



Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

Lab 2 Programs

UE23CS251B

Name of Student: Pranav Hemanth

SRN: PES1UG23CS433

1

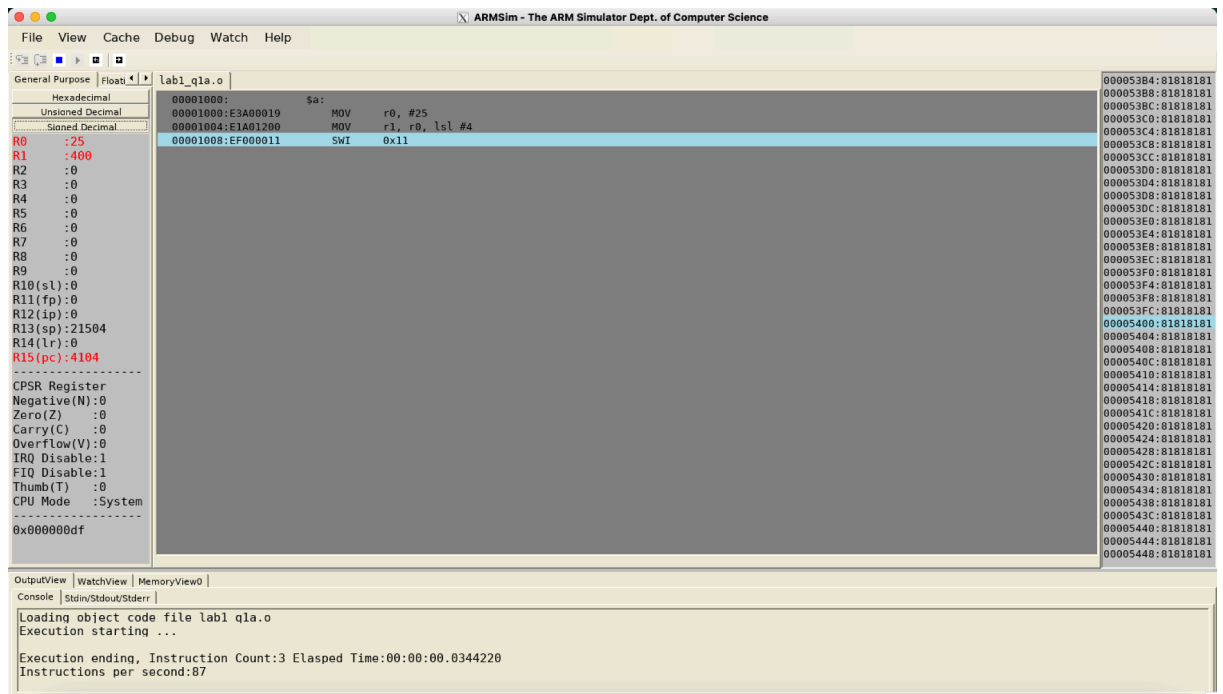
Write an ALP using ARM7TDMI to perform the multiplication of 16X25 without using MUL instructions.

(Hint: barrel shifter instructions.)

(Note : Any number can be considered as multiplier)

