Assignment 2

Digital stopwatch in ARMLite

COS10004

[redacted]

Overview

We were tasked to make a digital stopwatch where we can see each tick and then be able to pause un-pause and reset the clock. Along with it we had to make it in a way so that we could be able to store time in split time. The code I have made is a user friendly one which shows the instructions on the starting for the start stop, pause and reset it. I have made my code as easy and simple according to my knowledge.

STAGE 1

The program displays the seconds incrementing in the text output display area in a perceivable way with each tick visible and functions to pause un-pause and reset the timer. It has "p" for pause/resume and "R" for reset. It has a loop that counts seconds in the ones place and increments the counter stored in the memory. When the counter tens "10" then it shifts the counting to the tens place. When the counter reaches the 6 in the tens place it stops. (FIG1.1-1.2)

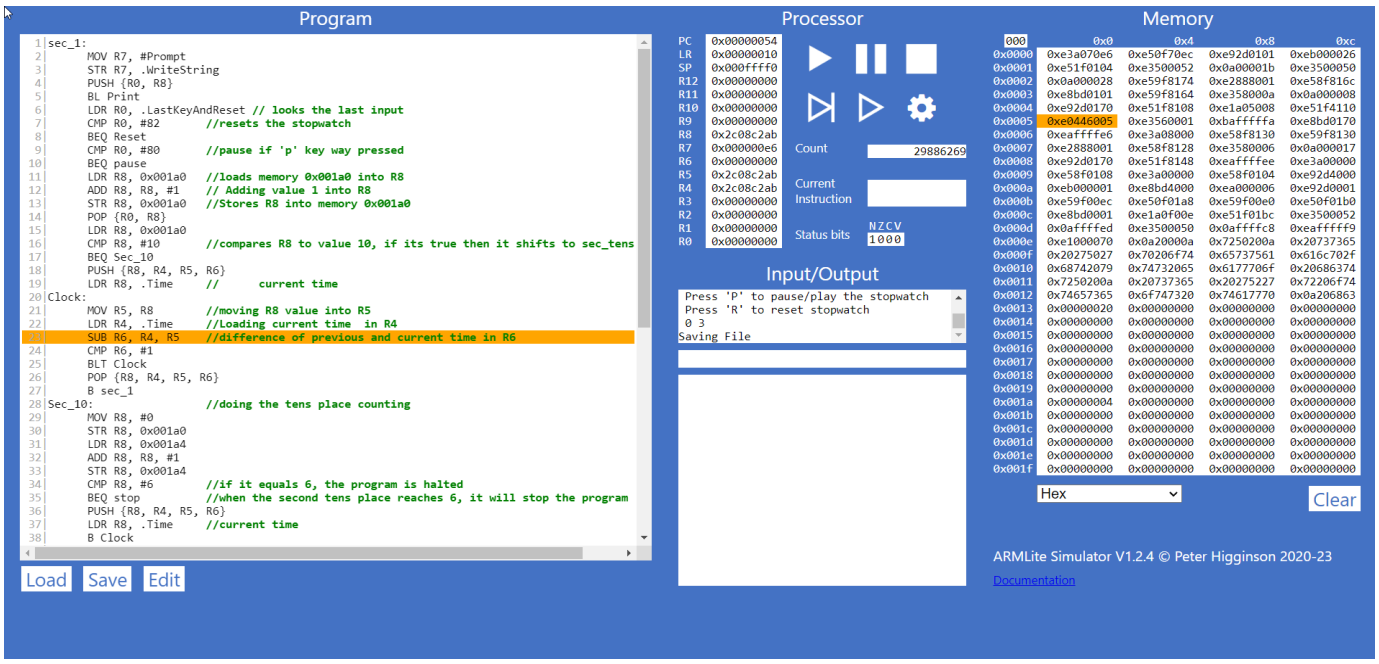
Program
Processor
Memory
sec_1:
MOV R7, #Prompt
STR R7, .WriteString
PUSH {R0, R8}
BL Print
LDR R0, .LastKeyAndReset // looks the last input
CMP R0, #82 //resets the stopwatch
BEQ Reset
CMP R0, #80 //pause if 'p' key way pressed
BEQ pause
LDR R8, 0x001a0 //loads memory 0x001a0 into R8
ADD R8, R8, #1 // Adding value 1 into R8
STR R8, 0x001a0 //Stores R8 into memory 0x001a0
POP {R0, R8}
LDR R8, 0x001a0
CMP R8, #10 //compares R8 to value 10, if its true then it shifts to sec_tens
BEQ Sec_10
PUSH {R8, R4, R5, R6}
LDR R8, .Time // current time
Clock:
MOV R5, R8 //moving R8 value into R5
LDR R4, .Time //Loading current time in R4
SUB R6, R4, R5 //difference of previous and current time in R6
CMP R6, #1
BLT Clock
POP {R8, R4, R5, R6}
B sec_1
Sec_10: //doing the tens place counting
MOV R8, #0
STR R8, 0x001a0
LDR R8, 0x001a4
ADD R8, R8, #1
STR R8, 0x001a4
CMP R8, #6 //if it equals 6, the program is halted
BEQ stop //when the second tens place reaches 6, it will stop the program
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B Clock
Load
Save
Edit
PC 0x00000054
LR 0x00000010
SP 0x000ffff0
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x2c08c2ab
R7 0x000000e6
R6 0x00000000
R5 0x2c08c2ab
R4 0x2c08c2ab
R3 0x00000000
R2 0x00000000
R1 0x00000000
R0 0x00000000
Count 29886269
Current Instruction
Status bits NZCV 1000
Input/Output
Press 'P' to pause/play the stopwatch
Press 'R' to reset stopwatch
0 3
Saving File
000 0x0 0x4 0x8 0xc
0x0000 0xe3a070e6 0xe50f70ec 0xe92d0101 0xeb000026
0x0001 0xe51f0104 0xe3500052 0x0a00001b 0xe3500050
0x0002 0x0a000028 0xe59f8174 0xe2888001 0xe58f816c
0x0003 0xe8bd0101 0xe59f8164 0xe358000a 0x0a000008
0x0004 0xe92d0170 0xe51f8108 0xe1a05008 0xe51f4110
0x0005 0xe0446005 0xe3560001 0xbaffffa 0xe8bd0170
0x0006 0xeafffe6 0xe3a08000 0xe58f8130 0xe59f8130
0x0007 0xe2888001 0xe58f8128 0xe3580006 0x0a000017
0x0008 0xe92d0170 0xe51f8148 0xeafffee 0xe3a00000
0x0009 0xe58f0108 0xe3a00000 0xe58f0104 0xe92d4000
0x000a 0xeb000001 0xe8bd4000 0xea000006 0xe92d0001
0x000b 0xe59f00ec 0xe50f01a8 0xe59f00e0 0xe50f01b0
0x000c 0xe8bd0001 0xe1a0f00e 0xe51f01bc 0xe3500052
0x000d 0x0afffed 0xe3500050 0x0afffc8 0xeafffff9
0x000e 0xe1000070 0x0a20000a 0x7250200a 0x20737365
0x000f 0x20275027 0x70206f74 0x65737561 0x616c702f
0x0010 0x68742079 0x74732065 0x6177706f 0x20686374
0x0011 0x7250200a 0x20737365 0x20275227 0x72206f74
0x0012 0x74657365 0x6f747320 0x74617770 0x0a206863
0x0013 0x00000020 0x00000000 0x00000000 0x00000000
0x0014 0x00000000 0x00000000 0x00000000 0x00000000
0x0015 0x00000000 0x00000000 0x00000000 0x00000000
0x0016 0x00000000 0x00000000 0x00000000 0x00000000
0x0017 0x00000000 0x00000000 0x00000000 0x00000000
0x0018 0x00000000 0x00000000 0x00000000 0x00000000
0x0019 0x00000000 0x00000000 0x00000000 0x00000000
0x001a 0x00000004 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

