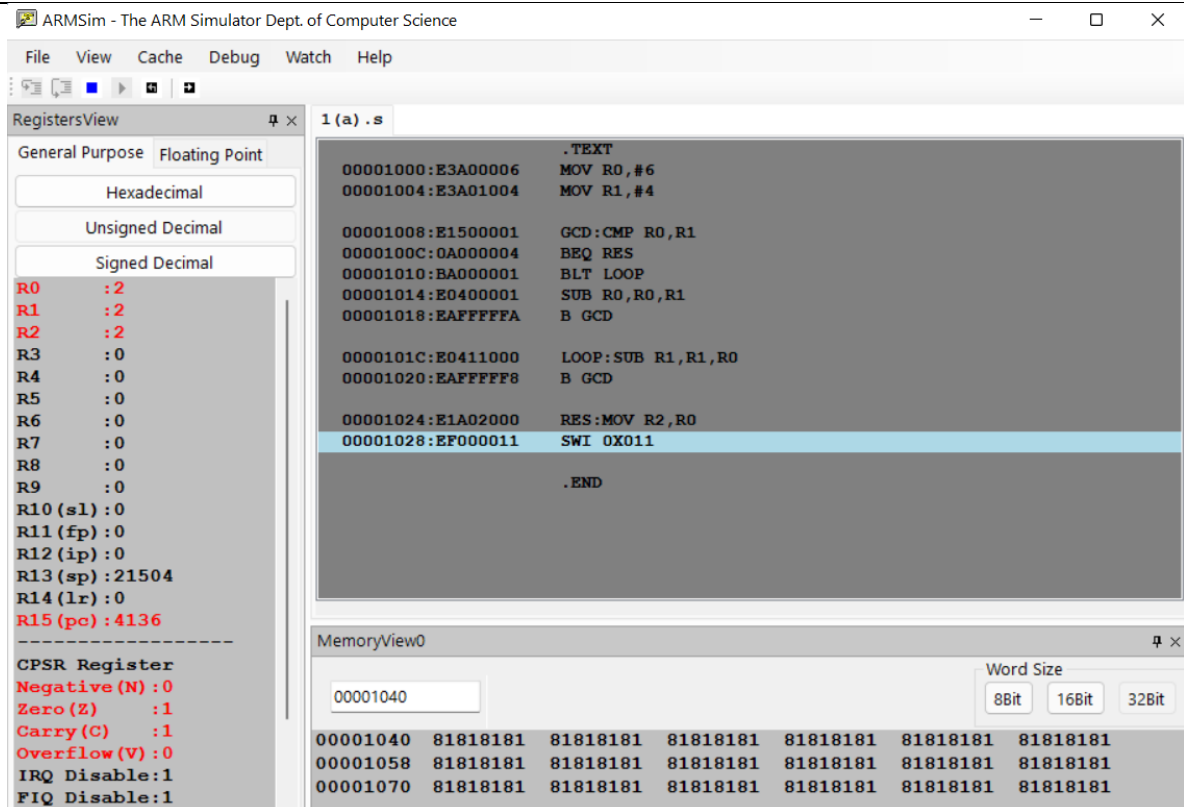
Department of Computer Science & Engineering
Microprocessor & Computer Architecture
MPCA–Laboratory/Assignment/Hands–on/Project
UE20CS252

Vrushank G

PES1UG20CS516

SEC-I

Sl. No.	Programs
Week No. 3	<p>1. Write a program in ARM7TDMI–ISA to find GCD of two numbers. a. Assume operands to be in the CPU registers.</p> <pre>.TEXT MOV R0,#6 MOV R1,#4 GCD: CMP R0,R1 BEQ RES BLT LOOP SUB R0,R0,R1 B GCD LOOP: SUB R1,R1,R0 B GCD RES: MOV R2,R0 SWI 0X011 .END</pre> <p>Output:</p>



b. Assume operands in the memory locations.

```
.TEXT
LDR R0,=A
LDR R1,=B
LDR R4,=C

LDR R2,[R0]
LDR R3,[R1]

L: CMP R2,R3
   BEQ RES
   BLT LOOP
   SUB R2,R2,R3
   B L

LOOP: SUB R3,R3,R2
      B L

RES: STR R2,[R4]

SWI 0X011
```

