	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
		UNALLOCATED									**0**	**0**				
		BAD					invalid opera	0	0	0	0	0	0	0	0	0
		Branch,exception generatic									**1**	**0**	**1**			
		Compare _ Branch (immediat						**-**	**0**	**1**	**1**	**0**	**1**	**0**	**-**	
		CBZ		w		W		0	0	1	1	0	1	0	0	
		CBNZ		w		W		0	0	1	1	0	1	0	1	
		CBZ		x		X		1	0	1	1	0	1	0	0	
		CBNZ		x		X		1	0	1	1	0	1	0	1	
		Test bit & branch (immediate)						**b5**	**0**	**1**	**1**	**0**	**1**	**1**	**-**	
		TBZ						b5	0	1	1	0	1	1	0	
		TBNZ						b5	0	1	1	0	1	1	1	
		Conditional branch (immediat						**0**	**1**	**0**	**1**	**0**	**1**	**0**	**-**	
		B_cond						0	1	0	1	0	1	0	0	
		Exception generation						**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**-**
	//	SVC						1	1	0	1	0	1	0	0	0
	//	HVC						1	1	0	1	0	1	0	0	0
	//	SMC						1	1	0	1	0	1	0	0	0
		BRK		arm64	arm64		AArch64 Sp	1	1	0	1	0	1	0	0	0
	//	HLT						1	1	0	1	0	1	0	0	0
	//	DCPS1						1	1	0	1	0	1	0	0	1
	//	DCPS2						1	1	0	1	0	1	0	0	1
	//	DCPS3						1	1	0	1	0	1	0	0	1
	//	**System**						**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
	//	MSR		imm	imm			1	1	0	1	0	1	0	1	0
	//	HINT						1	1	0	1	0	1	0	1	0
	//	CLREX						1	1	0	1	0	1	0	1	0
	//	DSB						1	1	0	1	0	1	0	1	0
	//	DMB						1	1	0	1	0	1	0	1	0
	//	ISB						1	1	0	1	0	1	0	1	0
	//	SYS						1	1	0	1	0	1	0	1	0
	//	MSR						1	1	0	1	0	1	0	1	0
	//	SYSL						1	1	0	1	0	1	0	1	0
	//	MRS						1	1	0	1	0	1	0	1	0
		Unconditional branch (registe						**1**	**1**	**0**	**1**	**0**	**1**	**1**		**op**
		BR						1	1	0	1	0	1	1	0	0
		BLR						1	1	0	1	0	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
38		RET						1	1	0	1	0	1	1	0	0
39	//	ERET						1	1	0	1	0	1	1	0	1
40	//	DRPS						1	1	0	1	0	1	1	0	1
41	//	**Unconditional branch (immed**						**-**	**0**	**0**	**1**	**0**	**1**			
42		B						0	0	0	1	0	1			
43		BL						1	0	0	1	0	1			
44		**Loads and stores**										**1**		**0**		
45		**Load/store exclusive**						**-**	**-**	**0**	**0**	**1**	**0**	**0**	**0**	**-**
46		STXRB						0	0	0	0	1	0	0	0	0
47		STLXRB						0	0	0	0	1	0	0	0	0
48		LDXRB						0	0	0	0	1	0	0	0	0
49		LDAXRB						0	0	0	0	1	0	0	0	0
50		STLRB						0	0	0	0	1	0	0	0	1
51		LDARB						0	0	0	0	1	0	0	0	1
52		STXRH						0	1	0	0	1	0	0	0	0
53		STLXRH						0	1	0	0	1	0	0	0	0
54		LDXRH						0	1	0	0	1	0	0	0	0
55		LDAXRH						0	1	0	0	1	0	0	0	0
56		STLRH						0	1	0	0	1	0	0	0	1
57		LDARH						0	1	0	0	1	0	0	0	1
58		STXR		w		W		1	0	0	0	1	0	0	0	0
59		STLXR		w		W		1	0	0	0	1	0	0	0	0
60		STXP		w		W		1	0	0	0	1	0	0	0	0
61		STLXP		w		W		1	0	0	0	1	0	0	0	0
62		LDXR		w		W		1	0	0	0	1	0	0	0	0
63		LDAXR		w		W		1	0	0	0	1	0	0	0	0
64		LDXP		w		W		1	0	0	0	1	0	0	0	0
65		LDAXP		w		W		1	0	0	0	1	0	0	0	0
66		STLR		w		W		1	0	0	0	1	0	0	0	1
67		LDAR		w		W		1	0	0	0	1	0	0	0	1
68		STXR		x		X		1	1	0	0	1	0	0	0	0
69		STLXR		x		X		1	1	0	0	1	0	0	0	0
70		STXP		x		X		1	1	0	0	1	0	0	0	0
71		STLXP		x		X		1	1	0	0	1	0	0	0	0
72		LDXR		x		X		1	1	0	0	1	0	0	0	0
73		LDAXR		x		X		1	1	0	0	1	0	0	0	0
74		LDXP		x		X		1	1	0	0	1	0	0	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