FIG 1.1

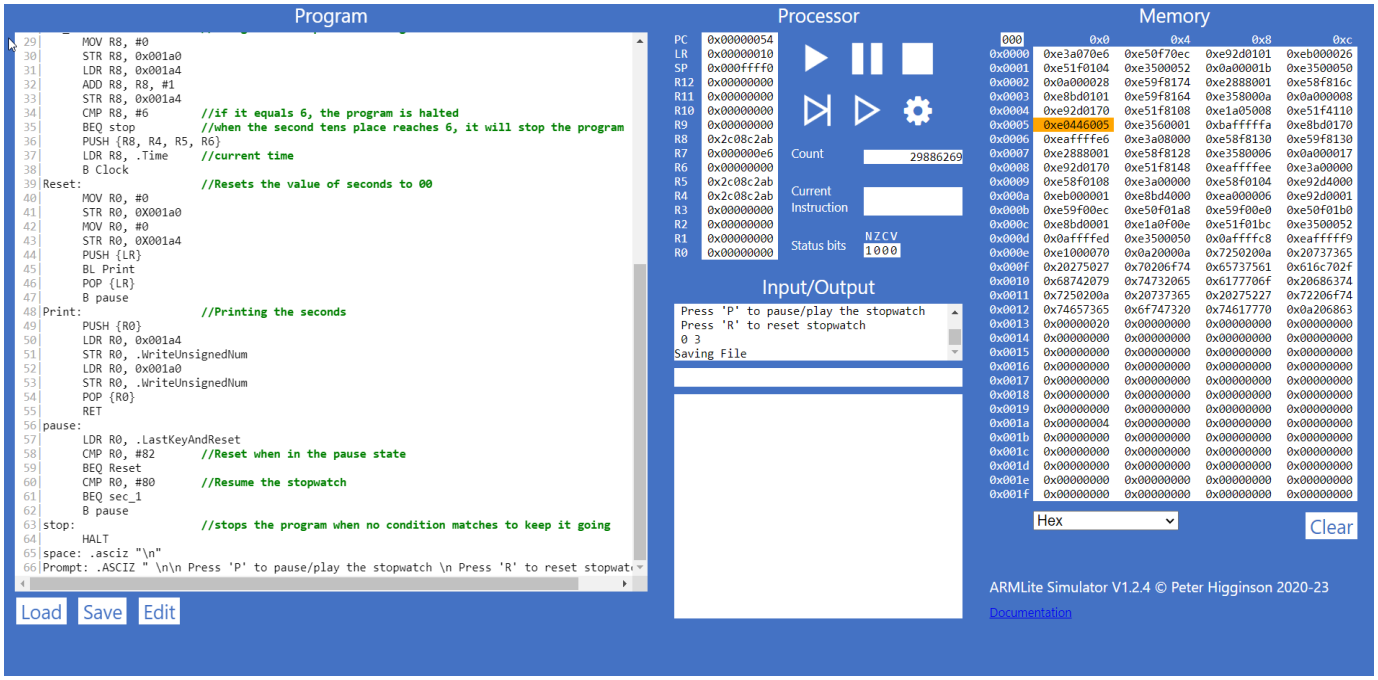
Program
Processor
Memory
MOV R8, #0
STR R8, 0x001a0
LDR R8, 0x001a4
ADD R8, R8, #1
STR R8, 0x001a4
CMP R8, #6 //if it equals 6, the program is halted
BEQ stop //when the second tens place reaches 6, it will stop the program
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B Clock
Reset: //Resets the value of seconds to 00
MOV R0, #0
STR R0, 0X001a0
MOV R0, #0
STR R0, 0X001a4
PUSH {LR}
BL Print
POP {LR}
B pause
Print: //Printing the seconds
PUSH {R0}
LDR R0, 0x001a4
STR R0, .WriteUnsignedNum
LDR R0, 0x001a0
STR R0, .WriteUnsignedNum
POP {R0}
RET
pause:
LDR R0, .LastKeyAndReset
CMP R0, #82 //Reset when in the pause state
BEQ Reset
CMP R0, #80 //Resume the stopwatch
BEQ sec_1
B pause
stop: //stops the program when no condition matches to keep it going
HALT
space: .asciz "\n"
Prompt: .ASCIZ " \n\n Press 'P' to pause/play the stopwatch \n Press 'R' to reset stopwat
Load
Save
Edit
PC 0x00000054
LR 0x00000010
SP 0x000ffff0
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x2c08c2ab
R7 0x000000e6
R6 0x00000000
R5 0x2c08c2ab
R4 0x2c08c2ab
R3 0x00000000
R2 0x00000000
R1 0x00000000
R0 0x00000000
Count 29886269
Current Instruction
Status bits NZCV 1000
Input/Output
Press 'P' to pause/play the stopwatch
Press 'R' to reset stopwatch
0 3
Saving File
000 0x0 0x4 0x8 0xc
0x0000 0xe3a070e6 0xe50f70ec 0xe92d0101 0xeb000026
0x0001 0xe51f0104 0xe3500052 0x0a00001b 0xe3500050
0x0002 0x0a000028 0xe59f8174 0xe2888001 0xe58f816c
0x0003 0xe8bd0101 0xe59f8164 0xe358000a 0x0a000008
0x0004 0xe92d0170 0xe51f8108 0xe1a05008 0xe51f4110
0x0005 0xe0446005 0xe3560001 0xbaffffa 0xe8bd0170
0x0006 0xeafffe6 0xe3a08000 0xe58f8130 0xe59f8130
0x0007 0xe2888001 0xe58f8128 0xe3580006 0x0a000017
0x0008 0xe92d0170 0xe51f8148 0xeafffee 0xe3a00000
0x0009 0xe58f0108 0xe3a00000 0xe58f0104 0xe92d4000
0x000a 0xeb000001 0xe8bd4000 0xea000006 0xe92d0001
0x000b 0xe59f00ec 0xe50f01a8 0xe59f00e0 0xe50f01b0
0x000c 0xe8bd0001 0xe1a0f00e 0xe51f01bc 0xe3500052
0x000d 0x0afffed 0xe3500050 0x0afffc8 0xeafffff9
0x000e 0xe1000070 0x0a20000a 0x7250200a 0x20737365
0x000f 0x20275027 0x70206f74 0x65737561 0x616c702f
0x0010 0x68742079 0x74732065 0x6177706f 0x20686374
0x0011 0x7250200a 0x20737365 0x20275227 0x72206f74
0x0012 0x74657365 0x6f747320 0x74617770 0x0a206863
0x0013 0x00000020 0x00000000 0x00000000 0x00000000
0x0014 0x00000000 0x00000000 0x00000000 0x00000000
0x0015 0x00000000 0x00000000 0x00000000 0x00000000
0x0016 0x00000000 0x00000000 0x00000000 0x00000000
0x0017 0x00000000 0x00000000 0x00000000 0x00000000
0x0018 0x00000000 0x00000000 0x00000000 0x00000000
0x0019 0x00000000 0x00000000 0x00000000 0x00000000
0x001a 0x00000004 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