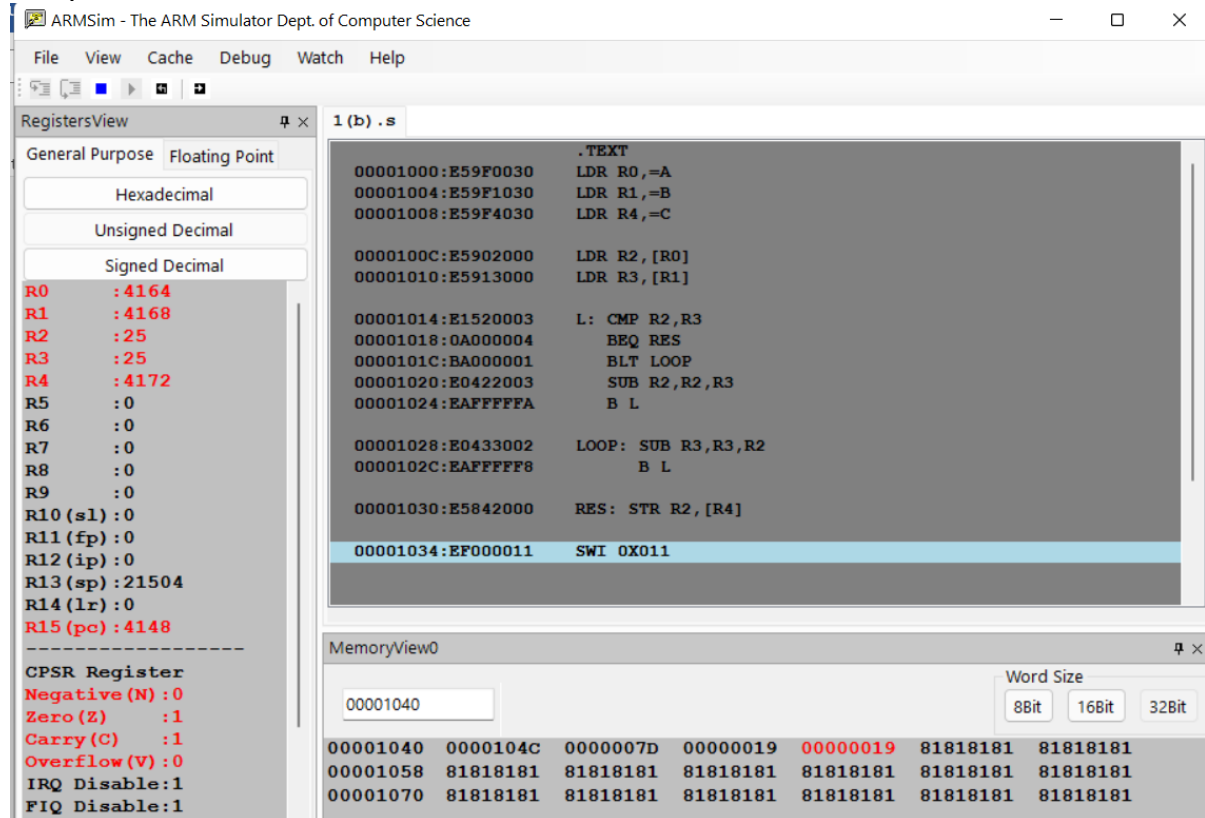
```
.DATA
```

```
A: .WORD 125
```

```
B: .WORD 25
```

```
C: .WORD
```

Output:



2. Write a program in ARM7TDMI-ISA to find the sum of N data items in the memory. Store the result in the memory location.
 - a. Use Pre-indexing addressing mode.

```
.DATA
```

```
A: .WORD 10,20,30,40,50,60,70,80,90,100
```

```
SUM: .WORD 0
```

```
.TEXT
```

```
LDR R0,=A
```

```
LDR R1,=SUM
```

```
MOV R2,#0
```

```
MOV R3,#4
```

```
MOV R4,#1
```

```
SUB R0,R0,#4
```

```
LOOP: LDR R5,[R0,R3] ;R5<-MEM[R0+R3]
```

```
ADD R2,R2,R5
```

```

ADD R3,R3,#4
ADD R4,R4,#1
CMP R4,#11
BNE LOOP