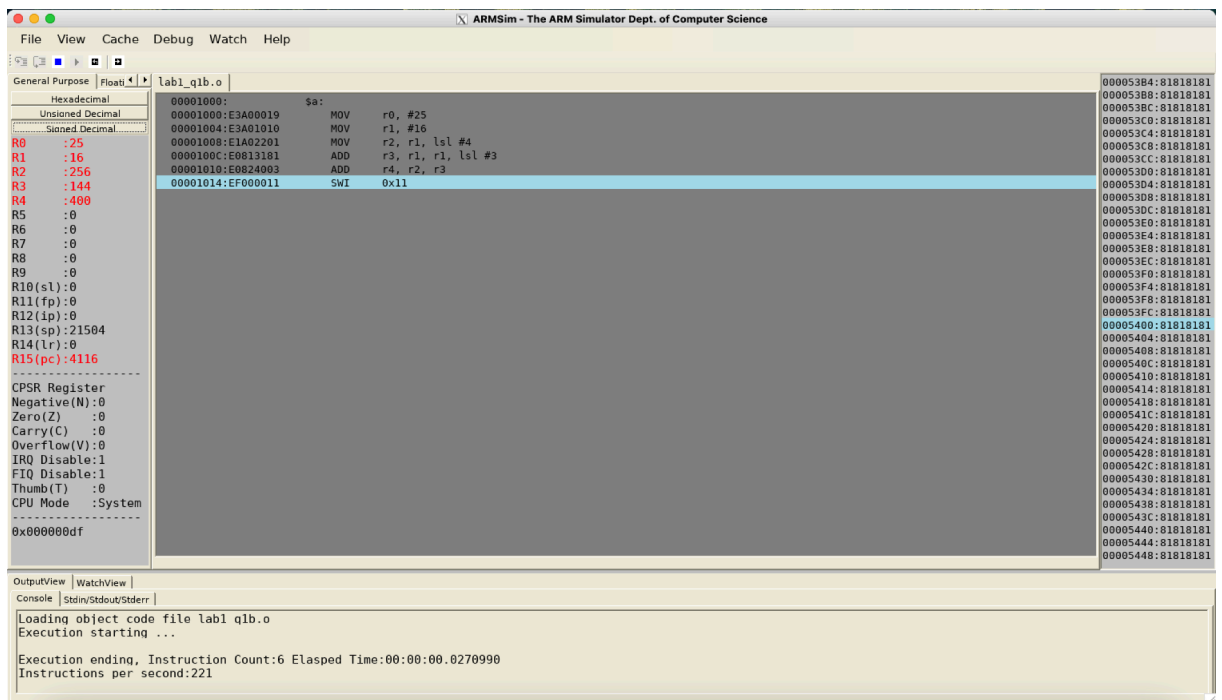
ARMSim_files > ASM lab1_q1a.s

```
1 @ Write an ALP using ARM7TDMI to perform multiplication of 16X25 without using mul instruction
2 @ Hint barrel shifter instructions
3 @ Note: any number
4
5 .text
6 MOV R0, #25
7 MOV R1, R0, LSL#4
8 SWI 0x011
9
```



ARMSim_files > *ASM* lab1_q1b.s

```
1  @ Write an ALP using ARM7TDMI to perform multiplication of 16X25 without using mul instruction
2  @ Hint barrel shifter instructions
3  @ Store n in R0 and result in R1
4  @ 16 is considered as multiplier
5
6  .text
7  MOV R0, #25
8  MOV R1, #16
9  MOV R2, R1, LSL#4
10 ADD R3, R1, R1, LSL#3
11 ADD R4, R2, R3
12 SWI 0x011
13
```



2

Write an ALP using ARM7TDMI to add only even numbers stored in memory location for a given set of numbers and store the sum in the memory location.

Array:. WORD 15,10,12,13,9,45,16,8,25,33

evensum:. WORD

ARMSim_files > `asm lab1_q2.s`

```
1  @ Write an ALP to add only even numbers stored in memory location for a given set of numbers and store sum in memory location
2  @ Array: .WORD 15, 10, 12, 13, 9, 45, 16, 8, 25, 33
3  @ evensum: .WORD
4
5  .data
6  array: .word 15, 10, 12, 13, 9, 45, 16, 8, 25, 33
7  evensum: .word 0
8
9  .text
10
11 init:
12     LDR R0, =array
13     LDR R1, =evensum
14     MOV R4, #0
15     MOV R5, #10
16
17 start:
18     CMP R5, #0
19     BNE loop
20     B end
21
22 loop:
23     LDR R3, [R0]
24     ADD R0, R0, #4
25     TST R3, #1
26     ADDEQ R4, R4, R3
27     SUB R5, R5, #1
28     B start
29
30 end:
31     STR R4, [R1]
32     SWI 0x011
33
```