Fig 1.2

STAGE 2

The stage 2 is made on the existing stage 1 code, but it introduces additional functionality for the minutes as now they go upto 99. There is a loop for counting the seconds in the one place and when it reaches 10, it switches the count to the tenth place. It features the same pause and resume from stage 1. The reset action resets both the seconds and minutes counter. The Program also prints instructions for the users ease. Fig (2.1-2.3)

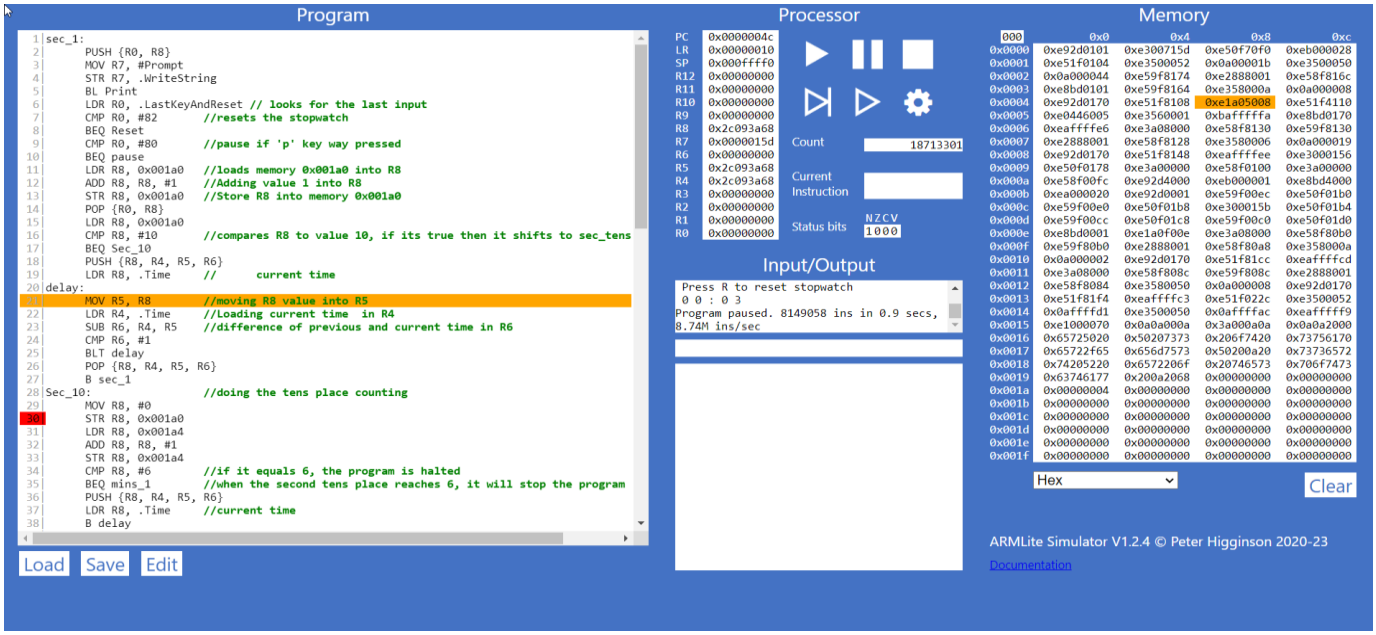

Fig 2.1

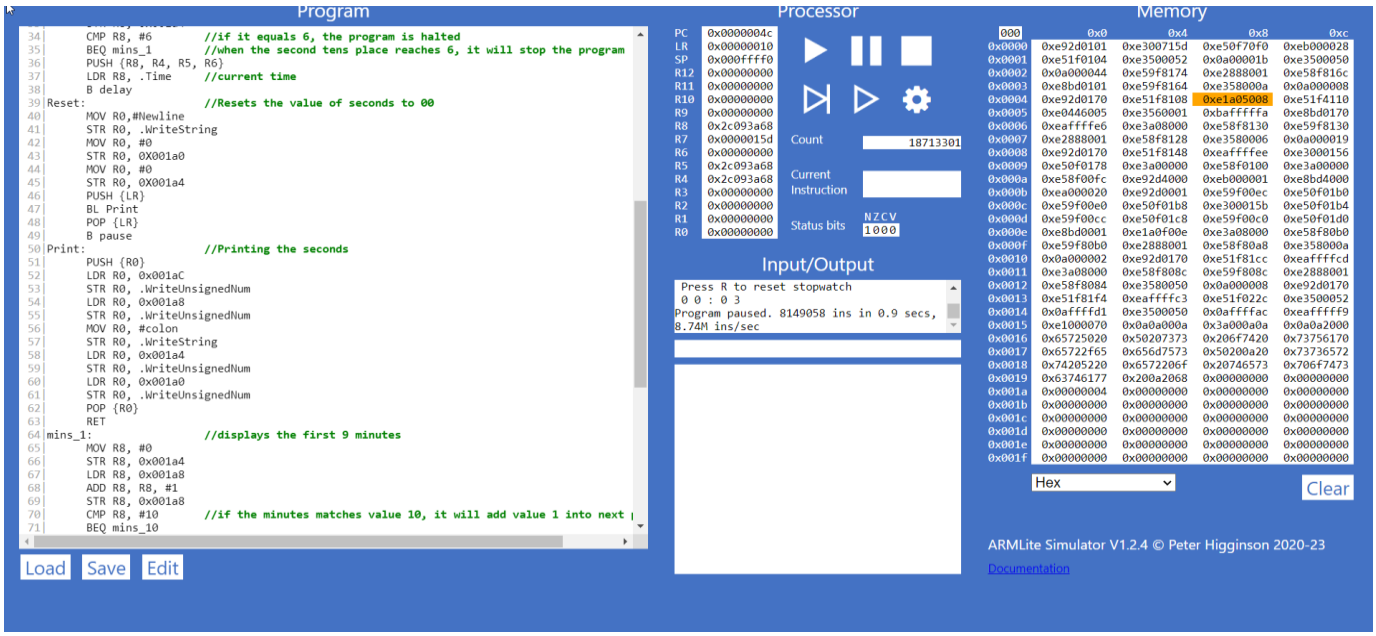

Fig 2.2

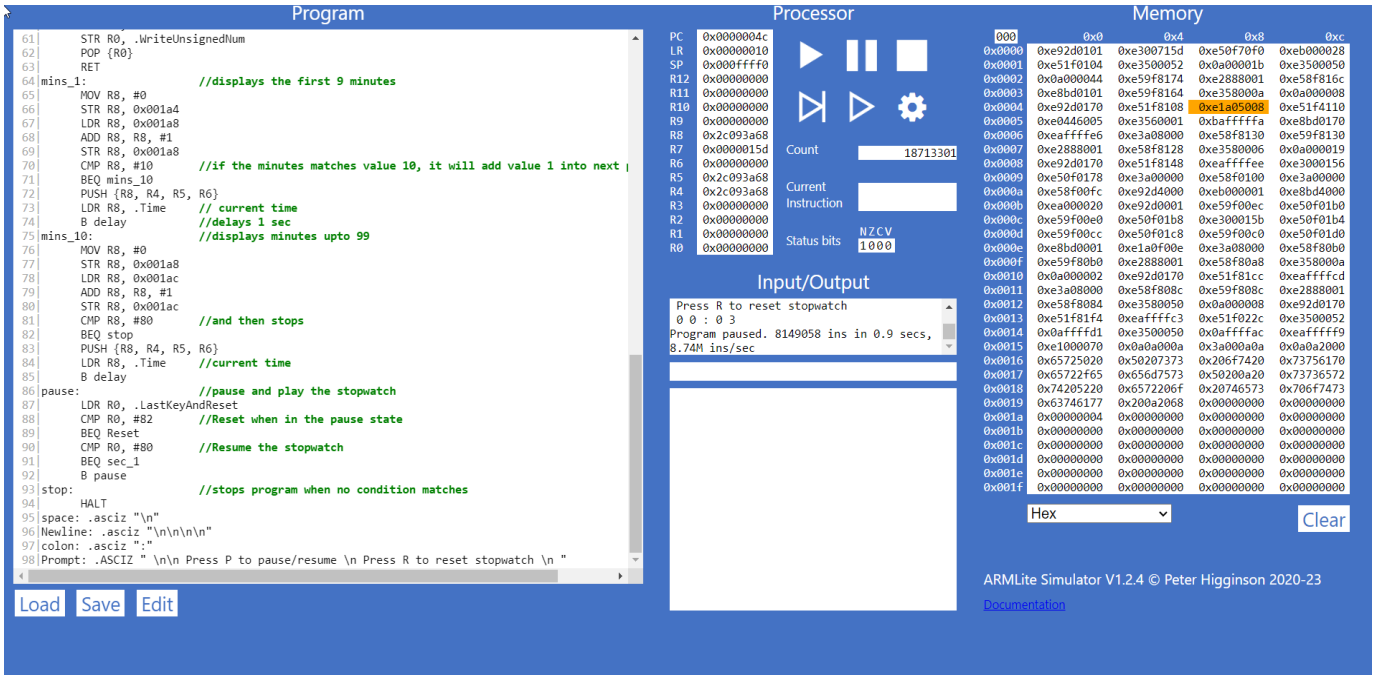
Program
Processor
Memory
Input/Output
Load
Save
Edit
Count
Current Instruction
Status bits
NZCV
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