75		LDAXP		x		X		1	1	0	0	1	0	0	0	0
76		STLR		x		X		1	1	0	0	1	0	0	0	1
77		LDAR		x		X		1	1	0	0	1	0	0	0	1
78		**Load register (literal)**						-	-	**0**	**1**	**1**	-	**0**	**0**	
79		LDR		w		W		0	0	0	1	1	0	0	0	
80		LDR	v	s		S		0	0	0	1	1	1	0	0	
81		LDR		x		X		0	1	0	1	1	0	0	0	
82		LDR	v	d		D		0	1	0	1	1	1	0	0	
83		LDRSW						1	0	0	1	1	0	0	0	
84		LDR	v	q		Q		1	0	0	1	1	1	0	0	
85		PRFM						1	1	0	1	1	0	0	0	
86		**Load/store no-allocate pair (o**						-	-	**1**	**0**	**1**	-	**0**	**0**	**0**
87		STNP		w		W		0	0	1	0	1	0	0	0	0
88		LDNP		w		W		0	0	1	0	1	0	0	0	0
89		STNP	v	s		S		0	0	1	0	1	1	0	0	0
90		LDNP	v	s		S		0	0	1	0	1	1	0	0	0
91		STNP	v	d		D		0	1	1	0	1	1	0	0	0
92		LDNP	v	d		D		0	1	1	0	1	1	0	0	0
93		STNP		x		X		1	0	1	0	1	0	0	0	0
94		LDNP		x		X		1	0	1	0	1	0	0	0	0
95		STNP	v	q		Q		1	0	1	0	1	1	0	0	0
96		LDNP	v	q		Q		1	0	1	0	1	1	0	0	0
97		**Load/store register pair (post**						**opc**		**1**	**0**	**1**	**V**	**0**	**0**	**1**
98		STP		postw	post	W		0	0	1	0	1	0	0	0	1
99		LDP		postw	post	W		0	0	1	0	1	0	0	0	1
100		STP	v	posts	post	S		0	0	1	0	1	1	0	0	1
101		LDP	v	posts	post	S		0	0	1	0	1	1	0	0	1
102		LDPSW		post	post			0	1	1	0	1	0	0	0	1
103		STP	v	postd	post	D		0	1	1	0	1	1	0	0	1
104		LDP	v	postd	post	D		0	1	1	0	1	1	0	0	1
105		STP		postx	post	X		1	0	1	0	1	0	0	0	1
106		LDP		postx	post	X		1	0	1	0	1	0	0	0	1
107		STP	v	postq	post	Q		1	0	1	0	1	1	0	0	1
108		LDP	v	postq	post	Q		1	0	1	0	1	1	0	0	1
109		**Load/store register pair (offse**						**opc**		**1**	**0**	**1**	**V**	**0**	**1**	**0**

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
110		STP		offw	off	W		0	0	1	0	1	0	0	1	0
111		LDP		offw	off	W		0	0	1	0	1	0	0	1	0
112		STP	v	offs	off	S		0	0	1	0	1	1	0	1	0
113		LDP	v	offs	off	S		0	0	1	0	1	1	0	1	0
114		LDPSW		off	off			0	1	1	0	1	0	0	1	0
115		STP	v	offd	off	D		0	1	1	0	1	1	0	1	0
116		LDP	v	offd	off	D		0	1	1	0	1	1	0	1	0
117		STP		offx	off	X		1	0	1	0	1	0	0	1	0
118		LDP		offx	off	X		1	0	1	0	1	0	0	1	0
119		STP	v	offq	off	Q		1	0	1	0	1	1	0	1	0
120		LDP	v	offq	off	Q		1	0	1	0	1	1	0	1	0
121		**Load/store register pair (pre-i**						**opc**		**1**	**0**	**1**	**V**	**0**	**1**	**1**
122		STP		prew	pre	W		0	0	1	0	1	0	0	1	1
123		LDP		prew	pre	W		0	0	1	0	1	0	0	1	1
124		STP	v	pres	pre	S		0	0	1	0	1	1	0	1	1
125		LDP	v	pres	pre	S		0	0	1	0	1	1	0	1	1
126		LDPSW		pre	pre			0	1	1	0	1	0	0	1	1
127		STP	v	pred	pre	D		0	1	1	0	1	1	0	1	1
128		LDP	v	pred	pre	D		0	1	1	0	1	1	0	1	1
129		STP		prex	pre	X		1	0	1	0	1	0	0	1	1
130		LDP		prex	pre	X		1	0	1	0	1	0	0	1	1
131		STP	v	preq	pre	Q		1	0	1	0	1	1	0	1	1
132		LDP	v	preq	pre	Q		1	0	1	0	1	1	0	1	1
133		**Load/store register (unscaled**						**size**		**1**	**1**	**1**	**V**	**0**	**0**	**op**
134		STURB						0	0	1	1	1	0	0	0	0
135		LDURB						0	0	1	1	1	0	0	0	0
136		LDURSB		x		X		0	0	1	1	1	0	0	0	1
137		LDURSB		w		W		0	0	1	1	1	0	0	0	1
138		STUR	v	b		B		0	0	1	1	1	1	0	0	0
139		LDUR	v	b		B		0	0	1	1	1	1	0	0	0
140		STUR	v	q		Q		0	0	1	1	1	1	0	0	1
141		LDUR	v	q		Q		0	0	1	1	1	1	0	0	1
142		STURH						0	1	1	1	1	0	0	0	0
143		LDURH						0	1	1	1	1	0	0	0	0
