ARM Assembly for Embedded Applications *sth edition* DANIEL W LEWIS

ARM Instructions Worksheet #9

Floating-Point Compares

And their effect on the NZCV Flags in the CPSR register:

N	7	C	V	0		
1	30	29	28	27	26	

Prerequisite Reading: Chapter 9 Revised: April 21, 2020

Objectives: To use the web-based simulator ("CPULator") to better understand.

- $1. \quad \mbox{The use of VCMP and VMRS to perform floating-point comparisons.}$
- 2. The use of VSUB and VMOV to simplify some floating-point comparisons.
- 3. The use of floating-point equality comparisons.

To do offline: Answer the questions that follow the listing below. (Numbers at far left are memory addresses.)

.global _start // *** EXECUTION STARTS HERE *** 00000000 _start: MOVS R0,0 // N flag = 0 00000004 _start: MOVS S0,posPt4 // S0 = +0.4 00000000 _start: MOVS S1,posPt5 // S1 = +0.5 00000000 VLDR S1,posPt5 // 0.4 < 0.5 ? 00000010 VMRS APSR_nzcv,FPSCR 00000011 LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000021 LDR R1,S2 // Same as R0? 00000022 VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000032 UDR R2,=1 // Assume EQ 00000032 BEQ done // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 00000040 point4: .float +0.4			syntax	unified	
// *** EXECUTION STARTS HERE *** 00000000 _start: MOVS R0,0 // N flag = 0 00000004 VLDR S0,posPt4 // S0 = +0.4 00000008 VLDR S1,posPt5 // S1 = +0.5 00000000 VCMP.F32 S0,S1 // 0.4 < 0.5 ?			global	start	
<pre>// *** EXECUTION STARTS HERE *** 00000000 _start: MOVS R0,0 // N flag = 0 0000004 VLDR S0,posPt4 // S0 = +0.4 00000008 VLDR S1,posPt5 // S1 = +0.5 0000000C VCMP.F32 S0,S1 // 0.4 < 0.5 ? 00000014 LDR R0,=1 // Assume MI 00000018 BMI L1 0000001C LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000028 LSR R1,R1,31 // Same as R0? 00000028 LSR R1,R1,31 // Same as R0? 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 UMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000034 UMRS APSR_nczv,FPSCR 00000034 UDR R2,=1 // Assume EQ 00000044 done: B done // Infinite loop 00000044 done: B done // Infinite loop</pre>			•8100a1		
00000000 _start: MOVS R0,0 // N flag = 0 00000000 VLDR S0,posPt4 // S0 = +0.4 00000000 VLDR S1,posPt5 // S1 = +0.5 00000000 VCMP.F32 S0,S1 // 0.4 < 0.5 ?		// *** EXE	CUTION STARTS	HERE ***	
00000004 VLDR S0,posPt4 // S0 = +0.4 00000008 VLDR S1,posPt5 // S1 = +0.5 00000000 VCMP.F32 S0,S1 // 0.4 < 0.5 ?	00000000	_start:	MOVS	R0,0	// N flag = 0
00000008 VLDR S1,posPt5 // S1 = +0.5 000000C VCMP.F32 S0,S1 // 0.4 < 0.5 ?	00000004	_	VLDR	S0,posPt4	// S0 = +0.4
0000000C VCMP.F32 S0,S1 // 0.4 < 0.5 ?	0000008		VLDR	S1,posPt5	// S1 = +0.5
00000010 VMRS APSR_nzcv,FPSCR 00000014 LDR R0,=1 // Assume MI 00000018 BMI L1 00000010 LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 00000028 LSR R1,R1,31 // Same as R0? 00000020 VLDR S3,negPt1 // S3 = -0.1 00000026 VCMP.F32 S2,S3 // S2 == S3 ? 00000030 VCMP.F32 S2,S3 // Assume EQ 00000038 LDR R2,=1 // Assume EQ 00000030 BEQ done // Wasn't EQ 00000040 LDR R2,=0 // Infinite loop 00000044 done: B done // Infinite loop 00000044 ifloat +0.5 00000042 point1: .float +0.4 00000050 point1: .float -0.1	000000C		VCMP.F32	S0,S1	// 0.4 < 0.5 ?
00000014 LDR R0,=1 // Assume MI 00000018 BMI L1 00000010 LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 00000028 S3,negPt1 // S3 = -0.1 00000020 VLDR S3,negPt1 // S3 = -0.1 00000030 00000020 VLDR S3,negPt1 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000030 DR R2,=0 // Wasn't EQ 00000040 LDR R2,=0 // Infinite loop 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 .float +0.4 00000048 point4: .float +0.4 .float -0.1	00000010		VMRS	APSR_nzcv,FPSCR	
00000018 BMI L1 0000001C LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 00000028 // S3 = -0.1 00000020 VLDR S3,negPt1 // S3 = -0.1 00000020 VLDR S2,S3 // S2 == S3 ? 00000030 VCMP.F32 S2,S3 // Assume EQ 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000030 DR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 00000042 point4: .float +0.4 00000050 point1: .float -0.1	00000014		LDR	R0,=1	// Assume MI
0000001C LDR R0,=0 // Wasn't MI 00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 // Same as R0? 00000028 LSR R1,R1,31 // Same as R0? 00000020 VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 // Infinite loop 00000045 point4: .float +0.4 00000050	00000018		BMI	L1	
00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 // Same as R0? 00000028 LSR R1,R1,31 // Same as R0? 00000020 VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000030 BEQ done 00000040 LDR R2,=1 // Assume EQ 00000044 done: B done // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 00000045 point4: .float +0.4 00000050 point1: .float -0.1	0000001C		LDR	R0,=0	// Wasn't MI
00000020 L1: VSUB.F32 S2,S0,S1 // S2 = 0.4 - 0.5 00000024 VMOV R1,S2 // Same as R0? 00000028 LSR R1,R1,31 // Same as R0? 00000020 VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR // Assume EQ 00000030 BEQ done // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 00000042 point4: .float +0.4 00000050 point1: .float -0.1				·	
00000024 VMOV R1,S2 00000028 LSR R1,R1,31 // Same as R0? 00000020 VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000030 BEQ done 00000044 done: B done 00000044 done: B done 00000044 point5: .float +0.5 00000042 point4: .float +0.4 00000050 point1: .float -0.1	00000020	L1:	VSUB.F32	S2,S0,S1	// S2 = 0.4 - 0.5
00000028 LSR R1,R1,31 // Same as R0? 0000002C VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 . 00000040 point1: .float +0.4 .	00000024		VMOV	R1,S2	
0000002C VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 0000003C BEQ done 00000044 done: B done 00000044 done: B done 00000044 done: B done 00000044 done: B done 00000044 done: Acceleration 00000045 point5: .float +0.5 00000046 point4: .float +0.4 00000050 point1: .float -0.1	00000028		LSR	R1,R1,31	// Same as R0?
0000002C VLDR S3,negPt1 // S3 = -0.1 00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000030 BEQ done 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000044 point5: .float +0.5 . 00000040 point4: .float +0.4 . 00000050 point1: .float -0.1					
00000030 VCMP.F32 S2,S3 // S2 == S3 ? 00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 00000030 BEQ done 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 00000040 point4: .float +0.4 00000050 point1: .float -0.1	0000002C		VLDR	S3,negPt1	// S3 = -0.1
00000034 VMRS APSR_nczv,FPSCR 00000038 LDR R2,=1 // Assume EQ 0000003C BEQ done 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 00000040 point1: .float -0.1	0000030		VCMP.F32	S2,S3	// S2 == S3 ?
00000038 LDR R2,=1 // Assume EQ 0000003C BEQ done // Wasn't EQ 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 00000042 point4: .float +0.4 00000050 point1: .float -0.1	00000034		VMRS	APSR_nczv,FPSCR	
0000003C BEQ done 00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1	0000038		LDR	R2,=1	// Assume EQ
00000040 LDR R2,=0 // Wasn't EQ 00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1	000003C		BEQ	done	-
00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1	00000040		LDR	R2,=0	// Wasn't EQ
00000044 done: B done // Infinite loop 00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1				-	-
00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1	00000044	done:	В	done	// Infinite loop
00000048 point5: .float +0.5 0000004C point4: .float +0.4 00000050 point1: .float -0.1					· · · · · · · · · · · · · · · · · · ·
0000004C point4: .float +0.4 00000050 point1: .float -0.1	00000048	point5:	.float	+0.5	
00000050 point1: .float -0.1	0000004C	point4:	.float	+0.4	
	00000050	point1:	.float	-0.1	
		-			
.end			.end		

