469 EMBEDDED SYSTEMS

Week 14

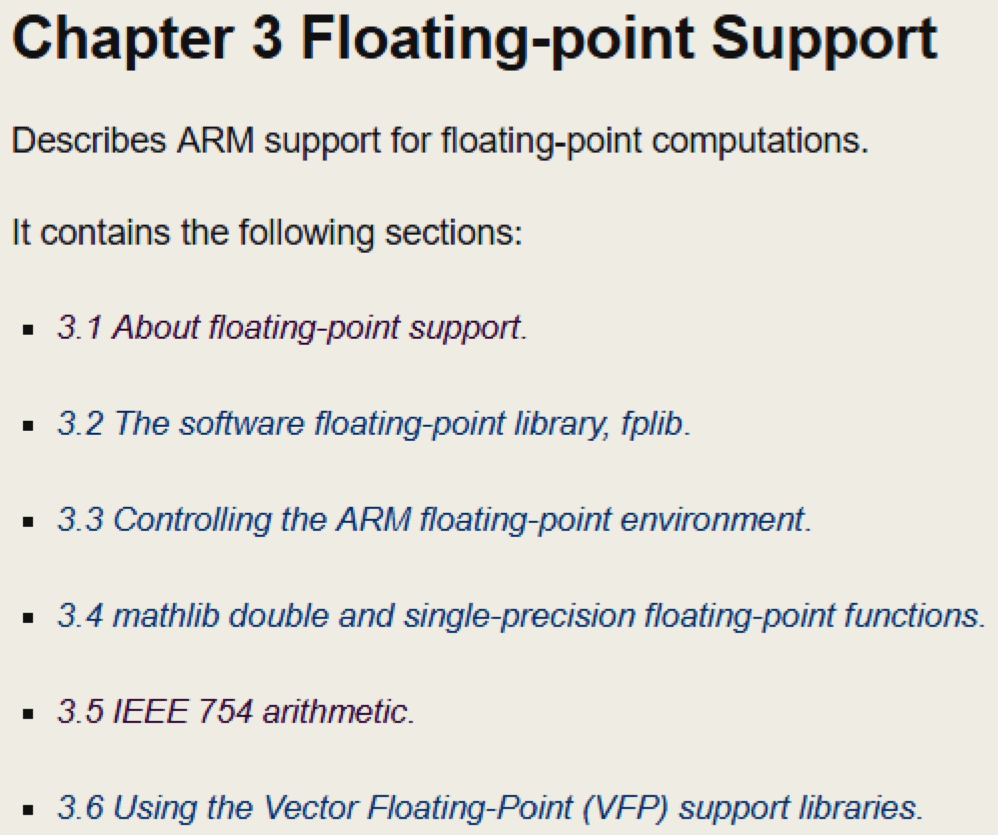
“VFP in Arm Assembly”

FPU usage in C

- Your C codes automatically use the FPU. Because you compilers are designed for it.
- So in Keil
  - If you compile for a system with a hardware VFP (Vector Floating Point) coprocessor (in other words an FPU, like in our TM4C123G),
    - the ARM compiler makes use of the FPU.
  - If you compile for a system without a coprocessor, the compiler implements the computations in software.
- So Keil handles everything for us.

FPU usage in Keil

- Please visit FP support page



Chapter 3 Floating-point Support

Describes ARM support for floating-point computations.