144		LDURSH		x		X		0	1	1	1	1	0	0	0	1
145		LDURSH		w		W		0	1	1	1	1	0	0	0	1
146		STUR	v	h		H		0	1	1	1	1	1	0	0	0
147		LDUR	v	h		H		0	1	1	1	1	1	0	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
148		STUR		w		W		1	0	1	1	1	0	0	0	0
149		LDUR		w		W		1	0	1	1	1	0	0	0	0
150		LDURSW						1	0	1	1	1	0	0	0	1
151		STUR	v	s		S		1	0	1	1	1	1	0	0	0
152		LDUR	v	s		S		1	0	1	1	1	1	0	0	0
153		STUR		x		X		1	1	1	1	1	0	0	0	0
154		LDUR		x		X		1	1	1	1	1	0	0	0	0
155		PRFUM						1	1	1	1	1	0	0	0	1
156		STUR	v	d		D		1	1	1	1	1	1	0	0	0
157		LDUR	v	d		D		1	1	1	1	1	1	0	0	0
158		**Load/store register (immediat**						**size**		**1**	**1**	**1**	**V**	**0**	**0**	**op**
159		STRB		post	post			0	0	1	1	1	0	0	0	0
160		LDRB		post	post			0	0	1	1	1	0	0	0	0
161		LDRSB		postx	post	X		0	0	1	1	1	0	0	0	1
162		LDRSB		postw	post	W		0	0	1	1	1	0	0	0	1
163		STR	v	postb	post	B		0	0	1	1	1	1	0	0	0
164		LDR	v	postb	post	B		0	0	1	1	1	1	0	0	0
165		STR	v	postq	post	Q		0	0	1	1	1	1	0	0	1
166		LDR	v	postq	post	Q		0	0	1	1	1	1	0	0	1
167		STRH		post	post			0	1	1	1	1	0	0	0	0
168		LDRH		post	post			0	1	1	1	1	0	0	0	0
169		LDRSH		postx	post	X		0	1	1	1	1	0	0	0	1
170		LDRSH		postw	post	W		0	1	1	1	1	0	0	0	1
171		STR	v	posth	post	H		0	1	1	1	1	1	0	0	0
172		LDR	v	posth	post	H		0	1	1	1	1	1	0	0	0
173		STR		postw	post	W		1	0	1	1	1	0	0	0	0
174		LDR		postw	post	W		1	0	1	1	1	0	0	0	0
175		LDRSW		post	post			1	0	1	1	1	0	0	0	1
176		STR	v	posts	post	S		1	0	1	1	1	1	0	0	0
177		LDR	v	posts	post	S		1	0	1	1	1	1	0	0	0
178		STR		postx	post	X		1	1	1	1	1	0	0	0	0
179		LDR		postx	post	X		1	1	1	1	1	0	0	0	0
180		STR	v	postd	post	D		1	1	1	1	1	1	0	0	0
181		LDR	v	postd	post	D		1	1	1	1	1	1	0	0	0
182		**Load/store register (unprivile**						**size**		**1**	**1**	**1**	**V**	**0**	**0**	**op**
183		STTRB						0	0	1	1	1	0	0	0	0
184		LDTRB						0	0	1	1	1	0	0	0	0
185		LDTRSB		x		X		0	0	1	1	1	0	0	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
186		LDTRSB		w		W		0	0	1	1	1	0	0	0	1
187		STTRH						0	1	1	1	1	0	0	0	0
188		LDTRH						0	1	1	1	1	0	0	0	0
189		LDTRSH		x		X		0	1	1	1	1	0	0	0	1
190		LDTRSH		w		W		0	1	1	1	1	0	0	0	1
191		STTR		w		W		1	0	1	1	1	0	0	0	0
192		LDTR		w		W		1	0	1	1	1	0	0	0	0
193		LDTRSW						1	0	1	1	1	0	0	0	1
194		STTR		x		X		1	1	1	1	1	0	0	0	0
195		LDTR		x		X		1	1	1	1	1	0	0	0	0
196		**Load/store register (immediat**						**size**		**1**	**1**	**1**	**V**	**0**	**0**	**op**
197		STRB		pre	pre			0	0	1	1	1	0	0	0	0
198		LDRB		pre	pre			0	0	1	1	1	0	0	0	0
199		LDRSB		prex	pre	X		0	0	1	1	1	0	0	0	1
200		LDRSB		prew	pre	W		0	0	1	1	1	0	0	0	1
201		STR	v	preb	pre	B		0	0	1	1	1	1	0	0	0
202		LDR	v	preb	pre	B		0	0	1	1	1	1	0	0	0
203		STR	v	preq	pre	Q		0	0	1	1	1	1	0	0	1
204		LDR	v	preq	pre	Q		0	0	1	1	1	1	0	0	1
205		STRH		pre	pre			0	1	1	1	1	0	0	0	0
206		LDRH		pre	pre			0	1	1	1	1	0	0	0	0
207		LDRSH		prex	pre	X		0	1	1	1	1	0	0	0	1
208		LDRSH		prew	pre	W		0	1	1	1	1	0	0	0	1
209		STR	v	preh	pre	H		0	1	1	1	1	1	0	0	0
210		LDR	v	preh	pre	H		0	1	1	1	1	1	0	0	0
211		STR		prew	pre	W		1	0	1	1	1	0	0	0	0
212		LDR		prew	pre	W		1	0	1	1	1	0	0	0	0
213		LDRSW		pre	pre			1	0	1	1	1	0	0	0	1