What is in the N flag (CPSR bit 31) after executing the VCMP at address $000000C_{16}$? What is in the N flag (CPSR bit 31) after executing the VMRS at address 00000010_{16} ? What is in register R0 *before* executing the VSUB instruction at address 00000020_{16} ? What is in register S2 after executing the VSUB instruction at address 00000020_{16} ? What is in register R1 after executing the VMOV instruction at address 00000024_{16} ? What is in register R1 after executing the LSR instruction at address 00000028_{16} ? What is in register S3 after executing the VLDR instruction at address 00000028_{16} ? What is in the Z flag (CPSR bit 29) after executing the VMRS at address 00000034_{16} ? What is in register R2 *before* executing the B instruction at address 00000034_{16} ?



Getting ready: Now use the simulator to collect the following information and compare to your earlier answers.

1. Click here to open a browser for the ARM instruction simulator with pre-loaded code.

Note: You can change the number format in the "Settings" window between hex, unsigned decimal and signed decimal as needed

Step 1: Press F2 once per ARM instruction as needed to see what the simulator says for the following:

What is in the N flag (CPSR bit 31) after executing the VCMP at address 0000000C₁₆? What is in the N flag (CPSR bit 31) after executing the VMRS at address 00000010₁₆? What is in register R0 *before* executing the VSUB instruction at address 00000020₁₆? What is in register S2 after executing the VSUB instruction at address 00000020₁₆? What is in register R1 after executing the VMOV instruction at address 00000024₁₆? What is in register R1 after executing the LSR instruction at address 00000028₁₆? What is in register S3 after executing the VLDR instruction at address 0000002C₁₆? What is in the Z flag (CPSR bit 29) after executing the VMRS at address 00000034₁₆?