```

```

STR R2,[R1]
SWI 0X011

```

Output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 2 (a) .s

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4156
R1 : 4200
R2 : 550
R3 : 44
R4 : 11
R5 : 100
R6 : 0
R7 : 0
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 4148

CPSR Register
Negative (N) : 0
Zero (Z) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable: 1
FIQ Disable: 1

```

.DATA
00001040: A: .WORD 10,20,30,40,50,60,70,80,90,100
00001068: SUM: .WORD 0

.TEXT
00001000:E3A00D41 LDR R0,=A
00001004:E59F102C LDR R1,=SUM

00001008:E3A02000 MOV R2,#0
0000100C:E3A03004 MOV R3,#4
00001010:E3A04001 MOV R4,#1
00001014:E2400004 SUB R0,R0,#4

00001018:E7905003 LOOP: LDR R5,[R0,R3] ;R5<--MEM[R0+R3],R0<--R0+R3
0000101C:E0822005 ADD R2,R2,R5
00001020:E2833004 ADD R3,R3,#4
00001024:E2844001 ADD R4,R4,#1
00001028:E354000B CMP R4,#11
0000102C:1AFFFFF9 BNE LOOP

00001030:E5812000 STR R2,[R1]
SWI 0X011

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

00001040	0000000A	00000014	0000001E	00000028	00000032	0000003C
00001058	00000046	00000050	0000005A	00000064	00000226	81818181
00001070	81818181	81818181	81818181	81818181	81818181	81818181

b. Use Post-indexing addressing mode.

```

.DATA
A: .WORD 10,20,30,40,50,60,70,80,90,100
SUM: .WORD 0

.TEXT
LDR R0,=A
LDR R1,=SUM

MOV R2,#0
MOV R3,#4
MOV R4,#1

```

```

LOOP: LDR R5,[R0],#4
      ADD R2,R2,R5
      ADD R4,R4,#1
      CMP R4,#11
      BNE LOOP

STR R2,[R1]

.END

```

Output:

The screenshot displays the ARMSim - The ARM Simulator interface. The main window is titled '2 (b) .s' and shows the following assembly code:

```

.DATA
00001034: A: .WORD 10,20,30,40,50,60,70,80,90,100
0000105C: SUM: .WORD 0

.TEXT
00001000:E59F0024 LDR R0,=A
00001004:E59F1024 LDR R1,=SUM

00001008:E3A02000 MOV R2,#0
0000100C:E3A03004 MOV R3,#4
00001010:E3A04001 MOV R4,#1

00001014:E4905004 LOOP: LDR R5,[R0],#4 ;R5<-MEM[R0+R3],R0<-R0+R3
00001018:E0822005 ADD R2,R2,R5
0000101C:E2844001 ADD R4,R4,#1
00001020:E354000B CMP R4,#11
00001024:1AFFFFFA BNE LOOP

00001028:E5812000 STR R2,[R1]

0000102C:00001034 .END
00001030:0000105C

```

The RegistersView panel on the left shows the following register values:

Register	Value
R0	:0
R1	:0
R2	:550
R3	:4
R4	:11
R5	:100
R6	:0
R7	:0
R8	:0
R9	:0
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:21504
R14 (lr)	:0
R15 (pc)	:70656

The CPSR Register shows the following status flags:

Flag	Value
Negative (N)	:0
Zero (Z)	:1
Carry (C)	:1
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1

The MemoryView panel at the bottom shows the memory layout starting at address 00001040:

Address	Value
00001040	00000028
00001058	00000064
00001070	81818181

c. Use Auto-indexing addressing mode.