214		STR	v	pres	pre	S		1	0	1	1	1	1	0	0	0
215		LDR	v	pres	pre	S		1	0	1	1	1	1	0	0	0
216		STR		prex	pre	X		1	1	1	1	1	0	0	0	0
217		LDR		prex	pre	X		1	1	1	1	1	0	0	0	0
218		STR	v	pred	pre	D		1	1	1	1	1	1	0	0	0
219		LDR	v	pred	pre	D		1	1	1	1	1	1	0	0	0
220		**Load/store register (register c**						**size**		**1**	**1**	**1**	**v**	**0**	**0**	**op**
221		STRB		off	off			0	0	1	1	1	0	0	0	0
222		LDRB		off	off			0	0	1	1	1	0	0	0	0
223		LDRSB		offx	off	X		0	0	1	1	1	0	0	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
224		LDRSB		offw	off	W		0	0	1	1	1	0	0	0	1
225		STR	v	offb	off	B		0	0	1	1	1	1	0	0	0
226		LDR	v	offb	off	B		0	0	1	1	1	1	0	0	0
227		STR	v	offq	off	Q		0	0	1	1	1	1	0	0	1
228		LDR	v	offq	off	Q		0	0	1	1	1	1	0	0	1
229		STRH		off	off			0	1	1	1	1	0	0	0	0
230		LDRH		off	off			0	1	1	1	1	0	0	0	0
231		LDRSH		offx	off	X		0	1	1	1	1	0	0	0	1
232		LDRSH		offw	off	W		0	1	1	1	1	0	0	0	1
233		STR	v	offh	off	H		0	1	1	1	1	1	0	0	0
234		LDR	v	offh	off	H		0	1	1	1	1	1	0	0	0
235		STR		offw	off	W		1	0	1	1	1	0	0	0	0
236		LDR		offw	off	W		1	0	1	1	1	0	0	0	0
237		LDRSW		off	off			1	0	1	1	1	0	0	0	1
238		STR	v	offs	off	S		1	0	1	1	1	1	0	0	0
239		LDR	v	offs	off	S		1	0	1	1	1	1	0	0	0
240		STR		offx	off	X		1	1	1	1	1	0	0	0	0
241		LDR		offx	off	X		1	1	1	1	1	0	0	0	0
243		STR	v	offd	off	D		1	1	1	1	1	1	0	0	0
244		LDR	v	offd	off	D		1	1	1	1	1	1	0	0	0
242		PRFM		off	off			1	1	1	1	1	0	0	0	1
245		**Load/store register (unsigned**						**size**		**1**	**1**	**1**	**v**	**0**	**1**	**op**
246		STRB		imm	imm			0	0	1	1	1	0	0	1	0
247		LDRB		imm	imm			0	0	1	1	1	0	0	1	0
248		LDRSB		immx	imm	X		0	0	1	1	1	0	0	1	1
249		LDRSB		immw	imm	W		0	0	1	1	1	0	0	1	1
250		STR	v	immb	imm	B		0	0	1	1	1	1	0	1	0
251		LDR	v	immb	imm	B		0	0	1	1	1	1	0	1	0
252		STR	v	immq	imm	Q		0	0	1	1	1	1	0	1	1
253		LDR	v	immq	imm	Q		0	0	1	1	1	1	0	1	1
254		STRH		imm	imm			0	1	1	1	1	0	0	1	0
255		LDRH		imm	imm			0	1	1	1	1	0	0	1	0
256		LDRSH		immx	imm	X		0	1	1	1	1	0	0	1	1
257		LDRSH		immw	imm	W		0	1	1	1	1	0	0	1	1
258		STR	v	immh	imm	H		0	1	1	1	1	1	0	1	0
259		LDR	v	immh	imm	H		0	1	1	1	1	1	0	1	0
260		STR		immw	imm	W		1	0	1	1	1	0	0	1	0
261		LDR		immw	imm	W		1	0	1	1	1	0	0	1	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
262		LDRSW		imm	imm			1	0	1	1	1	0	0	1	1
263		STR	v	imms	imm	S		1	0	1	1	1	1	0	1	0
264		LDR	v	imms	imm	S		1	0	1	1	1	1	0	1	0
265		STR		immx	imm	X		1	1	1	1	1	0	0	1	0
266		LDR		immx	imm	X		1	1	1	1	1	0	0	1	0
268		STR	v	immd	imm	D		1	1	1	1	1	1	0	1	0
269		LDR	v	immd	imm	D		1	1	1	1	1	1	0	1	0
267		PRFM		imm	imm			1	1	1	1	1	0	0	1	1
270		**Data processing – Immedia**									**1**	**0**	**0**			
271		**PC-rel. addressing**						**op**	**immlo**		**1**	**0**	**0**	**0**	**0**	
272		ADR						0	immlo		1	0	0	0	0	
273		ADRP						1	immlo		1	0	0	0	0	
274		**Add/subtract (immediate)**						**sf**	**op**	**S**	**1**	**0**	**0**	**0**	**1**	**sh**
275		ADD		immw	imm	W		0	0	0	1	0	0	0	1	-
276		ADDS		immw	imm	W		0	0	1	1	0	0	0	1	-
277		SUB		immw	imm	W		0	1	0	1	0	0	0	1	-
278		SUBS		immw	imm	W		0	1	1	1	0	0	0	1	-
279		ADD		immx	imm	X		1	0	0	1	0	0	0	1	-