Fig 2.3

STAGE 3

The stage 3 is the compilation of all the above stages with a split time included. When ‘s’ key is pressed for slipt time the program displays it in the text output box a line with a split time written and the time captured. The program is implemented in a way so that it can show or retrieve previous values. It also has the string constant showing users the messages and instructions. It could do till stage 3 it is working however it is not perfect Fig(3.1-3.4)

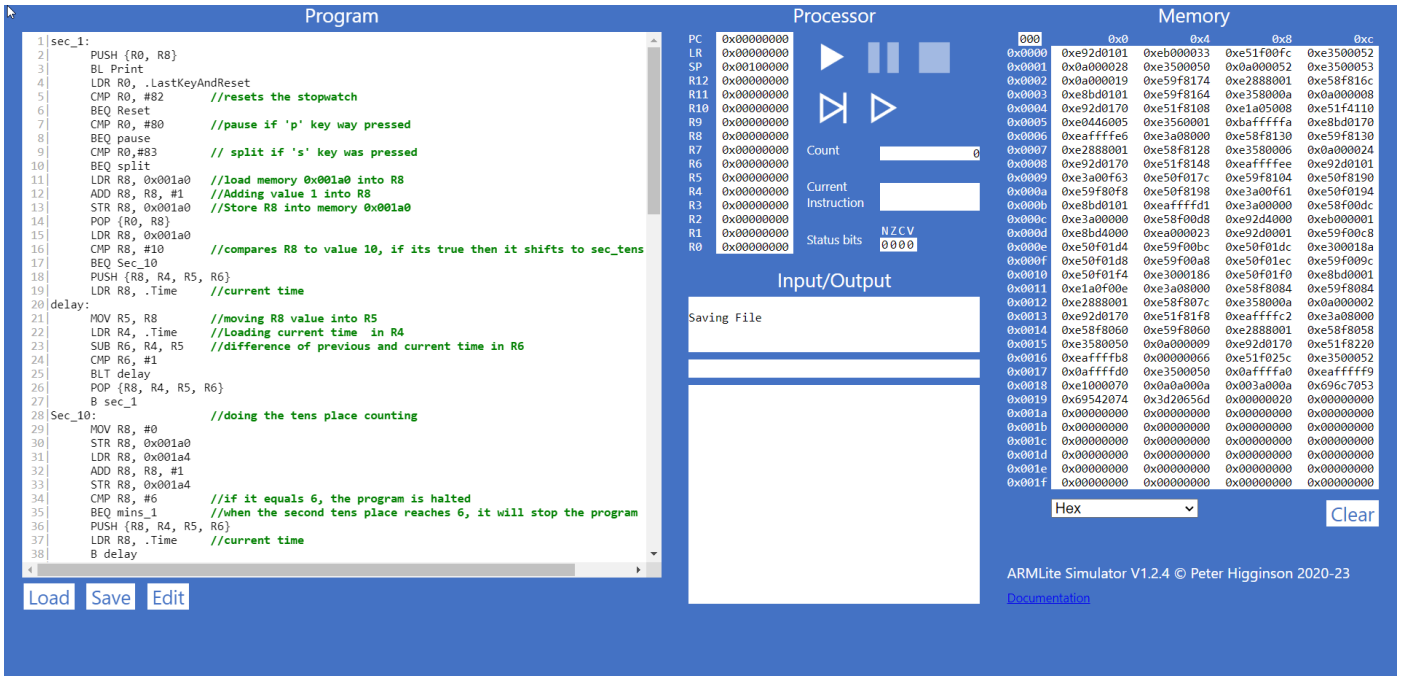

Fig 3.1

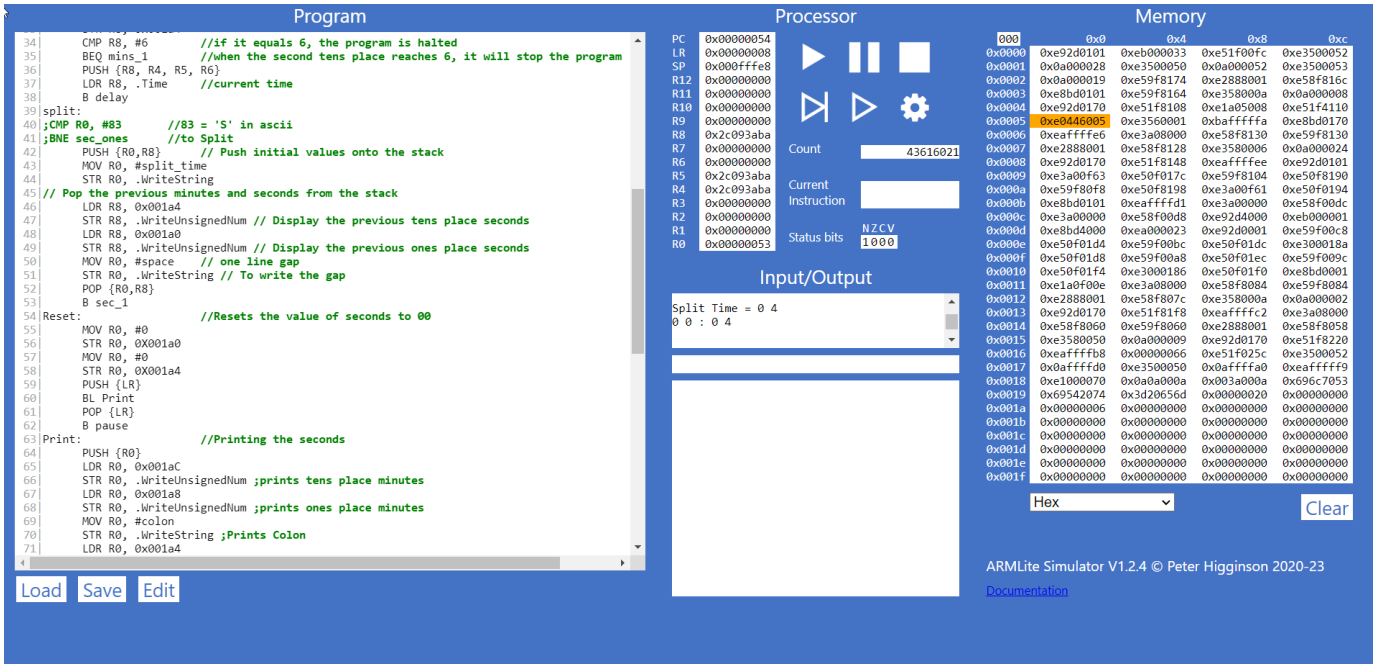
Program
CMP R8, #6 //if it equals 6, the program is halted
BEQ mins_1 //when the second tens place reaches 6, it will stop the program
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B delay
split:
;CMP R0, #83 //83 = 'S' in ascii
;BNE sec_ones //to Split
PUSH {R0,R8} // Push initial values onto the stack
MOV R0, #split_time
STR R0, .WriteString
// Pop the previous minutes and seconds from the stack
LDR R8, 0x001a4
STR R8, .WriteUnsignedNum // Display the previous tens place seconds
LDR R8, 0x001a0
STR R8, .WriteUnsignedNum // Display the previous ones place seconds
MOV R0, #space // one line gap
STR R0, .WriteString // To write the gap
POP {R0,R8}
B sec_1
Reset: //Resets the value of seconds to 00
MOV R0, #0
STR R0, 0X001a0
MOV R0, #0
STR R0, 0X001a4
PUSH {LR}
BL Print
POP {LR}
B pause
Print: //Printing the seconds
PUSH {R0}
LDR R0, 0x001aC
STR R0, .WriteUnsignedNum ;prints tens place minutes
LDR R0, 0x001a8
STR R0, .WriteUnsignedNum ;prints ones place minutes
MOV R0, #colon
STR R0, .WriteString ;Prints Colon
LDR R0, 0x001a4
Load
Save
Edit
Processor
PC 0x00000054
LR 0x00000008
SP 0x000fffe8