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

General Purpose | Float | Lab1_q2.o

	Hexadecimal	init:	
R0	:4204	00001000: LDR r0, #d	00005384: 81018181
R1	:4204	00001004: LDR r1, #d+0x4	00005388: 81018181
R2	:0	00001008: MOV r4, #0	0000538C: 81018181
R3	:33	0000100C: MOV r5, #10	00005390: 81018181
R4	:46	00001010: start: CMP r5, #0	00005394: 81018181
R5	:0	00001014: BNE loop	00005398: 81018181
R6	:0	00001018: B end	0000539C: 81018181
R7	:0	0000101C: loop: LDR r3, [r0, #0]	000053A0: 81018181
R8	:0	00001020: ADD r0, r0, #4	000053A4: 81018181
R9	:0	00001024: TST r3, #1	000053A8: 81018181
R10(sl):0		00001028: ADDEQ r4, r4, r3	000053AC: 81018181
R11(fp):0		0000102C: SUB r5, r5, #1	000053B0: 81018181
R12(ip):0		00001030: B start	000053B4: 81018181
R13(sp):21504		00001034: end: STR r4, [r1, #0]	000053B8: 81018181
R14(lr):0		00001038: SWI 0x11	000053BC: 81018181
R15(pc):4152		0000103C: sd: ANDEQ r1, r0, r4, asr #32	000053C0: 81018181
		00001040: ANDEQ r1, r0, ip, rrx	000053C4: 81018181
		00001044: array: .word 0x0000000F, 0x0000000A, 0x0000000C, 0x0000000D	000053C8: 81018181
		00001048: .word 0x00000009, 0x0000002D, 0x00000010, 0x00000008	000053CC: 81018181
		0000104C: .word 0x00000019, 0x00000021	000053D0: 81018181
		00001050: evensum: .word 0x00000000	000053D4: 81018181
		00001054: .word 0x00000000	000053D8: 81018181
		00001058: .word 0x00000000	000053DC: 81018181
		0000105C: .word 0x00000000	000053E0: 81018181
		00001060: .word 0x00000000	000053E4: 81018181
		00001064: .word 0x00000000	000053E8: 81018181
		00001068: .word 0x00000000	000053EC: 81018181
		0000106C: .word 0x00000000	000053F0: 81018181
		00001070: .word 0x00000000	000053F4: 81018181
		00001074: .word 0x00000000	000053F8: 81018181
		00001078: .word 0x00000000	000053FC: 81018181
		0000107C: .word 0x00000000	000053FF: 81018181
		00001080: .word 0x00000000	00005400: 81018181
		00001084: .word 0x00000000	00005404: 81018181
		00001088: .word 0x00000000	00005408: 81018181
		0000108C: .word 0x00000000	0000540C: 81018181
		00001090: .word 0x00000000	00005410: 81018181
		00001094: .word 0x00000000	00005414: 81018181
		00001098: .word 0x00000000	00005418: 81018181
		0000109C: .word 0x00000000	0000541C: 81018181
		000010A0: .word 0x00000000	00005420: 81018181
		000010A4: .word 0x00000000	00005424: 81018181
		000010A8: .word 0x00000000	00005428: 81018181
		000010AC: .word 0x00000000	0000542C: 81018181
		000010B0: .word 0x00000000	00005430: 81018181
		000010B4: .word 0x00000000	00005434: 81018181
		000010B8: .word 0x00000000	00005438: 81018181
		000010BC: .word 0x00000000	0000543C: 81018181
		000010C0: .word 0x00000000	00005440: 81018181
		000010C4: .word 0x00000000	00005444: 81018181
		000010C8: .word 0x00000000	00005448: 81018181
		000010CC: .word 0x00000000	0000544C: 81018181
		000010D0: .word 0x00000000	0000544E: 81018181
		000010D4: .word 0x00000000	0000544F: 81018181
		000010D8: .word 0x00000000	00005450: 81018181
		000010DC: .word 0x00000000	00005451: 81018181
		000010E0: .word 0x00000000	00005452: 81018181
		000010E4: .word 0x00000000	00005453: 81018181
		000010E8: .word 0x00000000	00005454: 81018181
		000010EC: .word 0x00000000	00005455: 81018181
		000010F0: .word 0x00000000	00005456: 81018181
		000010F4: .word 0x00000000	00005457: 81018181
		000010F8: .word 0x00000000	00005458: 81018181
		000010FC: .word 0x00000000	00005459: 81018181
		00001100: .word 0x00000000	0000545A: 81018181
		00001104: .word 0x00000000	0000545B: 81018181
		00001108: .word 0x00000000	0000545C: 81018181
		0000110C: .word 0x00000000	0000545D: 81018181
		0000110E: .word 0x00000000	0000545E: 81018181
		00001110: .word 0x00000000	0000545F: 81018181
		00001112: .word 0x00000000	00005460: 81018181
		00001114: .word 0x00000000	00005461: 81018181
		00001116: .word 0x00000000	00005462: 81018181
		00001118: .word 0x00000000	00005463: 81018181
		0000111A: .word 0x00000000	00005464: 81018181
		0000111C: .word 0x00000000	00005465: 81018181
		0000111E: .word 0x00000000	00005466: 81018181
		00001120: .word 0x00000000	00005467: 81018181
		00001122: .word 0x00000000	00005468: 81018181
		00001124: .word 0x00000000	00005469: 81018181
		00001126: .word 0x00000000	0000546A: 81018181
		00001128: .word 0x00000000	0000546B: 81018181
		0000112A: .word 0x00000000	0000546C: 81018181
		0000112C: .word 0x00000000	0000546D: 81018181
		0000112E: .word 0x00000000	0000546E: 81018181
		00001130: .word 0x00000000	0000546F: 81018181
		00001132: .word 0x00000000	00005470: 81018181
		00001134: .word 0x00000000	00005471: 81018181
		00001136: .word 0x00000000	00005472: 81018181
		00001138: .word 0x00000000	00005473: 81018181
		0000113A: .word 0x00000000	00005474: 81018181
		0000113C: .word 0x00000000	00005475: 81018181
		0000113E: .word 0x00000000	00005476: 81018181
		00001140: .word 0x00000000	00005477: 81018181
		00001142: .word 0x00000000	00005478: 81018181
		00001144: .word 0x00000000	00005479: 81018181
		00001146: .word 0x00000000	0000547A: 81018181
		00001148: .word 0x00000000	0000547B: 81018181
		0000114A: .word 0x00000000	0000547C: 81018181
		0000114C: .word 0x00000000	0000547D: 81018181
		0000114E: .word 0x00000000	0000547E: 81018181
		00001150: .word 0x00000000	0000547F: 81018181
		00001152: .word 0x00000000	00005480: 81018181
		00001154: .word 0x00000000	00005481: 81018181
		00001156: .word 0x00000000	00005482: 81018181
		00001158: .word 0x00000000	00005483: 81018181
		0000115A: .word 0x00000000	00005484: 81018181
		0000115C: .word 0x00000000	00005485: 81018181
		0000115E: .word 0x00000000	00005486: 81018181
		00001160: .word 0x00000000	00005487: 81018181
		00001162: .word 0x00000000	00005488: 81018181
		00001164: .word 0x00000000	00005489: 81018181
		00001166: .word 0x00000000	0000548A: 81018181
		00001168: .word 0x00000000	0000548B: 81018181
		0000116A: .word 0x00000000	0000548C: 81018181
		0000116C: .word 0x00000000	0000548D: 81018181
		0000116E: .word 0x00000000	0000548E: 81018181
		00001170: .word 0x00000000	0000548F: 81018181
		00001172: .word 0x00000000	00005490: 81018181
		00001174: .word 0x00000000	00005491: 81018181
		00001176: .word 0x00000000	00005492: 81018181
		00001178: .word 0x00000000	00005493: 81018181
		0000117A: .word 0x00000000	00005494: 81018181
		0000117C: .word 0x00000000	00005495: 81018181
		0000117E: .word 0x00000000	00005496: 81018181
		00001180: .word 0x00000000	00005497: 81018181
		00001182: .word 0x00000000	00005498: 81018181
		00001184: .word 0x00000000	00005499: 81018181
		00001186: .word 0x00000000	0000549A: 81018181
		00001188: .word 0x00000000	0000549B: 81018181
		0000118A: .word 0x00000000	0000549C: 81018181
		0000118C: .word 0x00000000	0000549D: 81018181
		0000118E: .word 0x00000000	0000549E: 81018181
		00001190: .word 0x00000000	0000549F: 81018181
		00001192: .word 0x00000000	000054A0: 81018181
		00001194: .word 0x00000000	000054A1: 81018181
		00001196: .word 0x00000000	000054A2: 81018181
		00001198: .word 0x00000000	000054A3: 81018181
		0000119A: .word 0x00000000	000054A4: 81018181
		0000119C: .word 0x00000000	000054A5: 81018181
		0000119E: .word 0x00000000	000054A6: 81018181
		000011A0: .word 0x00000000	000054A7: 81018181
		000011A2: .word 0x00000000	000054A8: 81018181
		000011A4: .word 0x00000000	000054A9: 81018181
		000011A6: .word 0x00000000	000054AA: 81018181
		000011A8: .word 0x00000000	000054AB: 81018181
		000011AA: .word 0x00000000	000054AC: 81018181
		000011AC: .word 0x00000000	000054AD: 81018181
		000011AE: .word 0x00000000	000054AE: 81018181
		000011B0: .word 0x00000000	000054AF: 81018181
		000011B2: .word 0x00000000	000054B0: 81018181
		000011B4: .word 0x00000000	000054B1: 81018181
		000011B6: .word 0x00000000	000054B2: 81018181
		000011B8: .word 0x00000000	000054B3: 81018181
		000011BA: .word 0x00000000	000054B4: 81018181
		000011BC: .word 0x00000000	000054B5: 81018181
		000011BE: .word 0x00000000	000054B6: 81018181
		000011C0: .word 0x00000000	000054B7: 81018181
		000011C2: .word 0x00000000	000054B8: 81018181
		000011C4: .word 0x00000000	000054B9: 81018181
		000011C6: .word 0x00000000	000054BA: 81018181
		000011C8: .word 0x00000000	000054BB: 81018181
		000011CA: .word 0x00000000	000054BC: 81018181
		000011CC: .word 0x00000000	000054BD: 81018181
		000011CE: .word 0x00000000	000054BE: 81018181
		000011D0: .word 0x00000000	000054BF: 81018181
		000011D2: .word 0x00000000	000054C0: 81018181
		000011D4: .word 0x00000000	000054C1: 81018181
		000011D6: .word 0x00000000	000054C2: 81018181
		000011D8: .word 0x00000000	000054C3: 81018181
		000011DA: .word 0x00000000	000054C4: 81018181
		000011DC: .word 0x00000000	000054C5: 81018181
		000011DE: .word 0x00000000	000054C6: 81018181
		000011E0: .word 0x00000000	000054C7: 81018181
		000011E2: .word 0x00000000	000054C8: 81018181
		000011E4: .word 0x00000000	000054C9: 81018181
		000011E6: .word 0x00000000	000054CA: 81018181
		000011E8: .word 0x00000000	000054CB: 81018181
		000011EA: .word 0x00000000	000054CC: 81018181
		000011EC: .word 0x00000000	000054CD: 81018181
		000011EE: .word 0x00000000	000054CE: 81018181
		000011F0: .word 0x00000000	000054CF: 81018181
		000011F2: .word 0x00000000	000054D0: 81018181
		000011F4: .word 0x00000000	000054D1: 81018181
		000011F6: .word 0x00000000	000054D2: 81018181
		000011F8: .word 0x00000000	000054D3: 81018181
		000011FA: .word 0x00000000	000054D4: 81018181
		000011FC: .word 0x00000000	000054D5: 81018181
		000011FE: .word 0x00000000	000054D6: 81018181
		00001200: .word 0x00000000	000054D7: 81018181

R0 :4204
R1 :4204
R2 :0
R3 :33
R4 :46
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
Thumb(T) :0

00001004:E59F10
00001008:E3A040
0000100C:E3A050
00001010:
00001010:E35500
00001014:1A0000
00001018:EA0000
0000101C:
0000101C:E59030
00001020:E28000
00001024:E31300
00001028:008440
0000102C:E24550
00001030:EAF0FF
00001034:
00001034:E58140
00001038:EF0000
0000103C:
0000103C:000010
00001040:000010
00001044:
00001044:
00001054:
00001064:
0000106C:
0000106C:

OutputView WatchView MemoryView0

0000106c

0000106C 0000002E 81818181 81818181

3

Write a ALP using ARMTDMI-ISA to store odd and even numbers in separate memory locations starting from LOCA and LOCB respectively

ARRAY: .word 10,50,41,55,30,20,11,5,100,77

LOCA: .word 0,0,0,0,0

LOCB: .word 0,0,0,0,0

```
ARMSim_files > ASM lab1_q3.s
1  @ Write a ALP using ARMTDMI-ISA to store odd and even numbers in separate memory locations starting from LOCA and LOCB respectively
2  @ ARRAY: .word 10,50,41,55,30,20,11,5,100,77
3  @ LOCA: .word 0,0,0,0,0
4  @ LOCB: .word 0,0,0,0,0
5
6  .data
7  array: .word 10, 50, 41, 55, 30, 20, 11, 5, 100, 77
8  oddarray: .word 0, 0, 0, 0, 0, 0
9  evenarray: .word 0, 0, 0, 0, 0, 0
10
11  init:
12      LDR R0, =array
13      LDR R1, =oddarray
14      LDR R2, =evenarray
15      MOV R3, #10
16
17  start:
18      CMP R3, #0
19      BNE loop
20      B end
21
22  loop:
23      LDR R4, [R0]
24      TST R4, #1
25      STREQ R4, [R2]
26      ADDEQ R2, R2, #4
27      STRNE R4, [R1]
28      ADDNE R1, R1, #4
29      ADD R0, R0, #4
30      SUB R3, R3, #1
31      B start
32
33  end:
34      SWI 0x011
35
```



A: .word 10,50,41,55,30,20,11,5,100,77

ARMSim_files > **asm** lab1_q4.s

```
1  @ Write an ALP using ARM7TDMI to find the largest number from a given set of numbers:
2  @ A: .word 10,50,41,55,30,20,11,5,100,77
3
4  .data
5  array: .word 10, 50, 41, 55, 30, 20, 11, 5, 100, 77
6
7  init:
8      LDR R0, =array
9      LDR R1, [R0]
10     MOV R2, #10
11
12  start:
13     CMP R2, #0
14     BNE loop
15     B end
16
17  loop:
18     LDR R3, [R0]
19     CMP R1, R3
20     MOVL R1, R3
21     ADD R0, R0, #4
22     SUB R2, R2, #1
23     B start
24
25  end:
26     SWI 0x011
27
```

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

General Purpose Floats Lab1_q4.o

Hexadecimal	00001000:	array:	000053B4: 81818181
Unsigned Decimal	00001000:	.word 0x0000000A, 0x00000032, 0x00000029, 0x00000037	000053B8: 81818181
Signed Decimal	00001000:	.word 0x0000001E, 0x00000014, 0x0000000B, 0x00000005	000053BC: 81818181
R0	:4136		000053C0: 81818181
R1	:100		000053C4: 81818181
R2	:0		000053C8: 81818181
R3	:77		000053CC: 81818181
R4	:0		000053D0: 81818181
R5	:0		000053D4: 81818181
R6	:0		000053D8: 81818181
R7	:0		000053DC: 81818181
R8	:0		000053E0: 81818181
R9	:0		000053E4: 81818181
R10(s1):0			000053E8: 81818181
R11(fp):0			000053EC: 81818181
R12(ip):0			000053F0: 81818181
R13(sp):21504			000053F4: 81818181
R14(lr):0			000053F8: 81818181
R15(pc):4184			000053FC: 81818181
CPSR Register			00005400: 81818181
Negative(N):0			00005404: 81818181
Zero(Z):1			00005408: 81818181
Carry(C):1			0000540C: 81818181
Overflow(V):0			00005410: 81818181
I/O Disable:1			00005414: 81818181
FIQ Disable:1			00005418: 81818181
Thumb(T):0			0000541C: 81818181
CPU Mode :System			00005420: 81818181
0x600000df			00005424: 81818181
			00005428: 81818181
			0000542C: 81818181
			00005430: 81818181
			00005434: 81818181
			00005438: 81818181
			0000543C: 81818181
			00005440: 81818181
			00005444: 81818181
			00005448: 81818181

OutputView WatchView

Console | stdin/stdout/stderr |

Loading object code file lab1_q4.o

Execution starting ...