```

.DATA
A: .WORD 10,20,30,40,50,60,70,80,90,100
SUM: .WORD 0

.TEXT
LDR R0,=A
LDR R1,=SUM

MOV R2,#0
MOV R3,#4

```

```

MOV R4,#1
SUB R0,R0,#4

LOOP: LDR R5,[R0,R3]! ;R5<-MEM[R0+R3],R0<-R0+R3
      ADD R2,R2,R5
      ADD R4,R4,#1
      CMP R4,#11
      BNE LOOP

STR R2,[R1]

.END

```

Output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0
R1 : 12
R2 : 550
R3 : 4
R4 : 11
R5 : 100
R6 : 0
R7 : 0
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 70656

CPSR Register

Negative (N) : 0
Zero (Z) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1

2 (c) .s

.DATA

00001038: A: .WORD 10,20,30,40,50,60,70,80,90,100
00001060: SUM: .WORD 0

.TEXT

00001000:E59F0028 LDR R0,=A
00001004:E59F1028 LDR R1,=SUM

00001008:E3A02000 MOV R2,#0
0000100C:E3A03004 MOV R3,#4
00001010:E3A04001 MOV R4,#1
00001014:E2400004 SUB R0,R0,#4

00001018:E7B05003 LOOP: LDR R5,[R0,R3]! ;R5<-MEM[R0+R3],R0<-R0+R3
0000101C:E0822005 ADD R2,R2,R5
00001020:E2844001 ADD R4,R4,#1
00001024:E354000B CMP R4,#11
00001028:1AFFFFFA BNE LOOP

0000102C:E5812000 STR R2,[R1]

00001030:00001038 .END

MemoryView0

Word Size 8Bit 16Bit 32Bit

00001040

00001040	0000001E	00000028	00000032	0000003C	00000046	00000050
00001058	0000005A	00000064	00000226	81818181	81818181	81818181
00001070	81818181	81818181	81818181	81818181	81818181	81818181

3. Write a program in ARM7TDMI-ISA to find the sum of N data items at alternate [odd or even positions] locations in the memory. Store the result in the memory location.
 - a. Use Pre-indexing addressing mode.

```

;here we are finding the sum of n numbers at odd locations
.DATA
A: .WORD 10,20,30,40,50,60

```

```

SUM: .WORD 0

.TEXT
MOV R2,#0
LDR R1,=A
LDR R3,=SUM
MOV R6,#0
SUB R1,R1,#8 ;change #8 to #4 to find the sum of numbers at even positions

LOOP:
LDR R4,[R1,#8]
ADD R1,R1,#8
ADD R2,R2,R4
ADD R6,R6,#2
CMP R6,#6
BNE LOOP
STR R2,[R3]
.END

```

Output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 3(a).s

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0
R1 : 0
R2 : 90
R3 : 4176
R4 : 50
R5 : 0
R6 : 6
R7 : 0
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 70656

CPSR Register

Negative (N) : 0
Zero (Z) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable: 1
FIQ Disable: 1

```

;here we are finding the sum of n numbers at odd location.
.DATA
A: .WORD 10,20,30,40,50,60
SUM: .WORD 0

.TEXT
00001000:E3A02000 MOV R2,#0
00001004:E59F1024 LDR R1,=A
00001008:E59F3024 LDR R3,=SUM
0000100C:E3A06000 MOV R6,#0
00001010:E2411008 SUB R1,R1,#8 ;change #8 to #4 to find the sum of numbers

00001014: LOOP:
00001014:E5914008 LDR R4,[R1,#8]
00001018:E2811008 ADD R1,R1,#8
0000101C:E0822004 ADD R2,R2,R4
00001020:E2866002 ADD R6,R6,#2
00001024:E3560006 CMP R6,#6
00001028:1AFFFFF9 BNE LOOP
0000102C:E5832000 STR R2,[R3]
00001030:00001038 .END

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

00001040

00001040	0000001E	00000028	00000032	0000003C	0000005A	81818181
00001058	81818181	81818181	81818181	81818181	81818181	81818181
00001070	81818181	81818181	81818181	81818181	81818181	81818181

b. Use Auto-indexing addressing mode.

```

;here we are finding the sum of n numbers at odd locations
.DATA
A: .WORD 10,20,30,40,50,60
SUM: .WORD 0

.TEXT
MOV R2,#0
LDR R1,=A
LDR R3,=SUM
MOV R6,#0

LOOP:
LDR R4,[R1],#4
ADD R2,R2,R4
ADD R6,R6,#2
CMP R6,#6
BNE LOOP
STR R2,[R3]
.END