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x2c093aba
R7 0x00000000
R6 0x00000000
R5 0x2c093aba
R4 0x2c093aba
R3 0x00000000
R2 0x00000000
R1 0x00000000
R0 0x00000053
Count 43616021
Current Instruction
Status bits NZCV 1000
Input/Output
Split Time = 0 4
0 0 : 0 4
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe92d0101 0xeb000033 0xe51f00fc 0xe3500052
0x0001 0x0a000028 0xe3500050 0x0a000052 0xe3500053
0x0002 0x0a000019 0xe59f8174 0xe2888001 0xe58f816c
0x0003 0xe8bd0101 0xe59f8164 0xe358000a 0x0a000008
0x0004 0xe92d0170 0xe51f8108 0xe1a05008 0xe51f4110
0x0005 0xe0446005 0xe3560001 0xbaffffa 0xe8bd0170
0x0006 0xeaffffe6 0xe3a08000 0xe58f8130 0xe59f8130
0x0007 0xe2888001 0xe58f8128 0xe3580006 0x0a000024
0x0008 0xe92d0170 0xe51f8148 0xeaffffee 0xe92d0101
0x0009 0xe3a00f63 0xe50f017c 0xe59f8104 0xe50f8190
0x000a 0xe59f80f8 0xe50f8198 0xe3a00f61 0xe50f0194
0x000b 0xe8bd0101 0xeafffffd1 0xe3a00000 0xe58f00dc
0x000c 0xe3a00000 0xe58f00d8 0xe92d4000 0xeb000001
0x000d 0xe8bd4000 0xea000023 0xe92d0001 0xe59f00c8
0x000e 0xe50f01d4 0xe59f00bc 0xe50f01dc 0xe300018a
0x000f 0xe50f01d8 0xe59f00a8 0xe50f01ec 0xe59f009c
0x0010 0xe50f01f4 0xe3000186 0xe50f01f0 0xe8bd0001
0x0011 0xe1a0f00e 0xe3a08000 0xe58f8084 0xe59f8084
0x0012 0xe2888001 0xe58f807c 0xe358000a 0x0a000002
0x0013 0xe92d0170 0xe51f81f8 0xeafffffc2 0xe3a08000
0x0014 0xe58f8060 0xe59f8060 0xe2888001 0xe58f8058
0x0015 0xe3580050 0x0a000009 0xe92d0170 0xe51f8220
0x0016 0xeaffffb8 0x00000066 0xe51f025c 0xe3500052
0x0017 0x0afffffd0 0xe3500050 0x0afffa0 0xeafffff9
0x0018 0xe1000070 0x0a0a000a 0x003a000a 0x696c7053
0x0019 0x69542074 0x3d20656d 0x00000020 0x00000000
0x001a 0x00000006 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

Fig 3.2

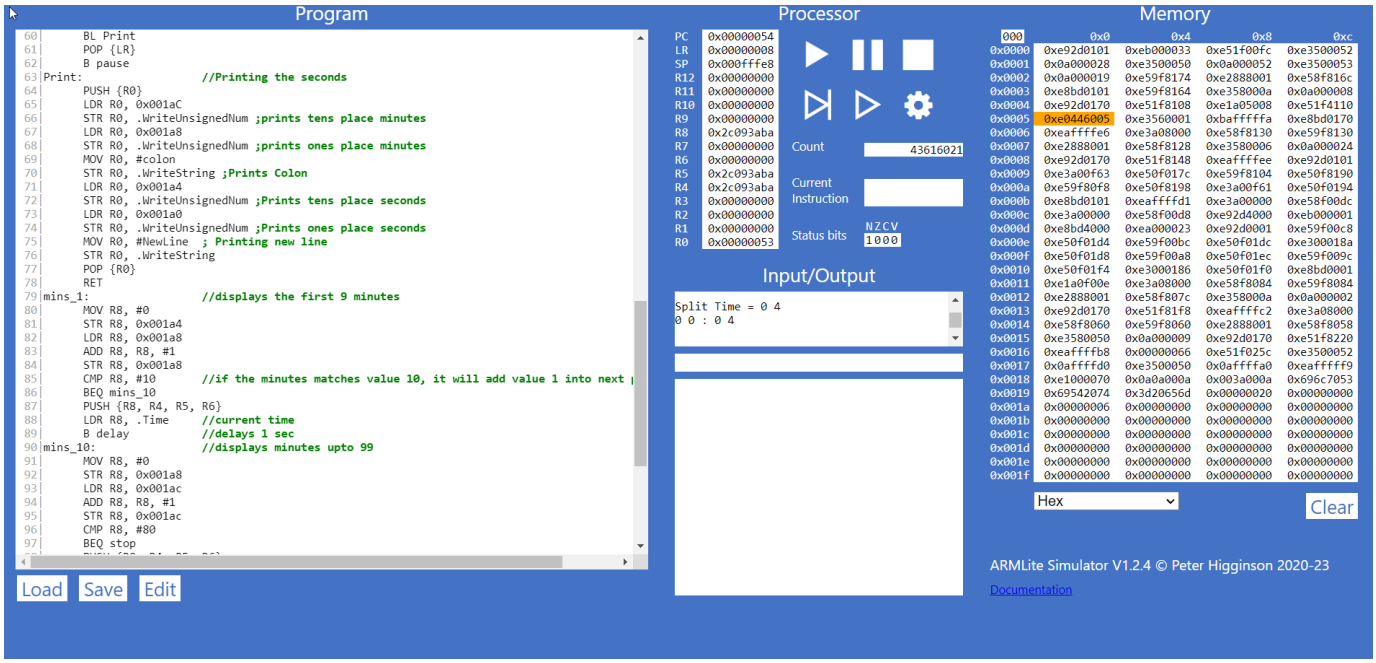
Program
BL Print
POP {LR}
B pause
Print: //Printing the seconds
PUSH {R0}
LDR R0, 0x001aC
STR R0, .WriteUnsignedNum ;prints tens place minutes
LDR R0, 0x001a8
STR R0, .WriteUnsignedNum ;prints ones place minutes
MOV R0, #colon
STR R0, .WriteString ;Prints Colon
LDR R0, 0x001a4
STR R0, .WriteUnsignedNum ;Prints tens place seconds
LDR R0, 0x001a0
STR R0, .WriteUnsignedNum ;Prints ones place seconds
MOV R0, #NewLine ; Printing new line
STR R0, .WriteString
POP {R0}
RET
mins_1: //displays the first 9 minutes
MOV R8, #0
STR R8, 0x001a4
LDR R8, 0x001a8
ADD R8, R8, #1
STR R8, 0x001a8
CMP R8, #10 //if the minutes matches value 10, it will add value 1 into next
BEQ mins_10
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B delay //delays 1 sec
mins_10: //displays minutes upto 99
MOV R8, #0
STR R8, 0x001a8
LDR R8, 0x001ac
ADD R8, R8, #1
STR R8, 0x001ac
CMP R8, #80
BEQ stop
Load
Save
Edit
Processor
PC 0x00000054
LR 0x00000008
SP 0x000fffe8
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x2c093aba
R7 0x00000000
R6 0x00000000
R5 0x2c093aba
R4 0x2c093aba
R3 0x00000000
R2 0x00000000
R1 0x00000000
R0 0x00000053
Count 43616021
Current Instruction
Status bits NZCV 1000
Input/Output
Split Time = 0 4
0 0 : 0 4
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe92d0101 0xeb000033 0xe51f00fc 0xe3500052
0x0001 0x0a000028 0xe3500050 0x0a000052 0xe3500053
0x0002 0x0a000019 0xe59f8174 0xe2888001 0xe58f816c
0x0003 0xe8bd0101 0xe59f8164 0xe358000a 0x0a000008
0x0004 0xe92d0170 0xe51f8108 0xe1a05008 0xe51f4110
0x0005 0xe0446005 0xe3560001 0xbaffffa 0xe8bd0170
0x0006 0xeaffffe6 0xe3a08000 0xe58f8130 0xe59f8130
0x0007 0xe2888001 0xe58f8128 0xe3580006 0x0a000024
0x0008 0xe92d0170 0xe51f8148 0xeaffffee 0xe92d0101
0x0009 0xe3a00f63 0xe50f017c 0xe59f8104 0xe50f8190
0x000a 0xe59f80f8 0xe50f8198 0xe3a00f61 0xe50f0194
0x000b 0xe8bd0101 0xeafffffd1 0xe3a00000 0xe58f00dc
0x000c 0xe3a00000 0xe58f00d8 0xe92d4000 0xeb000001
0x000d 0xe8bd4000 0xea000023 0xe92d0001 0xe59f00c8
0x000e 0xe50f01d4 0xe59f00bc 0xe50f01dc 0xe300018a
0x000f 0xe50f01d8 0xe59f00a8 0xe50f01ec 0xe59f009c
0x0010 0xe50f01f4 0xe3000186 0xe50f01f0 0xe8bd0001
0x0011 0xe1a0f00e 0xe3a08000 0xe58f8084 0xe59f8084
0x0012 0xe2888001 0xe58f807c 0xe358000a 0x0a000002
0x0013 0xe92d0170 0xe51f81f8 0xeafffffc2 0xe3a08000
0x0014 0xe58f8060 0xe59f8060 0xe2888001 0xe58f8058
0x0015 0xe3580050 0x0a000009 0xe92d0170 0xe51f8220
0x0016 0xeaffffb8 0x00000066 0xe51f025c 0xe3500052
0x0017 0x0afffffd0 0xe3500050 0x0afffa0 0xeafffff9
0x0018 0xe1000070 0x0a0a000a 0x003a000a 0x696c7053
0x0019 0x69542074 0x3d20656d 0x00000020 0x00000000
0x001a 0x00000006 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

Fig 3.3

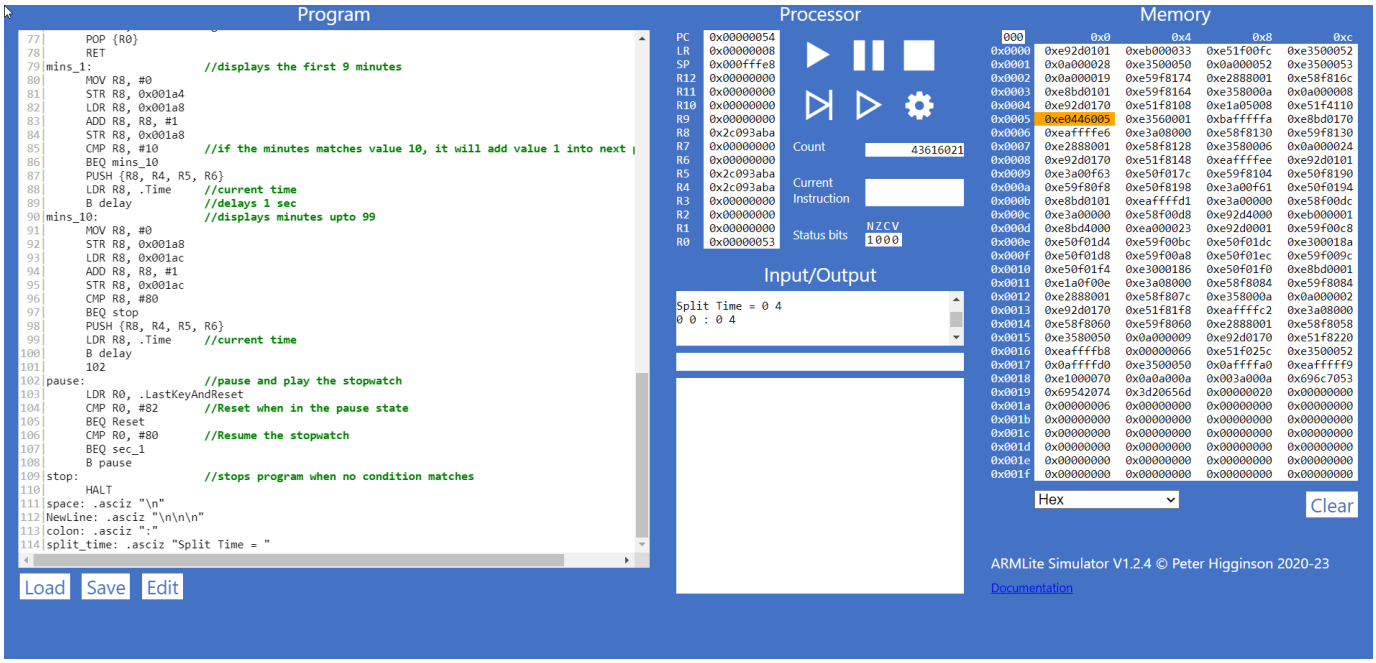
Program
POP {R0}
RET
mins_1: //displays the first 9 minutes
MOV R8, #0
STR R8, 0x001a4
LDR R8, 0x001a8
ADD R8, R8, #1
STR R8, 0x001a8
CMP R8, #10 //if the minutes matches value 10, it will add value 1 into next
BEQ mins_10
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B delay //delays 1 sec
mins_10: //displays minutes upto 99
MOV R8, #0
STR R8, 0x001a8
LDR R8, 0x001ac
ADD R8, R8, #1
STR R8, 0x001ac
CMP R8, #80
BEQ stop
PUSH {R8, R4, R5, R6}
LDR R8, .Time //current time
B delay
102
pause: //pause and play the stopwatch
LDR R0, .LastKeyAndReset
CMP R0, #82 //Reset when in the pause state
BEQ Reset
CMP R0, #80 //Resume the stopwatch
BEQ sec_1
B pause
stop: //stops program when no condition matches
HALT
space: .asciz "\n"
NewLine: .asciz "\n\n\n"
colon: .asciz ":"
split_time: .asciz "Split Time = "
Load
Save
Edit
Processor
PC 0x00000054
LR 0x00000008
SP 0x000fffe8
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x2c093aba
R7 0x00000000
R6 0x00000000
R5 0x2c093aba
R4 0x2c093aba
R3 0x00000000
R2 0x00000000
R1 0x00000000
R0 0x00000053
Count 43616021
Current Instruction
Status bits NZCV 1000
Input/Output
Split Time = 0 4
0 0 : 0 4
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe92d0101 0xeb000033 0xe51f00fc 0xe3500052
0x0001 0x0a000028 0xe3500050 0x0a000052 0xe3500053
0x0002 0x0a000019 0xe59f8174 0xe2888001 0xe58f816c
0x0003 0xe8bd0101 0xe59f8164 0xe358000a 0x0a000008
0x0004 0xe92d0170 0xe51f8108 0xe1a05008 0xe51f4110
0x0005 0xe0446005 0xe3560001 0xbaffffa 0xe8bd0170
0x0006 0xeaffffe6 0xe3a08000 0xe58f8130 0xe59f8130
0x0007 0xe2888001 0xe58f8128 0xe3580006 0x0a000024
0x0008 0xe92d0170 0xe51f8148 0xeaffffee 0xe92d0101
0x0009 0xe3a00f63 0xe50f017c 0xe59f8104 0xe50f8190
0x000a 0xe59f80f8 0xe50f8198 0xe3a00f61 0xe50f0194
0x000b 0xe8bd0101 0xeafffd1 0xe3a00000 0xe58f00dc
0x000c 0xe3a00000 0xe58f00d8 0xe92d4000 0xeb000001
0x000d 0xe8bd4000 0xea000023 0xe92d0001 0xe59f00c8
0x000e 0xe50f01d4 0xe59f00bc 0xe50f01dc 0xe300018a
0x000f 0xe50f01d8 0xe59f00a8 0xe50f01ec 0xe59f009c
0x0010 0xe50f01f4 0xe3000186 0xe50f01f0 0xe8bd0001
0x0011 0xe1a0f00e 0xe3a08000 0xe58f8084 0xe59f8084
0x0012 0xe2888001 0xe58f807c 0xe358000a 0x0a000002
0x0013 0xe92d0170 0xe51f81f8 0xeaffffc2 0xe3a08000
0x0014 0xe58f8060 0xe59f8060 0xe2888001 0xe58f8058
0x0015 0xe3580050 0x0a000009 0xe92d0170 0xe51f8220
0x0016 0xeaffffb8 0x00000066 0xe51f025c 0xe3500052
0x0017 0x0afffd0 0xe3500050 0x0afffa0 0xeafffff9
0x0018 0xe1000070 0x0a0a000a 0x003a000a 0x696c7053
0x0019 0x69542074 0x3d20656d 0x00000020 0x00000000
0x001a 0x00000006 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

Fig 3.4