ARM ASSEMBLY LAB-2

By

Swapneel Pimparkar (CS18M516)
CS6620a
Prof. Madhu Mutyam

STATUS UPDATE

The ARM assembly code was written for 32 bit processor and verified using ARMSim simulator successfully.

EXERCISE – FIND MIN, MAX AND COUNT OF SET OF NONZERO NUMBERS.

The main part of objective/problem statement of the exercise is as follows:

Write an assembly program to compute the maximum and minimum values in a given set of non-zero unsigned integer numbers. Your program also should compute the total number of integers present in the set (other than the terminating 0). Note that your program should scan through all the elements of the set only once.

There are multiple ways to produce the mnemonics. A global array (instead of localized to a subroutine) of numbers is chosen for implementation. No subroutine is used. The logic used is part of the code uploaded too.

The data sequence used to verification of the code is – 45,67,89,3,6,1,92,0.

Register R0 is used to count the numbers in a set.

Register R1 is used to store maximum of the numbers in a set.

Register R2 is used to store minimum of the numbers in a set.

The program is also verified for following set of numbers.

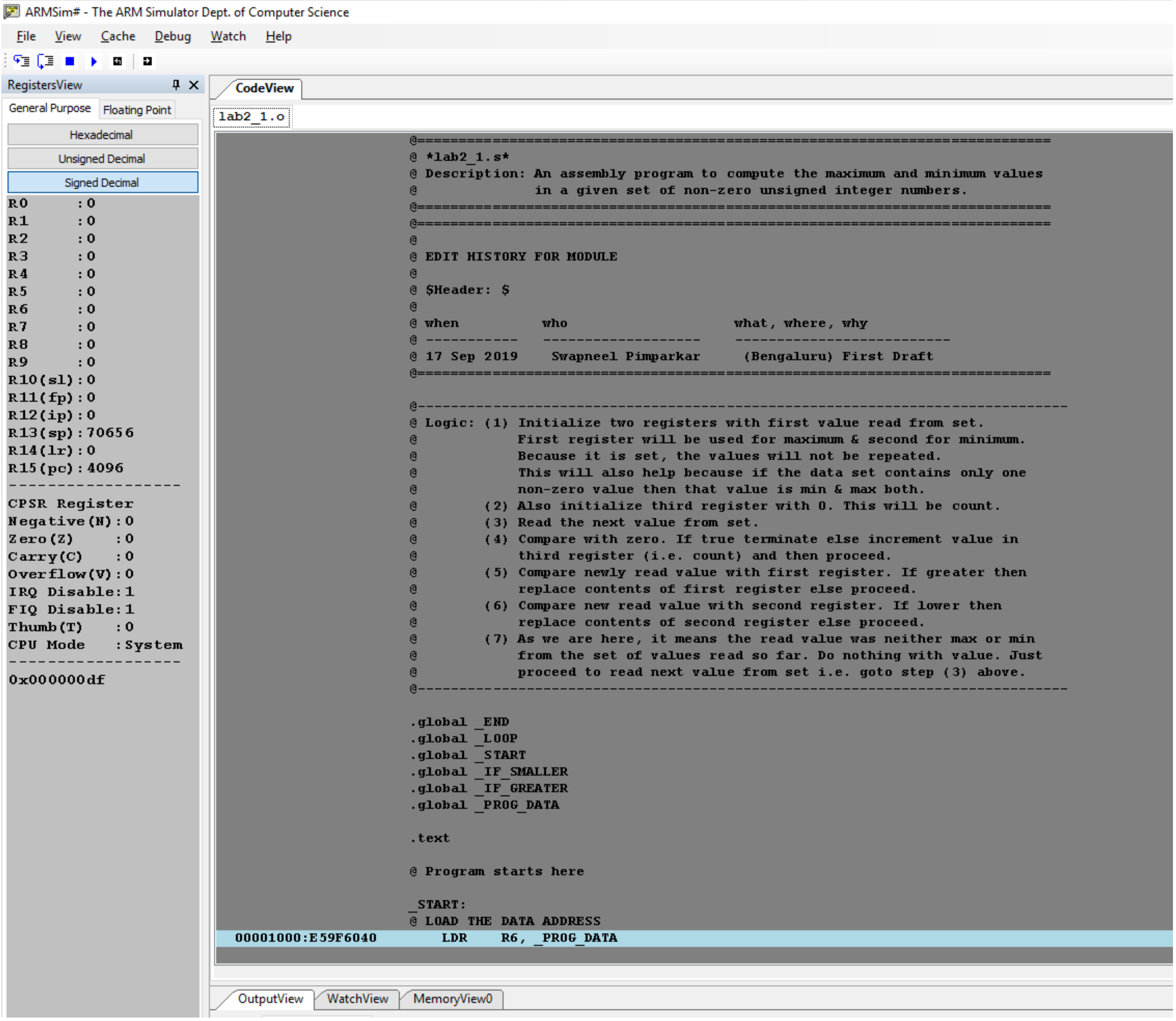
{0}

{1,0}

Right now, only positive integers within applicable range are used for programming and verification.