```

Output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 3(b) .s

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0
R1 : 0
R2 : 60
R3 : 4168
R4 : 30
R5 : 0
R6 : 6
R7 : 0
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 70656

CPSR Register
Negative (N) : 0
Zero (Z) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable: 1
FIQ Disable: 1

```

;here we are finding the sum of n numbers at odd locations
.DATA
00001030: A: .WORD 10,20,30,40,50,60
00001048: SUM: .WORD 0

.TEXT
00001000:E3A02000 MOV R2,#0
00001004:E59F101C LDR R1,=A
00001008:E59F301C LDR R3,=SUM
0000100C:E3A06000 MOV R6,#0

00001010: LOOP:
00001010:E4914004 LDR R4,[R1],#4
00001014:E0822004 ADD R2,R2,R4
00001018:E2866002 ADD R6,R6,#2
0000101C:E3560006 CMP R6,#6
00001020:1AFFFFFA BNE LOOP
00001024:E5832000 STR R2,[R3]
00001028:00001030 .END
0000102C:00001048

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

00001040	00000032	0000003C	0000003C	81818181	81818181	81818181
00001058	81818181	81818181	81818181	81818181	81818181	81818181
00001070	81818181	81818181	81818181	81818181	81818181	81818181

c. Use Post-indexing addressing mode.


```

;here we are finding the sum of n numbers at odd locations
.DATA
A: .WORD 10,20,30,40,50,60
SUM: .WORD 0

.TEXT
MOV R2,#0
LDR R1,=A
LDR R3,=SUM
MOV R6,#0
SUB R1,R1,#8

LOOP:LDR R4,[R1,#8]!
ADD R2,R2,R4
ADD R6,R6,#2
CMP R6,#6
BNE LOOP
STR R2,[R3]
.END

```

Output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 3 (c) .s

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0
R1 : 0
R2 : 0
R3 : 4172
R4 : 50
R5 : 0
R6 : 6
R7 : 0
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 70656

CPSR Register
Negative (N) : 0
Zero (Z) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable: 1
FIQ Disable: 1

```

;here we are finding the sum of n numbers at odd locations
.DATA
00001034: A: .WORD 10,20,30,40,50,60
0000104C: SUM: .WORD 0

.TEXT
00001000:E3A02000 MOV R2,#0
00001004:E59F1020 LDR R1,=A
00001008:E59F3020 LDR R3,=SUM
0000100C:E3A06000 MOV R6,#0
00001010:E2411008 SUB R1,R1,#8

00001014:E5B14008 LOOP:LDR R4,[R1,#8]!
00001018:E0822004 ADD R2,R2,R4
0000101C:E2866002 ADD R6,R6,#2
00001020:E3560006 CMP R6,#6
00001024:1AFFFFFA BNE LOOP
00001028:E5832000 STR R2,[R3]
0000102C:00001034 .END
00001030:0000104C

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

00001040	00000028	00000032	0000003C	0000005A	81818181	81818181
00001058	81818181	81818181	81818181	81818181	81818181	81818181
00001070	81818181	81818181	81818181	81818181	81818181	81818181

4. Write a program in ARM7TDMI–ISA to search for an element in an array. Store 00 if the search is unsuccessful and 01 if the search is successful in the register.

a. Use Linear Search Technique.

```
.TEXT

LDR R0,=A
LDR R1,=KEY

LDR R3,[R1]
MOV R4,#1

L:LDR R2,[R0]
  CMP R2,R3
  BEQ RES
  ADD R0,R0,#4
  ADD R4,R4,#1
  CMP R4,#6
  BNE L

MOV R5,#0
B LOOP

RES:MOV R5,#01

LOOP: SWI 0X011

.DATA
A: .WORD 10,20,30,40,50
KEY: .WORD 60
```

Output:

