PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture

MPCA–Laboratory/Assignment/Hands–on/Project

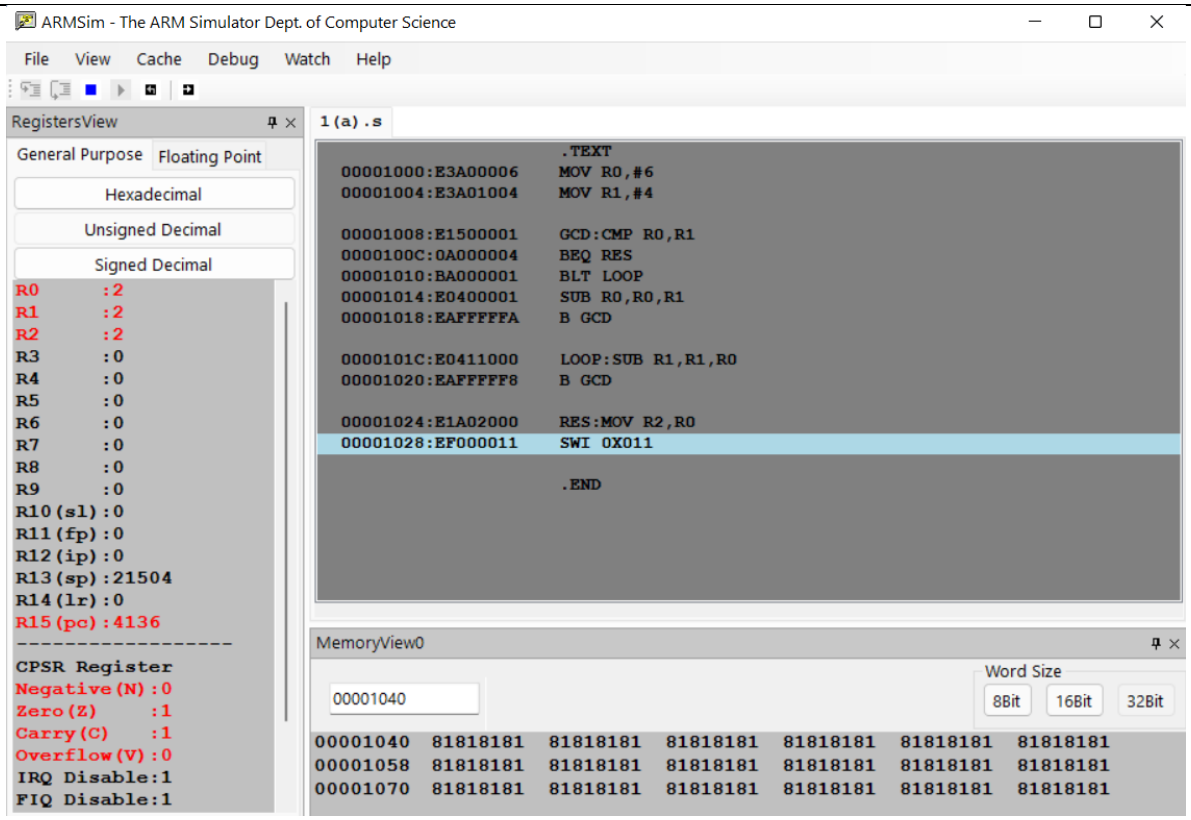
UE20CS252

Vrushank G **PES1UG20CS516** **SEC-I**

Sl. No.	Programs
Week No. 3	1. Write a program in ARM7TDMI–ISA to find GCD of two numbers. a. Assume operands to be in the CPU registers.

```
.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
```

Output:

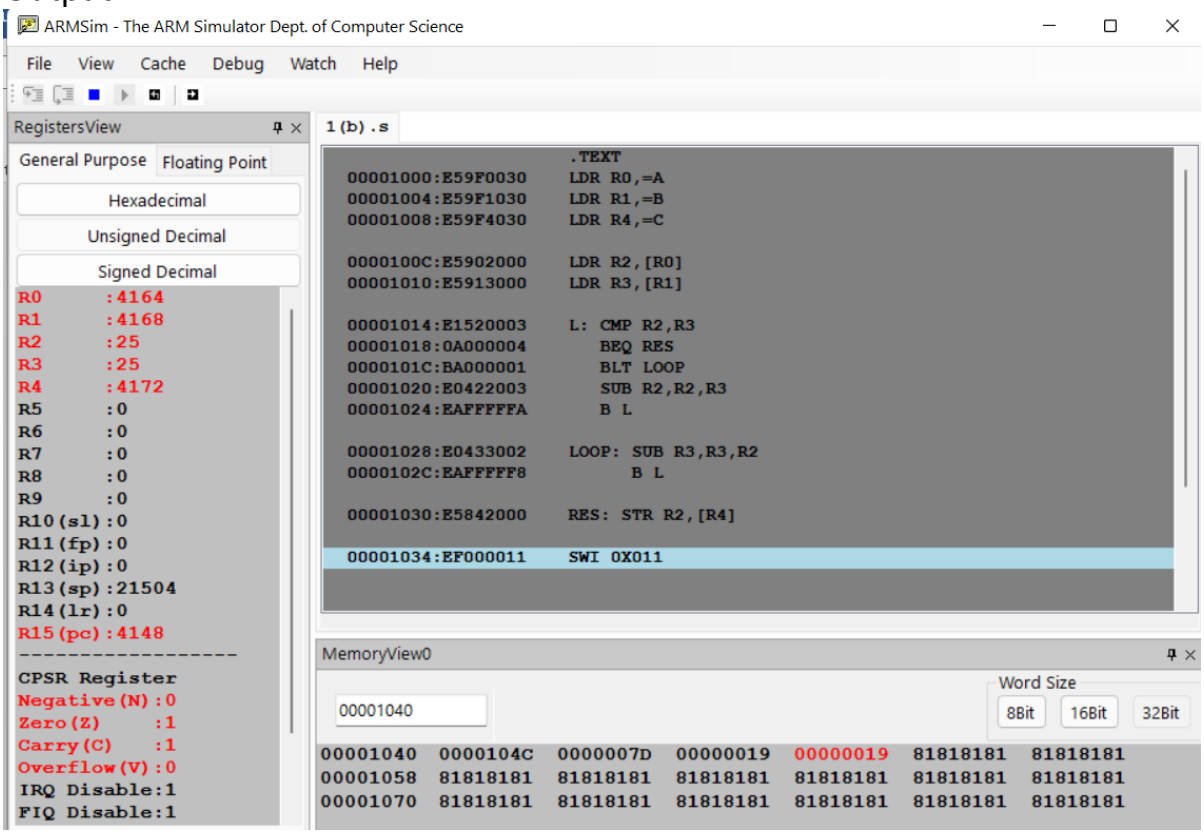


b. Assume operands in the memory locations.

```
.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:

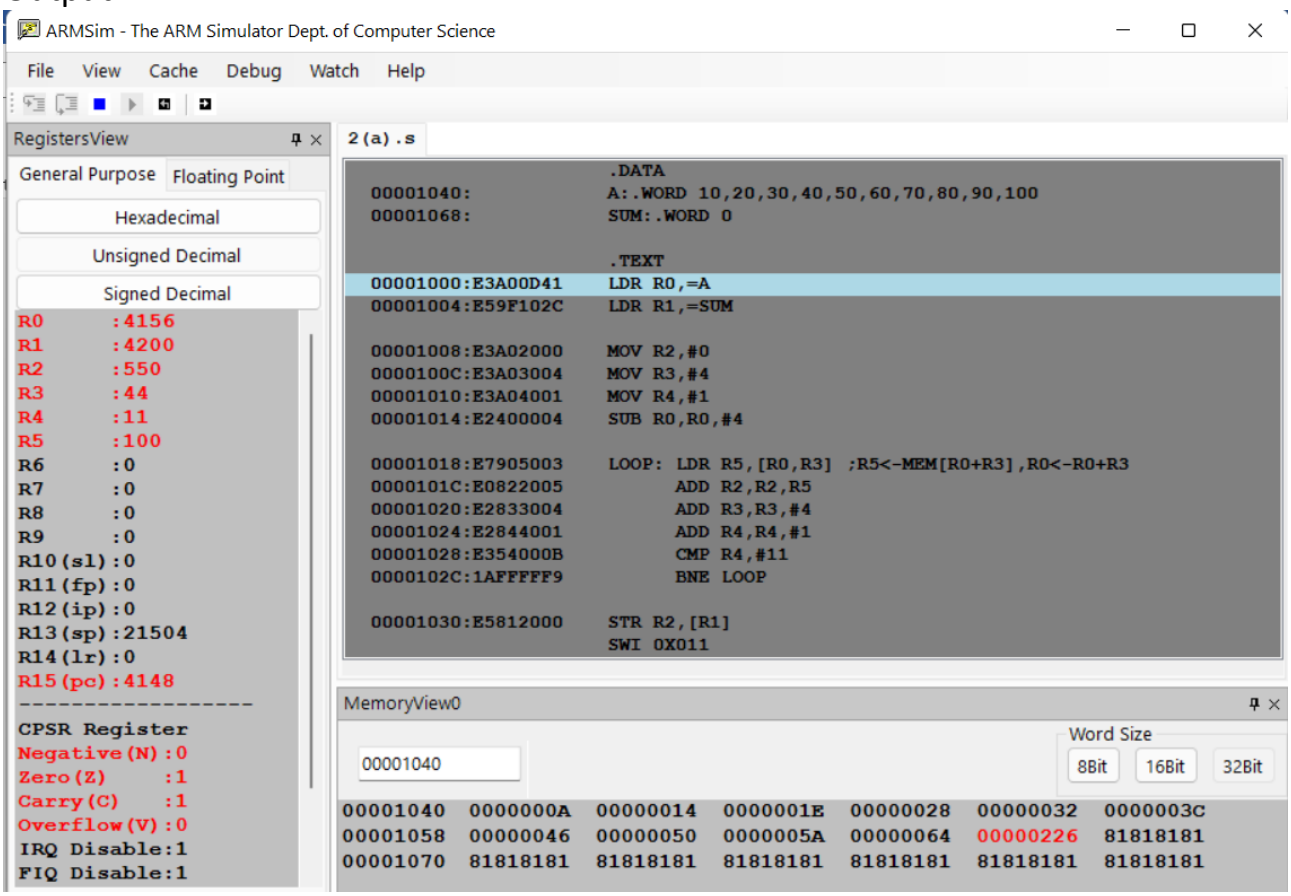


2. Write a program in ARM7TDMI–ISA to find the sum of N data items in the memory. Store the result in the memory location.
   a. Use Pre–indexing addressing mode.

```
.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:

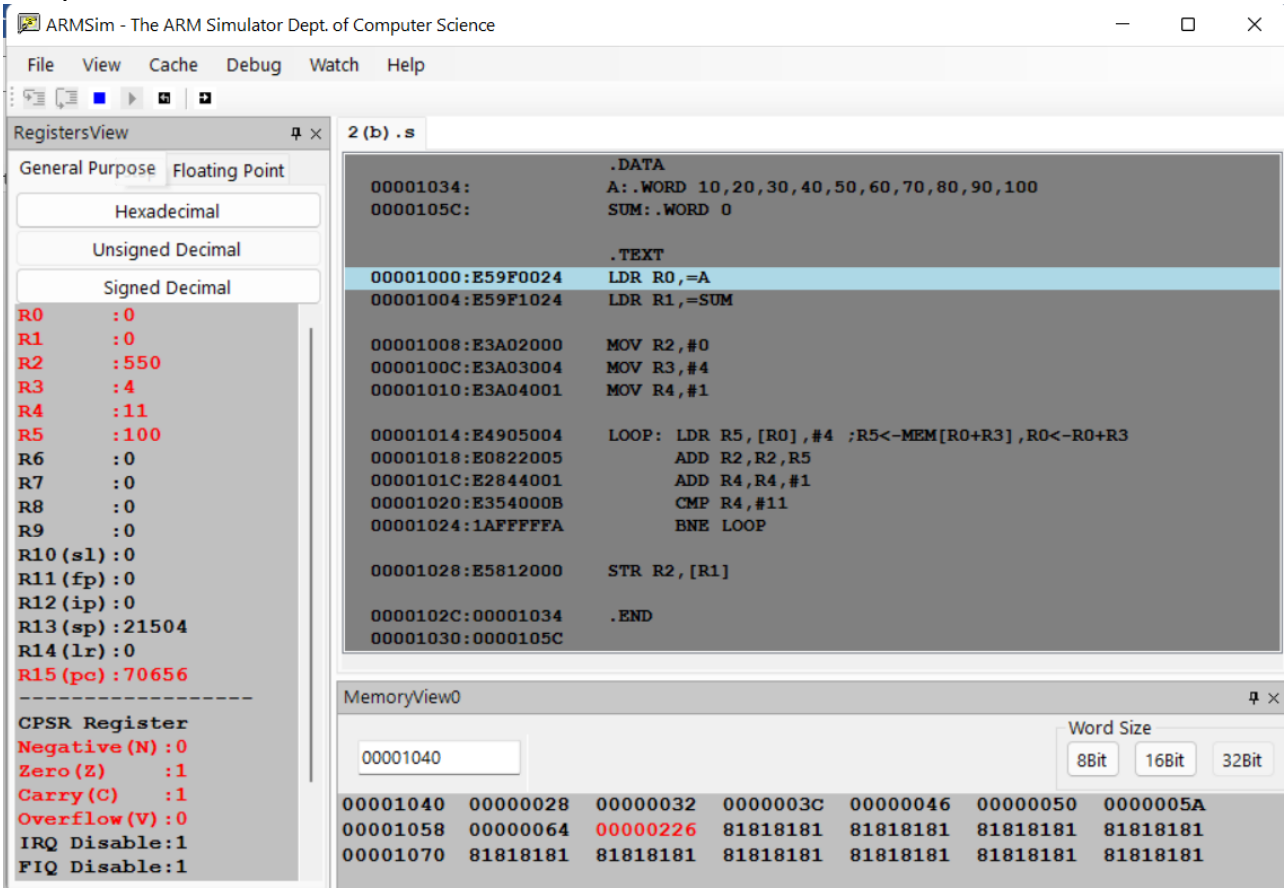


b. Use Post–indexing addressing mode.

```
.