280		ADDS		immx	imm	X		1	0	1	1	0	0	0	1	-
281		SUB		immx	imm	X		1	1	0	1	0	0	0	1	-
282		SUBS		immx	imm	X		1	1	1	1	0	0	0	1	-
283		**Logical (immediate)**		imm	imm			**sf**	**opc**		**1**	**0**	**0**	**1**	**0**	**0**
284		AND		immw	imm	W		0	0	0	1	0	0	1	0	0
285		ORR		immw	imm	W		0	0	1	1	0	0	1	0	0
286		EOR		immw	imm	W		0	1	0	1	0	0	1	0	0
287		ANDS		immw	imm	W		0	1	1	1	0	0	1	0	0
288		AND		immx	imm	X		1	0	0	1	0	0	1	0	0
289		ORR		immx	imm	X		1	0	1	1	0	0	1	0	0
290		EOR		immx	imm	X		1	1	0	1	0	0	1	0	0
291		ANDS		immx	imm	X		1	1	1	1	0	0	1	0	0
292		**Move wide (immediate)**						**sf**	**opc**		**1**	**0**	**0**	**1**	**0**	**1**
293		MOVN		w		W		0	0	0	1	0	0	1	0	1
294		MOVZ		w		W		0	1	0	1	0	0	1	0	1
295		MOVK		w		W		0	1	1	1	0	0	1	0	1
296		MOVN		x		X		1	0	0	1	0	0	1	0	1
297		MOVZ		x		X		1	1	0	1	0	0	1	0	1
298		MOVK		x		X		1	1	1	1	0	0	1	0	1
299		**Bitfield**						**sf**	**opc**		**1**	**0**	**0**	**1**	**1**	**0**

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
300		SBFM		w		W		0	0	0	1	0	0	1	1	0
301		BFM		w		W		0	0	1	1	0	0	1	1	0
302		UBFM		w		W		0	1	0	1	0	0	1	1	0
303		SBFM		x		X		1	0	0	1	0	0	1	1	0
304		BFM		x		X		1	0	1	1	0	0	1	1	0
305		UBFM		x		X		1	1	0	1	0	0	1	1	0
306		**Extract**						**sf**		**op21**	**1**	**0**	**0**	**1**	**1**	**1**
307		EXTR		w		W		0	0	0	1	0	0	1	1	1
308		EXTR		x		X		1	0	0	1	0	0	1	1	1
309		**Data Processing – register**										**1**	**0**	**1**		
310		**Logical (shifted register)**						**sf**		**opc**	**0**	**1**	**0**	**1**	**0**	**sh**
311		AND		w		W		0	0	0	0	1	0	1	0	sh
312		BIC		w		W		0	0	0	0	1	0	1	0	sh
313		ORR		w		W		0	0	1	0	1	0	1	0	sh
314		ORN		w		W		0	0	1	0	1	0	1	0	sh
315		EOR		w		W		0	1	0	0	1	0	1	0	sh
316		EON		w		W		0	1	0	0	1	0	1	0	sh
317		ANDS		w		W		0	1	1	0	1	0	1	0	sh
318		BICS		w		W		0	1	1	0	1	0	1	0	sh
319		AND		x		X		1	0	0	0	1	0	1	0	sh
320		BIC		x		X		1	0	0	0	1	0	1	0	sh
321		ORR		x		X		1	0	1	0	1	0	1	0	sh
322		ORN		x		X		1	0	1	0	1	0	1	0	sh
323		EOR		x		X		1	1	0	0	1	0	1	0	sh
324		EON		x		X		1	1	0	0	1	0	1	0	sh
325		ANDS		x		X		1	1	1	0	1	0	1	0	sh
326		BICS		x		X		1	1	1	0	1	0	1	0	sh
327		**Add/subtract (shifted register**						**sf**	**op**	**S**	**0**	**1**	**0**	**1**	**1**	**sh**
328		ADD		w		W		0	0	0	0	1	0	1	1	-
329		ADDS		w		W		0	0	1	0	1	0	1	1	-
330		SUB		w		W		0	1	0	0	1	0	1	1	-
331		SUBS		w		W		0	1	1	0	1	0	1	1	-
332		ADD		x		X		1	0	0	0	1	0	1	1	-
333		ADDS		x		X		1	0	1	0	1	0	1	1	-
334		SUB		x		X		1	1	0	0	1	0	1	1	-
335		SUBS		x		X		1	1	1	0	1	0	1	1	-
336		**Add/subtract (extended regist**						**sf**	**op**	**S**	**0**	**1**	**0**	**1**	**1**	**op**
337		ADD		extw	ext	W		0	0	0	0	1	0	1	1	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
338		ADDS		extw	ext	W		0	0	1	0	1	0	1	1	0
339		SUB		extw	ext	W		0	1	0	0	1	0	1	1	0
340		SUBS		extw	ext	W		0	1	1	0	1	0	1	1	0
341		ADD		extx	ext	X		1	0	0	0	1	0	1	1	0
342		ADDS		extx	ext	X		1	0	1	0	1	0	1	1	0
343		SUB		extx	ext	X		1	1	0	0	1	0	1	1	0
344		SUBS		extx	ext	X		1	1	1	0	1	0	1	1	0
345		**Add/subtract (with carry)**						**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**
346		ADC		w		W		0	0	0	1	1	0	1	0	0
347		ADCS		w		W		0	0	1	1	1	0	1	0	0
348		SBC		w		W		0	1	0	1	1	0	1	0	0
349		SBCS		w		W		0	1	1	1	1	0	1	0	0
350		ADC		x		X		1	0	0	1	1	0	1	0	0
351		ADCS		x		X		1	0	1	1	1	0	1	0	0
352		SBC		x		X		1	1	0	1	1	0	1	0	0
353		SBCS		x		X		1	1	1	1	1	0	1	0	0
354		**Conditional compare (registe**						**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**
355		CCMN		w		W		0	0	1	1	1	0	1	0	0
356		CCMN		x		X		1	0	1	1	1	0	1	0	0
357		CCMP		w		W		0	1	1	1	1	0	1	0	0
358		CCMP		x		X		1	1	1	1	1	0	1	0	0
359		**Conditional compare (immedi**						**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**
360		CCMN		immw	imm	W		0	0	1	1	1	0	1	0	0
361		CCMN		immx	imm	X		1	0	1	1	1	0	1	0	0
362		CCMP		immw	imm	W		0	1	1	1	1	0	1	0	0
363		CCMP		immx	imm	X		1	1	1	1	1	0	1	0	0
364		**Conditional select**						**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**1**
365		CSEL		w		W		0	0	0	1	1	0	1	0	1
366		CSINC		w		W		0	0	0	1	1	0	1	0	1
367		CSINV		w		W		0	1	0	1	1	0	1	0	1
368		CSNEG		w		W		0	1	0	1	1	0	1	0	1
369		CSEL		x		X		1	0	0	1	1	0	1	0	1
370		CSINC		x		X		1	0	0	1	1	0	1	0	1
371		CSINV		x		X		1	1	0	1	1	0	1	0	1
372		CSNEG		x		X		1	1	0	1	1	0	1	0	1
373		**Data-processing (3 source)**						**sf**	**op54**		**1**	**1**	**0**	**1**	**1**	**c**
374		MADD		w		W		0	0	0	1	1	0	1	1	0
375		MADD		x		X		1	0	0	1	1	0	1	1	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
376		SMADDL						1	0	0	1	1	0	1	1	0
377		UMADDL						1	0	0	1	1	0	1	1	1
378		MSUB		w		W		0	0	0	1	1	0	1	1	0
379		MSUB		x		X		1	0	0	1	1	0	1	1	0
380		SMSUBL						1	0	0	1	1	0	1	1	0
381		UMSUBL						1	0	0	1	1	0	1	1	1
382		SMULH						1	0	0	1	1	0	1	1	0
383		UMULH						1	0	0	1	1	0	1	1	1
384		**Data-processing (2 source)**						**sf**	**0**	**S**	**1**	**1**	**0**	**1**	**0**	**1**
385		CRC32X						1	0	0	1	1	0	1	0	1
386		CRC32CX						1	0	0	1	1	0	1	0	1
387		CRC32B						0	0	0	1	1	0	1	0	1
388		CRC32CB						0	0	0	1	1	0	1	0	1
389		CRC32H						0	0	0	1	1	0	1	0	1
390		CRC32CH						0	0	0	1	1	0	1	0	1
391		CRC32W						0	0	0	1	1	0	1	0	1
392		CRC32CW						0	0	0	1	1	0	1	0	1
393		UDIV		w		W		0	0	0	1	1	0	1	0	1
394		UDIV		x		X		1	0	0	1	1	0	1	0	1
395		SDIV		w		W		0	0	0	1	1	0	1	0	1
396		SDIV		x		X		1	0	0	1	1	0	1	0	1
397		LSLV		w		W		0	0	0	1	1	0	1	0	1
398		LSLV		x		X		1	0	0	1	1	0	1	0	1
399		LSRV		w		W		0	0	0	1	1	0	1	0	1
400		LSRV		x		X		1	0	0	1	1	0	1	0	1
401		ASRV		w		W		0	0	0	1	1	0	1	0	1
402		ASRV		x		X		1	0	0	1	1	0	1	0	1
403		RORV		w		W		0	0	0	1	1	0	1	0	1
404		RORV		x		X		1	0	0	1	1	0	1	0	1
405		**Data-processing (1 source)**						**sf**	**1**	**S**	**1**	**1**	**0**	**1**	**0**	**1**
406		RBIT		w		W		0	1	0	1	1	0	1	0	1
407		RBIT		x		X		1	1	0	1	1	0	1	0	1
408		CLZ		w		W		0	1	0	1	1	0	1	0	1
409		CLZ		x		X		1	1	0	1	1	0	1	0	1
410		CLS		w		W		0	1	0	1	1	0	1	0	1
411		CLS		x		X		1	1	0	1	1	0	1	0	1
412		REV		w		W		0	1	0	1	1	0	1	0	1
413		REV		x		X		1	1	0	1	1	0	1	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
414		REV16		w		W		1	1	0	1	1	0	1	0	1
415		REV16		x		X		0	1	0	1	1	0	1	0	1
416		REV32						1	1	0	1	1	0	1	0	1