Execution ending, Instruction Count:97 Elapsed Time:00:00:00.0249740

Instructions per second:3884

OutputView WatchView MemoryView0

Word Size	8bit	16bit	32bit
00001028	E59F002C	E5901000	E3A0200A
00001064	81818181	81818181	81818181
000010A0	81818181	81818181	81818181
000010DC	81818181	81818181	81818181

Assignments Questions

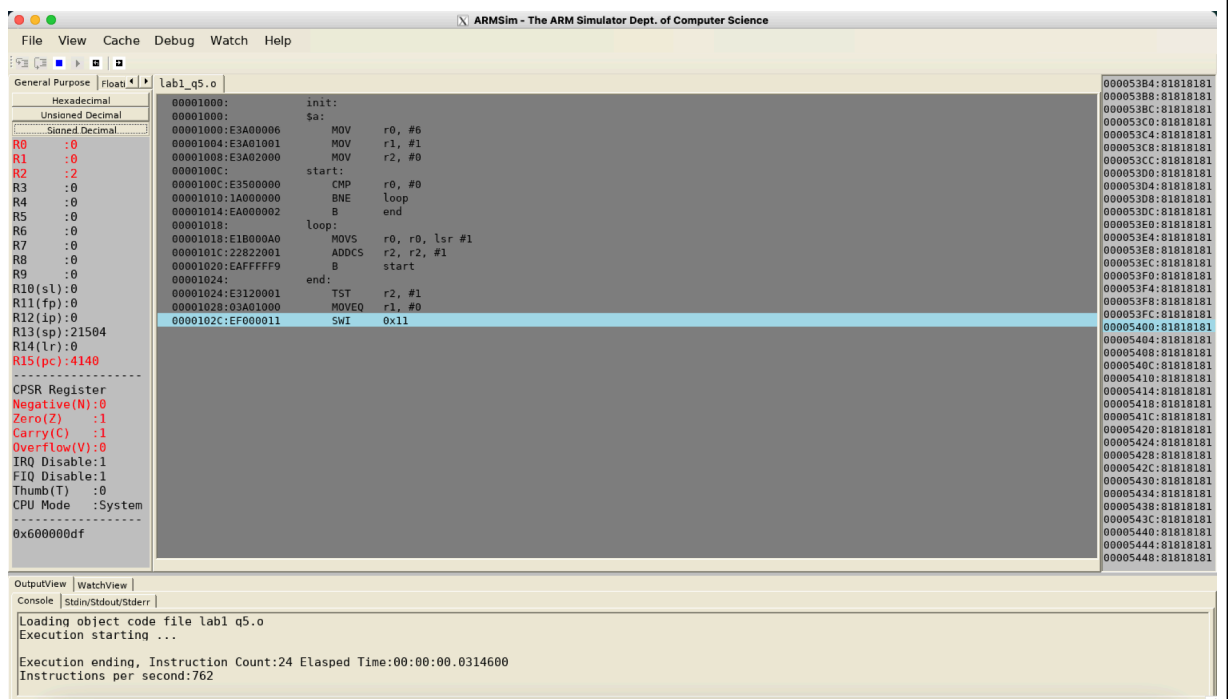
5 Write an ALP using ARM7TDMI to find whether the given number is even parity.

ARMSim_files > ASM lab1_q5.s

```

1  @ Write an ALP using ARM7TDMI to find whether the given number is even parity.
2  @ Set 0 on even parity and 1 on odd parity
3
4  .text
5
6  init:
7      MOV R0, #6
8      MOV R1, #1
9      MOV R2, #0
10
11  start:
12      CMP R0, #0
13      BNE loop
14      B end
15
16  loop:
17      MOVS R0, R0, LSR#1
18      ADDCS R2, R2, #1
19      B start
20
21  end:
22      TST R2, #1
23      MOVEQ R1, #0
24      SWI 0x011
25

```



6

Write an ALP using ARM7TDMI to multiplication of 38x72 without using MUL instructions.

(Hint: barrel shifter instructions.)

(Note :any number can be considered as multiplier)

ARMSim_files > **asm** lab1_q6.s

```

1  @ Write an ALP using ARM7TDMI to multiplication of 38x72 without using MUL instructions.
2  @ (Hint: barrel shifter instructions.)
3  @ (Note :any number can be considered as multiplier)
4
5  @ Logic used: break 38 into 32 + 4 + 2
6
7  .text
8  MOV R0, #38
9  MOV R1, #72
10 MOV R2, R1, LSL#5
11 ADD R3, R2, R1, LSL#2
12 ADD R4, R3, R1, LSL#1
13 SWI 0x011
14 |

```