It contains the following sections:

- *3.1 About floating-point support.*
- *3.2 The software floating-point library, fplib.*
- *3.3 Controlling the ARM floating-point environment.*
- *3.4 mathlib double and single-precision floating-point functions.*
- *3.5 IEEE 754 arithmetic.*
- *3.6 Using the Vector Floating-Point (VFP) support libraries.*

FPU usage in Assembly

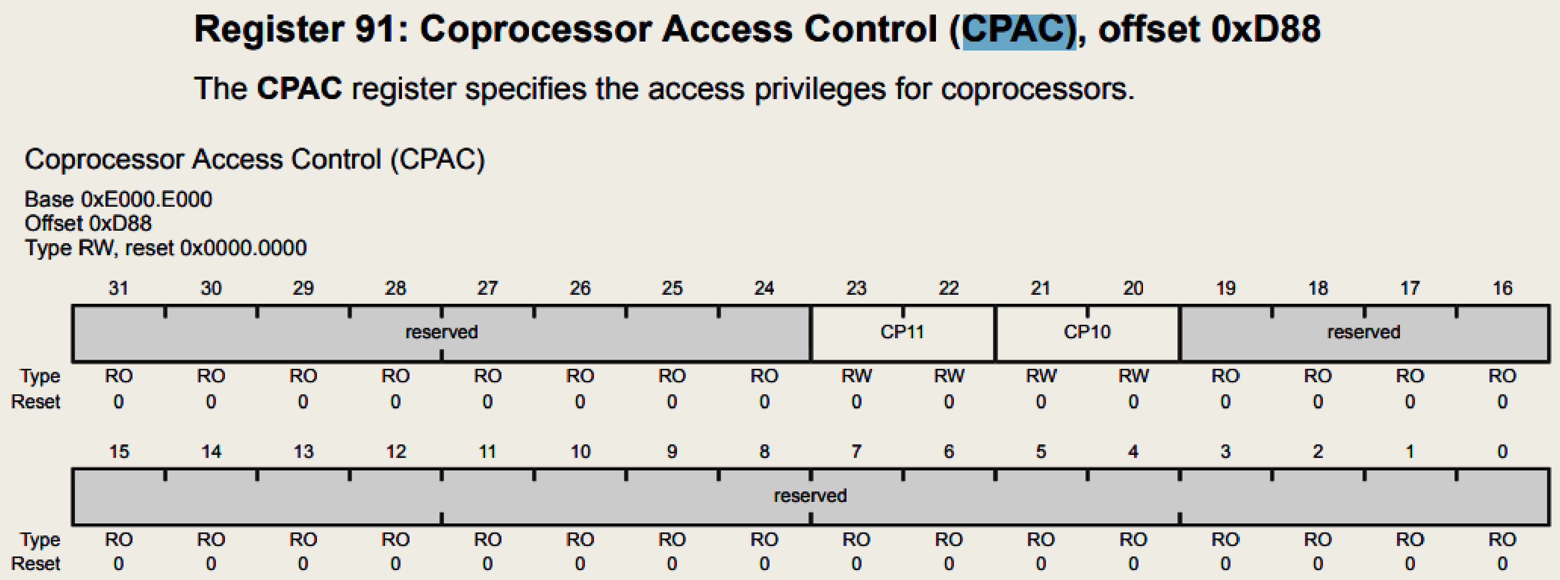
- In order to use the FPU in assembly, you will have to do it yourselves. No compiler is here to help you.
  - *First you enable it*
  - *Then use the necessary FPU assembly instructions*

Enabling the FPU

- The FPU is disabled from reset.
- You must enable it before you can use any floating-point instructions.
- The processor can read from and write to the Coprocessor Access Control (CPAC) register.
  - *The below example code sequence enables the FPU:*

```
; CPACR is located at address 0xE000ED88
LDR.W R0, =0xE000ED88
; Read CPACR
LDR R1, [R0]
; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR R1, R1, #(0xF << 20)
; Write back the modified value to the CPACR
STR R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB
```

Coprocessor Access Control (CPAC) Register (p.195)



Register 91: Coprocessor Access Control (CPAC), offset 0xD88

The **CPAC** register specifies the access privileges for coprocessors.