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:

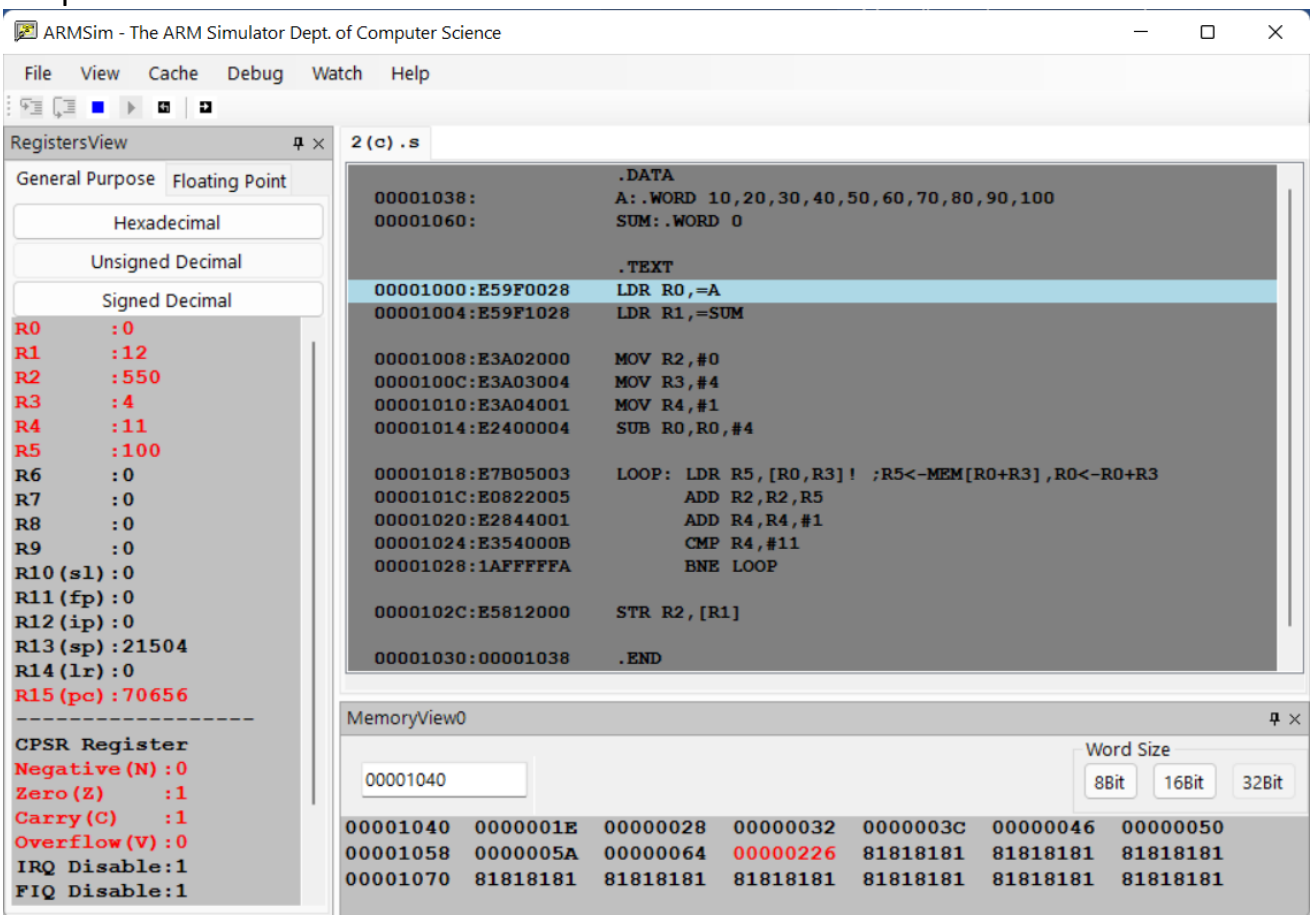


c. Use Auto–indexing addressing mode.