INITIAL STATE SCREENSHOT

The sample output screenshot for the logic is – (Registers listed on the left are to be noted. All are zero to begin with).

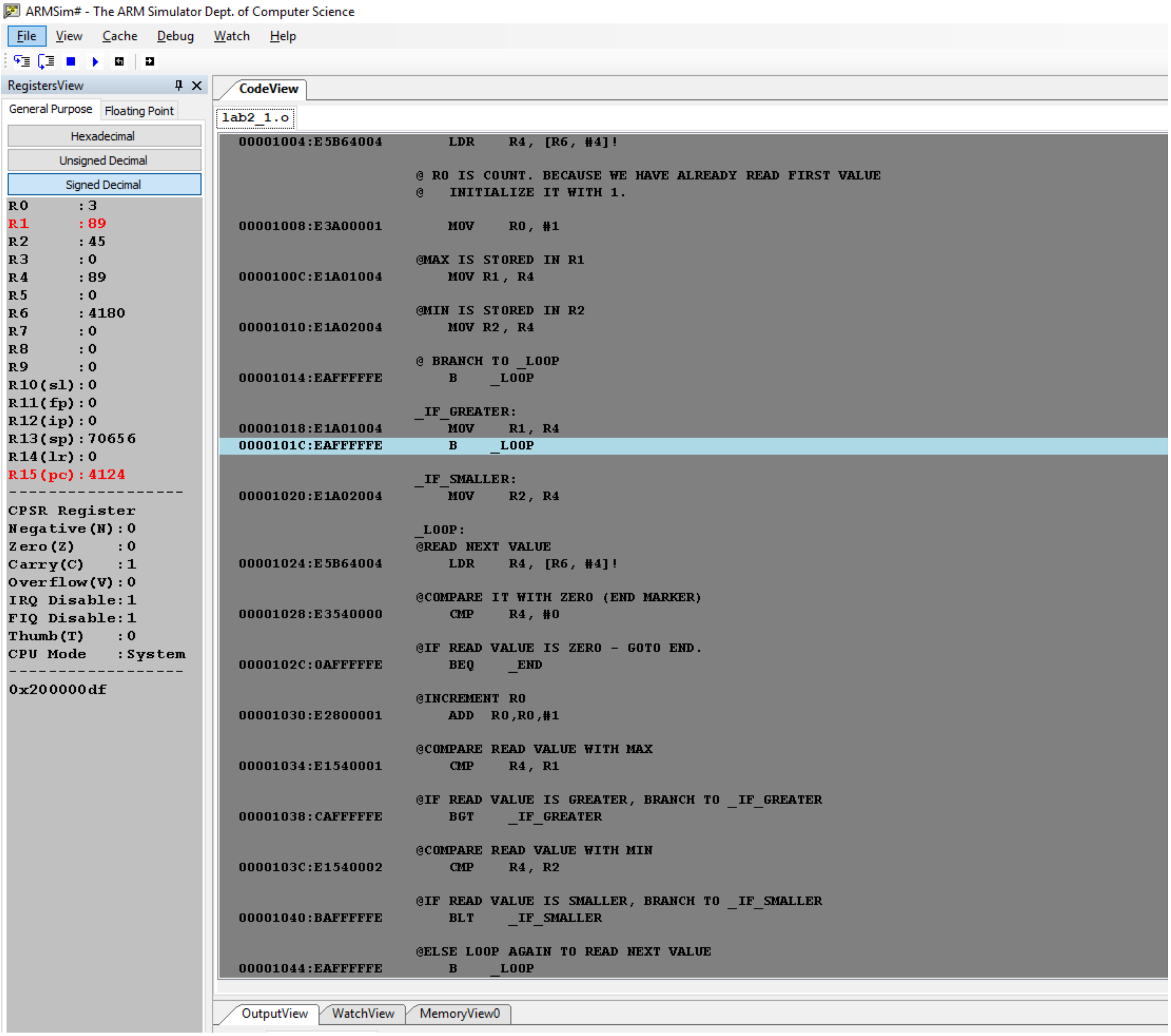


```
R0      :0
R1      :0
R2      :0
R3      :0
R4      :0
R5      :0
R6      :0
R7      :0
R8      :0
R9      :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4096
------------------
CPSR Register
Negative(N):0
Zero(Z)    :0
Carry(C)   :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x000000df
```

```
@==========================================================================
@ *lab2_1.s*
@ Description: An assembly program to compute the maximum and minimum values
@              in a given set of non-zero unsigned integer numbers.
@==========================================================================
@==========================================================================
@
@ EDIT HISTORY FOR MODULE
@
@ $Header: $
@
@ when          who                     what, where, why
@ -----------   -------------------     --------------------------
@ 17 Sep 2019    Swapneel Pimparkar      (Bengaluru) First Draft
@==========================================================================

@----------------------------------------------------------------------------
@ Logic: (1) Initialize two registers with first value read from set.
@            First register will be used for maximum & second for minimum.
@            Because it is set, the values will not be repeated.
@            This will also help because if the data set contains only one
@            non-zero value then that value is min & max both.
@        (2) Also initialize third register with 0. This will be count.
@        (3) Read the next value from set.
@        (4) Compare with zero. If true terminate else increment value in
@            third register (i.e. count) and then proceed.
@        (5) Compare newly read value with first register. If greater then
@            replace contents of first register else proceed.
@        (6) Compare new read value with second register. If lower then
@            replace contents of second register else proceed.
@        (7) As we are here, it means the read value was neither max or min
@            from the set of values read so far. Do nothing with value. Just
@            proceed to read next value from set i.e. goto step (3) above.
@----------------------------------------------------------------------------

.global _END
.global _LOOP
.global _START
.global _IF_SMALLER
.global _IF_GREATER
.global _PROG_DATA

.text

@ Program starts here

_START:
@ LOAD THE DATA ADDRESS
00001000:E59F6040       LDR     R6, _PROG_DATA
```

INTERMEDIATE STATE SCREENSHOT

After 3 values are read, the state of the registers is as follows:

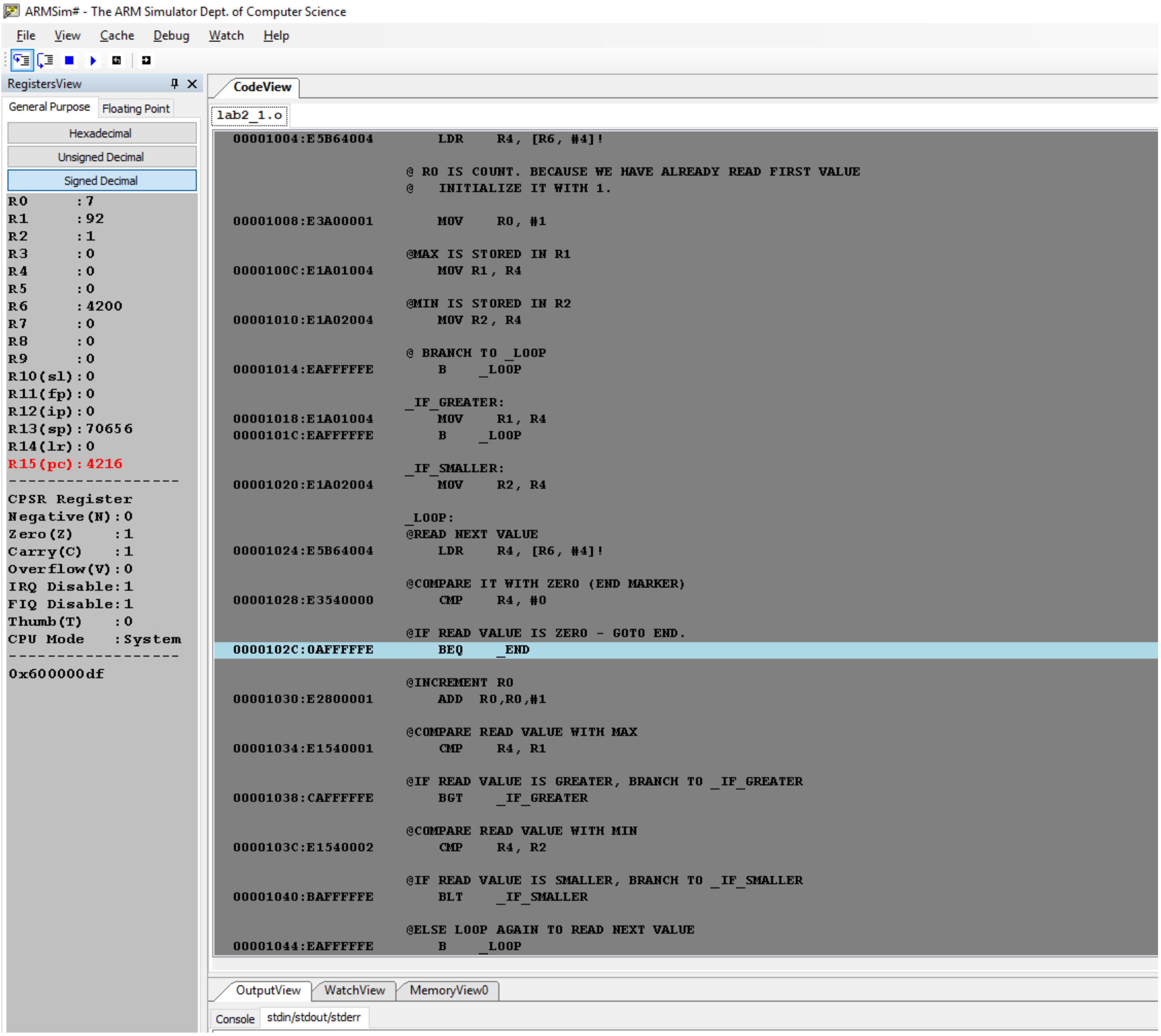


```
R0       :3
R1       :89
R2       :45
R3       :0
R4       :89
R5       :0
R6       :4180
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4124
------------------
CPSR Register
Negative(N):0
Zero(Z)     :0
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
------------------
0x200000df
```

```
00001004:E5B64004        LDR     R4, [R6, #4]!

                     @ R0 IS COUNT. BECAUSE WE HAVE ALREADY READ FIRST VALUE
                     @   INITIALIZE IT WITH 1.

00001008:E3A00001        MOV     R0, #1

                     @MAX IS STORED IN R1
0000100C:E1A01004        MOV R1, R4

                     @MIN IS STORED IN R2
00001010:E1A02004        MOV R2, R4

                     @ BRANCH TO _LOOP
00001014:EAFFFFFE        B     _LOOP

                     _IF_GREATER:
00001018:E1A01004        MOV     R1, R4
0000101C:EAFFFFFE        B     _LOOP

                     _IF_SMALLER:
00001020:E1A02004        MOV     R2, R4

                     _LOOP:
                     @READ NEXT VALUE
00001024:E5B64004        LDR     R4, [R6, #4]!

                     @COMPARE IT WITH ZERO (END MARKER)
00001028:E3540000        CMP     R4, #0

                     @IF READ VALUE IS ZERO - GOTO END.
0000102C:0AFFFFFE        BEQ     _END

                     @INCREMENT R0
00001030:E2800001        ADD  R0,R0,#1

                     @COMPARE READ VALUE WITH MAX
00001034:E1540001        CMP     R4, R1

                     @IF READ VALUE IS GREATER, BRANCH TO _IF_GREATER
00001038:CAFFFFFE        BGT     _IF_GREATER

                     @COMPARE READ VALUE WITH MIN
0000103C:E1540002        CMP     R4, R2

                     @IF READ VALUE IS SMALLER, BRANCH TO _IF_SMALLER
00001040:BAFFFFFE        BLT     _IF_SMALLER

                     @ELSE LOOP AGAIN TO READ NEXT VALUE
00001044:EAFFFFFE        B     _LOOP
```

FINAL STATE SCREENSHOT

At the end, the register set state is as follows (and it is as per expectations):



```
R0       :7
R1       :92
R2       :1
R3       :0
R4       :0
R5       :0
R6       :4200
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4216
------------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
------------------
0x600000df
```

```
00001004:E5B64004        LDR    R4, [R6, #4]!

                        @ RO IS COUNT. BECAUSE WE HAVE ALREADY READ FIRST VALUE
                        @   INITIALIZE IT WITH 1.

00001008:E3A00001        MOV    R0, #1

                        @MAX IS STORED IN R1
0000100C:E1A01004        MOV R1, R4

                        @MIN IS STORED IN R2
00001010:E1A02004        MOV R2, R4

                        @ BRANCH TO _LOOP
00001014:EAFFFFFE        B    _LOOP

                        _IF_GREATER:
00001018:E1A01004        MOV    R1, R4
0000101C:EAFFFFFE        B    _LOOP

                        _IF_SMALLER:
00001020:E1A02004        MOV    R2, R4

                        _LOOP:
                        @READ NEXT VALUE
00001024:E5B64004        LDR    R4, [R6, #4]!

                        @COMPARE IT WITH ZERO (END MARKER)
00001028:E3540000        CMP    R4, #0

                        @IF READ VALUE IS ZERO - GOTO END.
0000102C:0AFFFFFE        BEQ    _END

                        @INCREMENT R0
00001030:E2800001        ADD  R0,R0,#1

                        @COMPARE READ VALUE WITH MAX
00001034:E1540001        CMP    R4, R1

                        @IF READ VALUE IS GREATER, BRANCH TO _IF_GREATER
00001038:CAFFFFFE        BGT    _IF_GREATER

                        @COMPARE READ VALUE WITH MIN
0000103C:E1540002        CMP    R4, R2

                        @IF READ VALUE IS SMALLER, BRANCH TO _IF_SMALLER
00001040:BAFFFFFE        BLT    _IF_SMALLER

                        @ELSE LOOP AGAIN TO READ NEXT VALUE
00001044:EAFFFFFE        B    _LOOP
```