Coprocessor Access Control (CPAC)

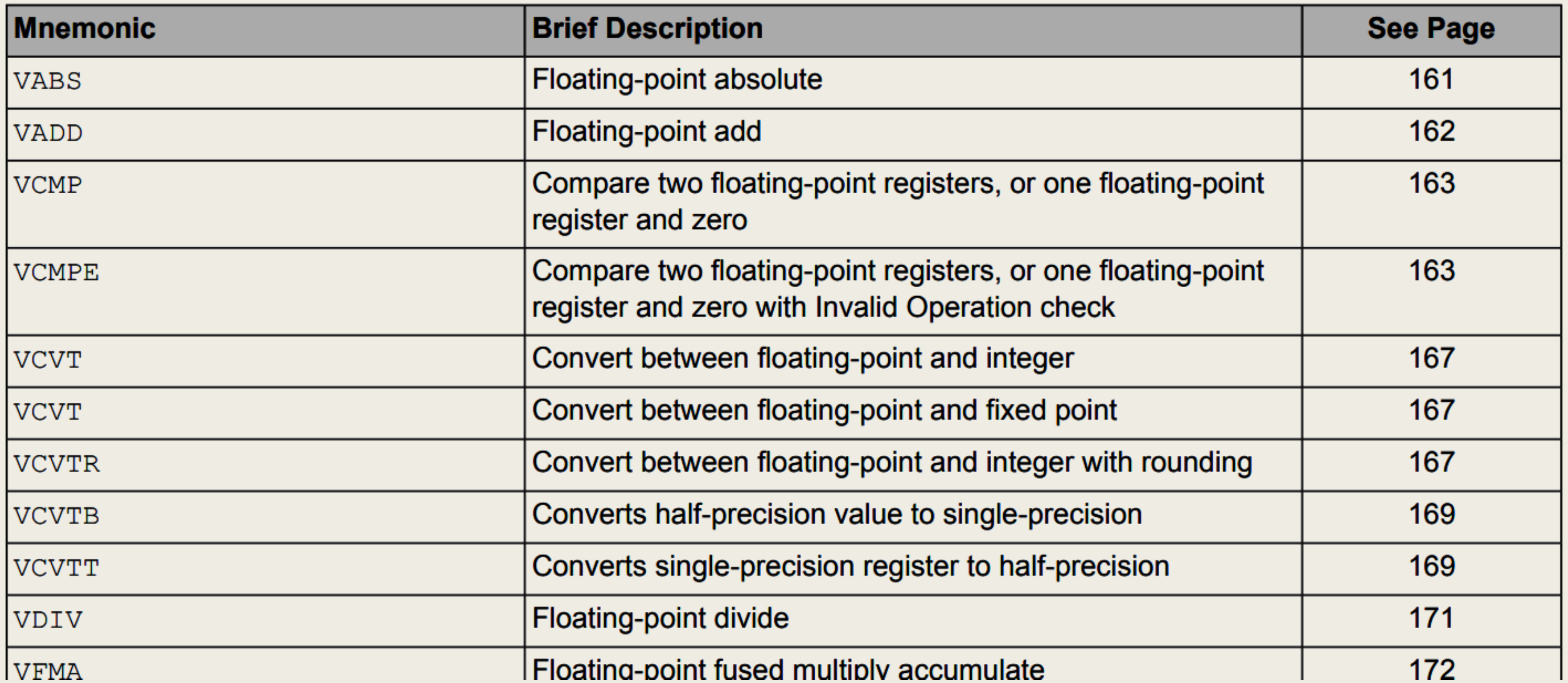
Base 0xE000.E000
Offset 0xD88
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								CP11		CP10		reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M4 FPU Assembly Instructions