```
.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:

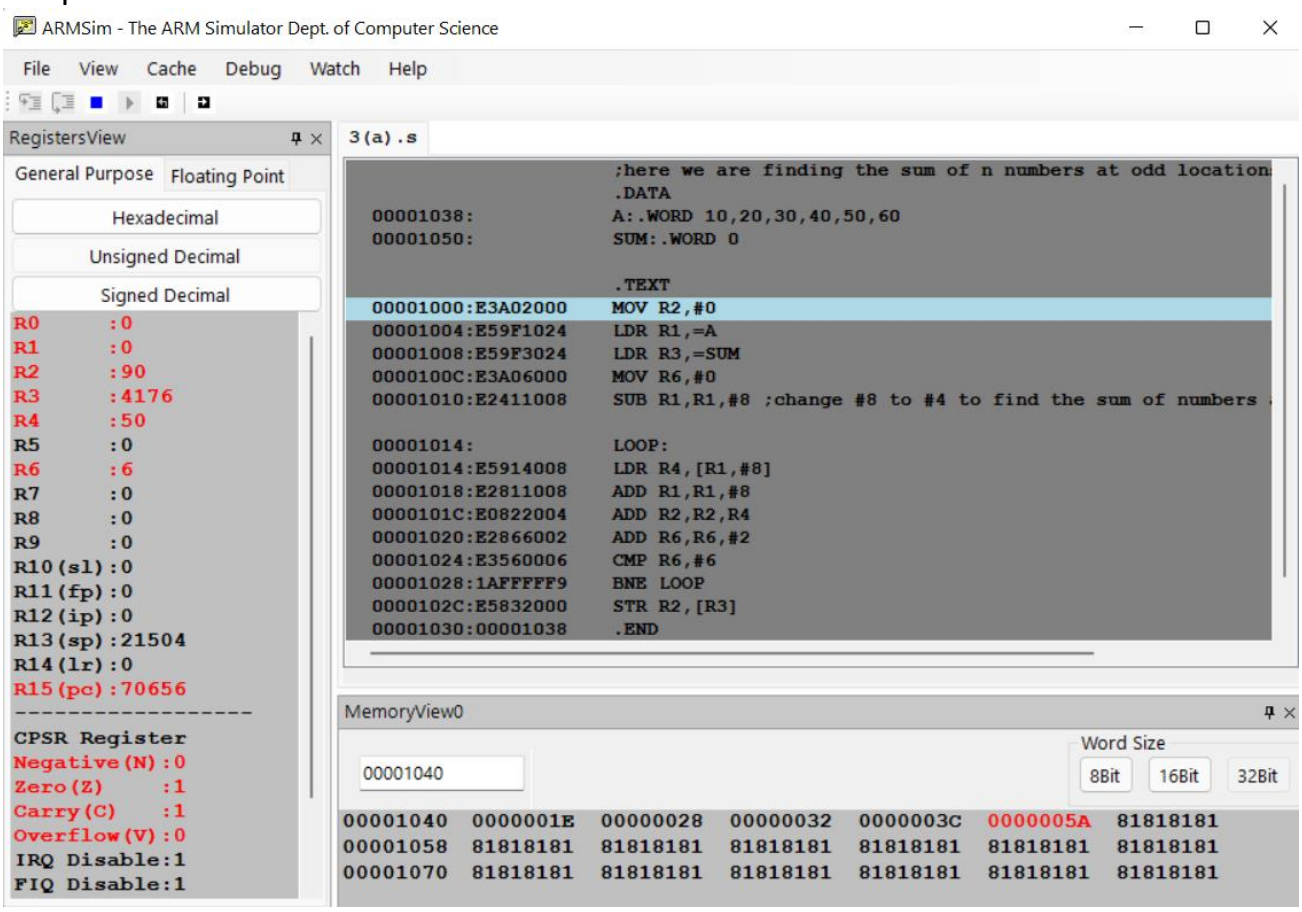


3. Write a program in ARM7TDMI–ISA to find the sum of N data items at alternate [**odd** or **even** positions] locations in the memory. Store the result in the memory location.
   a. Use Pre–indexing addressing mode.

```
;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
```

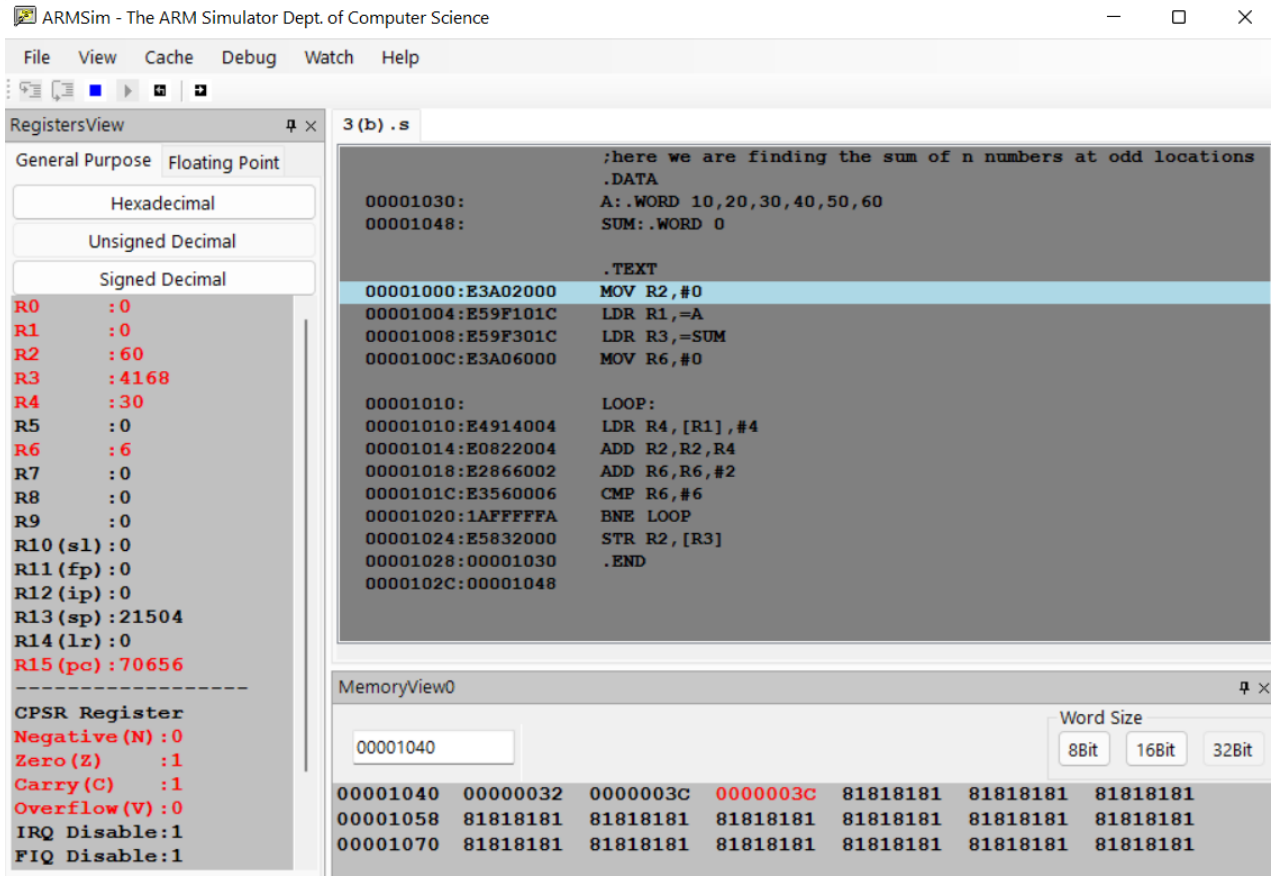
Output:



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
```

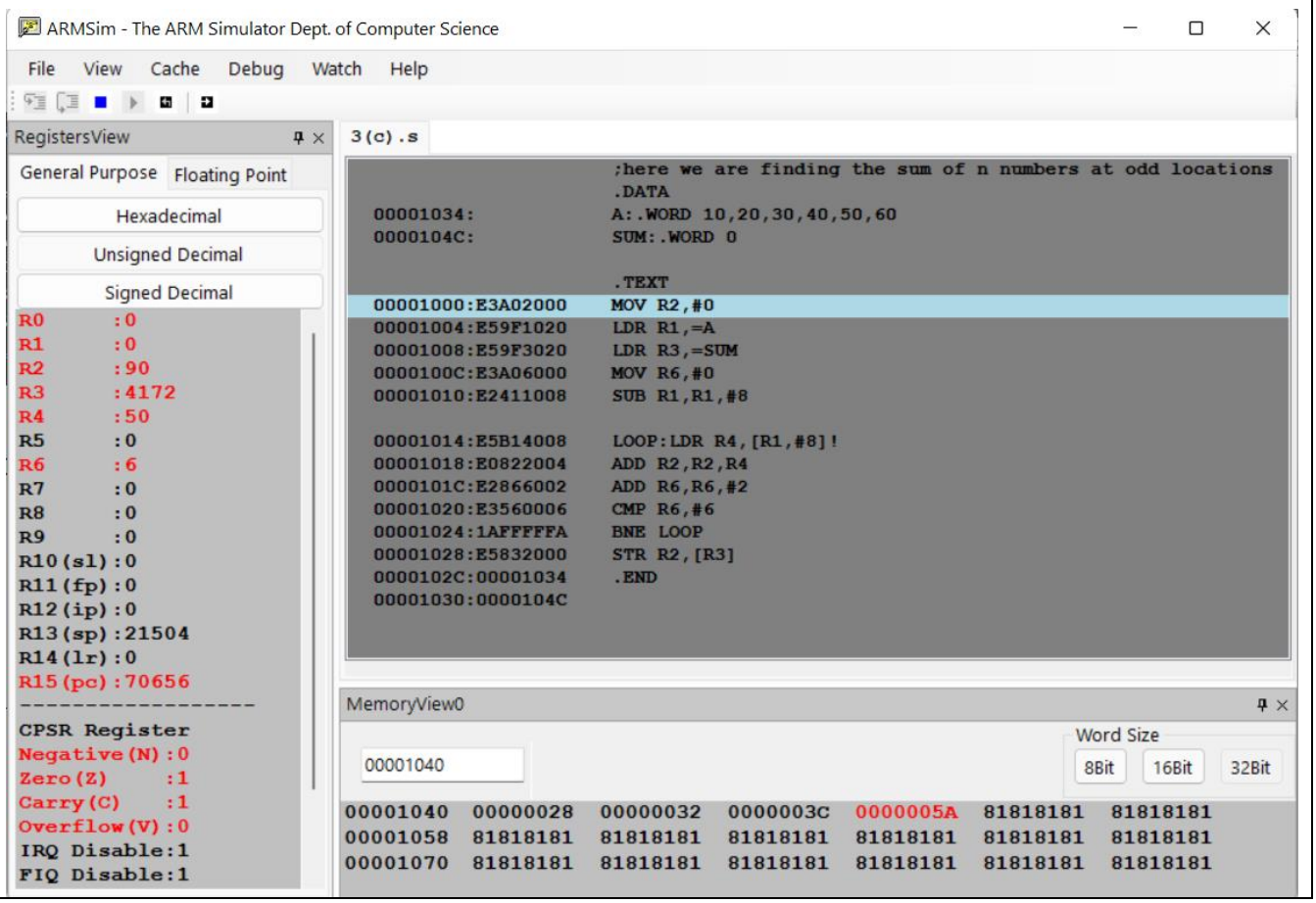
Output:



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
```