417	//	**Data Processing – SIMD an**										**1**	**1**	**1**		
418	//	**Floating-point<->fixed-point c**						**sf**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
419	//	SCVTF	v	scalar_fixed_	scalar_fixed	32_bit_to_sin		0	0	0	1	1	1	1	0	0
420	//	UCVTF	v	scalar_fixed_	scalar_fixed	32_bit_to_sin		0	0	0	1	1	1	1	0	0
421	//	FCVTZS	v	scalar_fixed_	scalar_fixed	Single_precisi		0	0	0	1	1	1	1	0	1
422	//	FCVTZU	v	scalar_fixed_	scalar_fixed	Single_precisi		0	0	0	1	1	1	1	0	1
423	//	SCVTF	v	scalar_fixed_	scalar_fixed	32_bit_to_dou		0	0	0	1	1	1	1	0	0
424	//	UCVTF	v	scalar_fixed_	scalar_fixed	32_bit_to_dou		0	0	0	1	1	1	1	0	0
425	//	FCVTZS	v	scalar_fixed_	scalar_fixed	Double_precis		0	0	0	1	1	1	1	0	1
426	//	FCVTZU	v	scalar_fixed_	scalar_fixed	Double_precis		0	0	0	1	1	1	1	0	1
427	//	SCVTF	v	scalar_fixed_	scalar_fixed	64_bit_to_sin		1	0	0	1	1	1	1	0	0
428	//	UCVTF	v	scalar_fixed_	scalar_fixed	64_bit_to_sin		1	0	0	1	1	1	1	0	0
429	//	FCVTZS	v	scalar_fixed_	scalar_fixed	Single_precisi		1	0	0	1	1	1	1	0	1
430	//	FCVTZU	v	scalar_fixed_	scalar_fixed	Single_precisi		1	0	0	1	1	1	1	0	1
431	//	SCVTF	v	scalar_fixed_	scalar_fixed	64_bit_to_dou		1	0	0	1	1	1	1	0	0
432	//	UCVTF	v	scalar_fixed_	scalar_fixed	64_bit_to_dou		1	0	0	1	1	1	1	0	0
433	//	FCVTZS	v	scalar_fixed_	scalar_fixed	Double_precis		1	0	0	1	1	1	1	0	1
434	//	FCVTZU	v	scalar_fixed_	scalar_fixed	Double_precis		1	0	0	1	1	1	1	0	1
435	//	**Floating-point conditional cor**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
436	//	FCCMP	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
437	//	FCCMPE	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
438	//	FCCMP	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
439	//	FCCMPE	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
440	//	**Floating-point data-processin**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
441	//	FMUL	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
442	//	FDIV	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
443	//	FADD	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
444	//	FSUB	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
445	//	FMAX	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
446	//	FMIN	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
447	//	FMAXNM	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
448	//	FMINNM	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
449	//	FNMUL	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
450	//	FMUL	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
451	//	FDIV	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
452	//	FADD	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
453	//	FSUB	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
454	//	FMAX	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
455	//	FMIN	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
456	//	FMAXNM	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
457	//	FMINNM	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
458	//	FNMUL	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