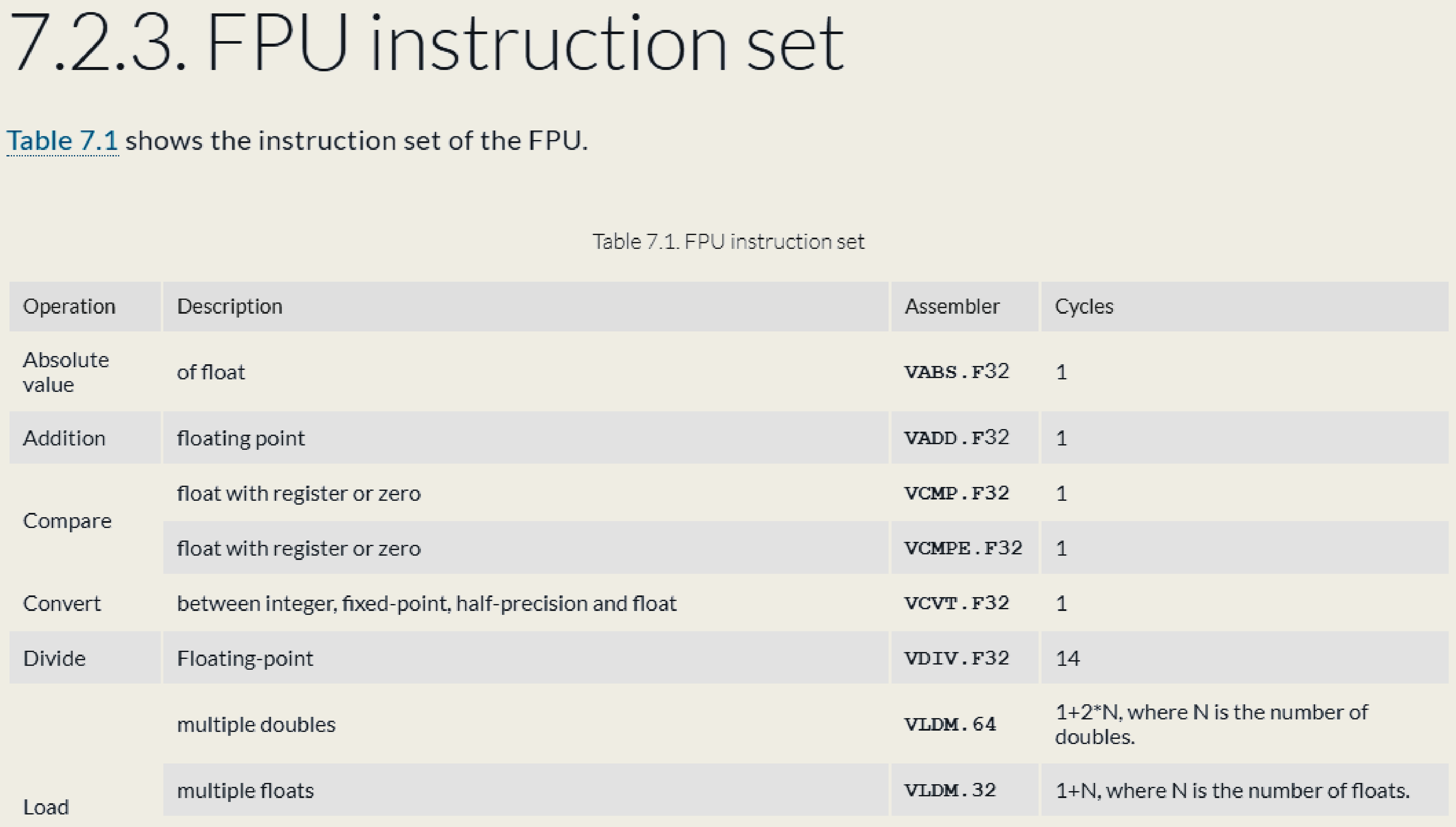
- Please check the [Arm Cortex M3/M4 instruction set]().
- On Page 159, you will find the FPU assembly instructions table:



Mnemonic	Brief Description	See Page
VABS	Floating-point absolute	161
VADD	Floating-point add	162
VCMP	Compare two floating-point registers, or one floating-point register and zero	163
VCMPE	Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check	163
VCVT	Convert between floating-point and integer	167
VCVT	Convert between floating-point and fixed point	167
VCVTR	Convert between floating-point and integer with rounding	167
VCVTB	Converts half-precision value to single-precision	169
VCVTT	Converts single-precision register to half-precision	169
VDIV	Floating-point divide	171
VFMA	Floating-point fused multiply accumulate	172

FPU Instruction durations

https://developer.arm.com/docs/ddi0439/b/floating-point-unit/fpu-functional-description/fpu-instruction-set



7.2.3. FPU instruction set

Table 7.1 shows the instruction set of the FPU.

Table 7.1. FPU instruction set

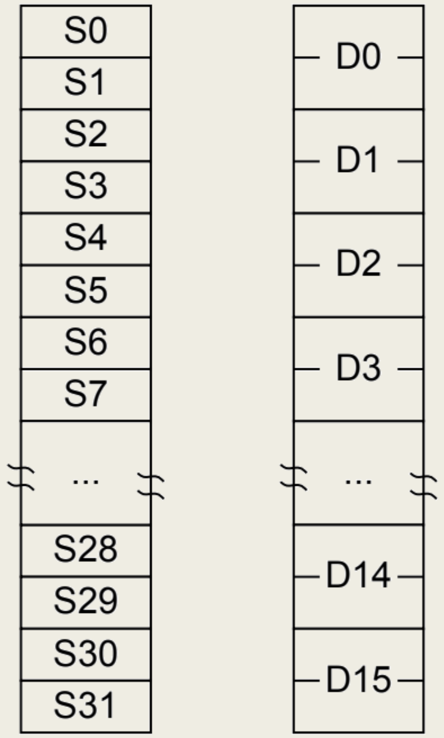
Operation	Description	Assembler	Cycles
Absolute value	of float	`VABS.F32`	1
Addition	floating point	`VADD.F32`	1
Compare	float with register or zero	`VCMP.F32`	1
	float with register or zero	`VCMPE.F32`	1
Convert	between integer, fixed-point, half-precision and float	`VCVT.F32`	1
Divide	Floating-point	`VDIV.F32`	14
Load	multiple doubles	`VLDM.64`	1+2*N, where N is the number of doubles.
	multiple floats	`VLDM.32`	1+N, where N is the number of floats.

FPU Instruction durations

Convert	between integer, fixed-point, half-precision and float	`VCVT.F32`	1
Divide	Floating-point	`VDIV.F32`	14
Load	multiple doubles	`VLDM.64`	1+2*N, where N is the number of doubles.
	multiple floats	`VLDM.32`	1+N, where N is the number of floats.
	single double	`VLDR.64`	3
	single float	`VLDR.32`	2
Move	top/bottom half of double to/from core register	`VMOV`	1
	immediate/float to float-register	`VMOV`	1
	two floats/one double to/from two core registers or one float to/from one core register	`VMOV`	2
	floating-point control/status to core register	`VMRS`	1
	core register to floating-point control/status	`VMSR`	1
Multiply	float	`VMUL.F32`	1
	then accumulate float	`VMLA.F32`	3
	then subtract float	`VMLS.F32`	3
	then accumulate then negate float	`VNMLA.F32`	3
	then subtract then negate float	`VNMLS.F32`	3