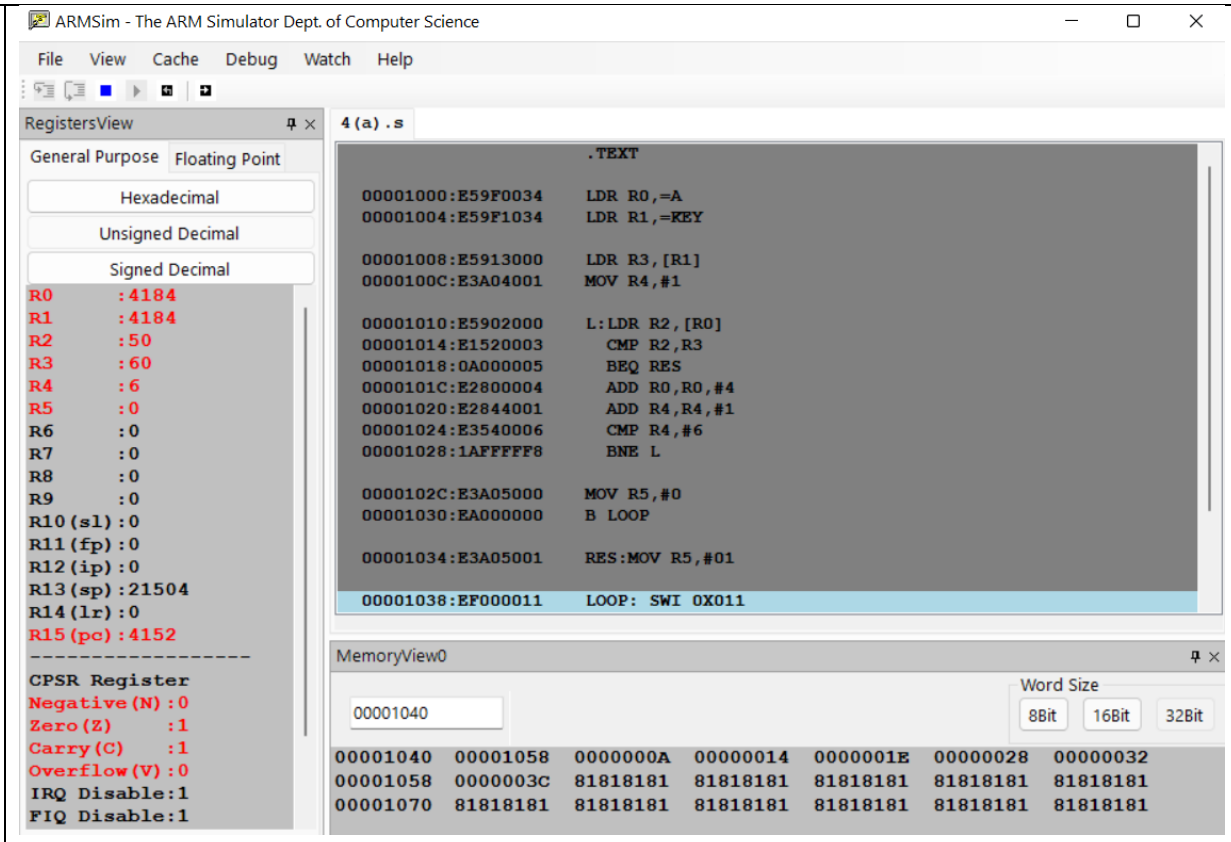
Output:



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:


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 :4184
R1 :4184
R2 :50
R3 :60
R4 :6
R5 :0
R6 :0
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4152
CPSR Register
Negative(N):0
Zero(Z) :1
Carry(C) :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
4(a).s
.TEXT
00001000:E59F0034 LDR R0,=A
00001004:E59F1034 LDR R1,=KEY
00001008:E5913000 LDR R3,[R1]
0000100C:E3A04001 MOV R4,#1
00001010:E5902000 L:LDR R2,[R0]
00001014:E1520003 CMP R2,R3
00001018:0A000005 BEQ RES
0000101C:E2800004 ADD R0,R0,#4
00001020:E2844001 ADD R4,R4,#1
00001024:E3540006 CMP R4,#6
00001028:1AFFFFF8 BNE L
0000102C:E3A05000 MOV R5,#0
00001030:EA000000 B LOOP
00001034:E3A05001 RES:MOV R5,#01
00001038:EF000011 LOOP: SWI 0X011
MemoryView0
00001040
Word Size
8Bit 16Bit 32Bit
00001040 00001058 0000000A 00000014 0000001E 00000028 00000032
00001058 0000003C 81818181 81818181 81818181 81818181 81818181
00001070 81818181 81818181 81818181 81818181 81818181 81818181