ARM M4 FPU Registers

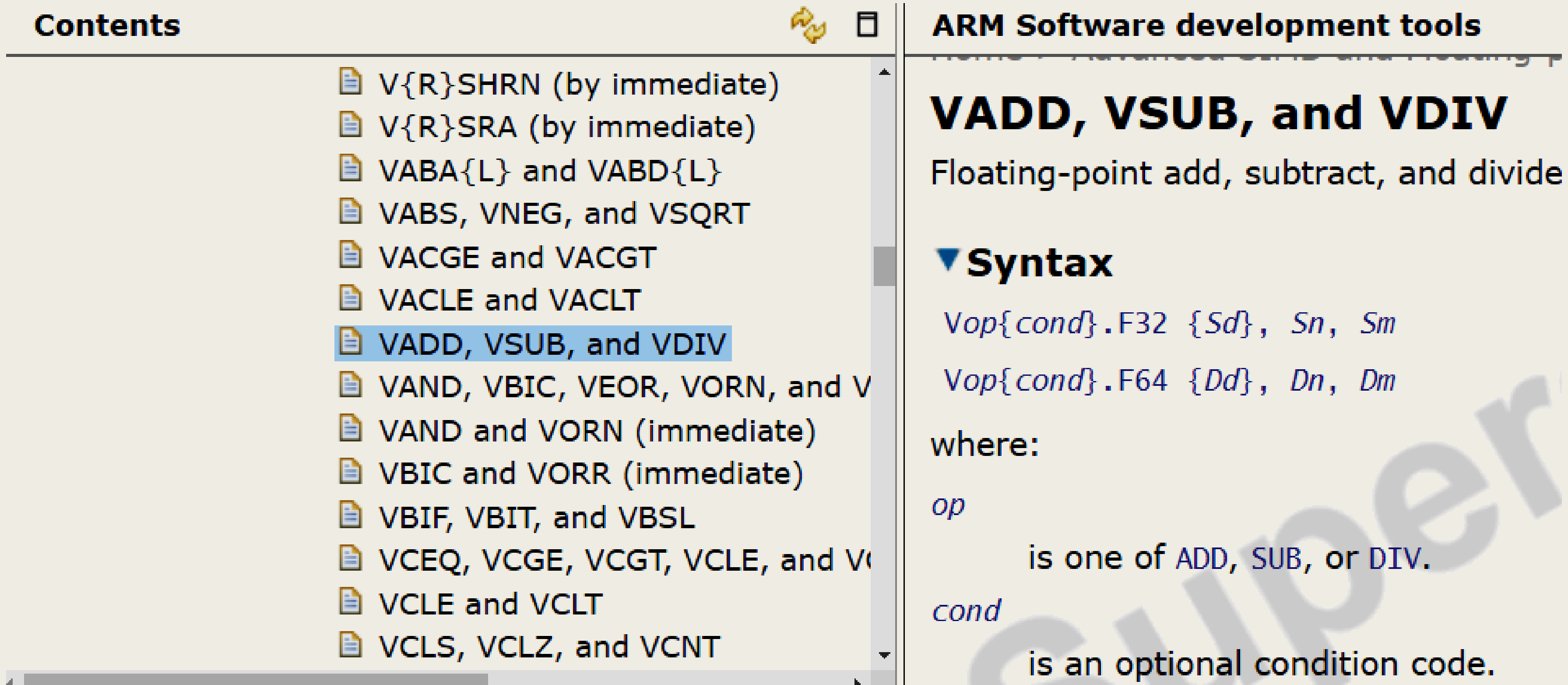
- The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:
  - *Sixteen 64-bit doubleword registers, D0-D15*
  - *Thirty-two 32-bit single-word registers, S0-S31*
  - *A combination of registers from the above views*



How to use them

- Same as fixed-point instructions

  http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0802a/Bcfchhif.html



Contents

- V{R}SHRN (by immediate)
- V{R}SRA (by immediate)
- VABA{L} and VABD{L}
- VABS, VNEG, and VSQRT
- VACGE and VACGT
- VACLE and VACLT
- VADD, VSUB, and VDIV
- VAND, VBIC, VEOR, VORN, and V
- VAND and VORN (immediate)
- VBIC and VORR (immediate)
- VBIF, VBIT, and VBSL
- VCEQ, VCGE, VCGT, VCLE, and V
- VCLE and VCLT
- VCLS, VCLZ, and VCNT

ARM Software development tools

VADD, VSUB, and VDIV

Floating-point add, subtract, and divide

Syntax

```
Vop{cond}.F32 {Sd}, Sn, Sm
Vop{cond}.F64 {Dd}, Dn, Dm
```

where:

op

is one of ADD, SUB, or DIV.

cond

is an optional condition code.