459	//	**Floating-point conditional sel**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
460	//	FCSEL	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
461	//	FCSEL	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
462	//	**Floating-point immediate**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
463	//	FMOV	v	scalar_immed	scalar_imme	Single_precisi		0	0	0	1	1	1	1	0	0
464	//	FMOV	v	scalar_immed	scalar_imme	Double_precis		0	0	0	1	1	1	1	0	0
465	//	**Floating-point compare**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
466	//	FCMP	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
467	//	FCMP	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
468	//	FCMPE	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
469	//	FCMPE	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
470	//	FCMP	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
471	//	FCMP	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
472	//	FCMPE	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
473	//	FCMPE	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
474	//	**Floating-point data-processin**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
475	//	FMOV	v	register_Sing	register	Single_precisi		0	0	0	1	1	1	1	0	0
476	//	FABS	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
477	//	FNEG	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
478	//	FSQRT	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
479	//	FCVT	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
480	//	FCVT	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	0	0
481	//	FRINTN	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
482	//	FRINTP	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
483	//	FRINTM	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
484	//	FRINTZ	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
485	//	FRINTA	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
486	//	FRINTX	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
487	//	FRINTI	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
488	//	FMOV	v	register_Dout	register	Double_precis		0	0	0	1	1	1	1	0	0
489	//	FABS	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
490	//	FNEG	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
491	//	FSQRT	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
492	//	FCVT	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
493	//	FCVT	v	Double_preci		Double_precis		0	0	0	1	1	1	1	0	0
494	//	FRINTN	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
495	//	FRINTP	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
496	//	FRINTM	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
497	//	FRINTZ	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
498	//	FRINTA	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
499	//	FRINTX	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
500	//	FRINTI	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
501	//	FCVT	v	Half_precisio		Half_precisior		0	0	0	1	1	1	1	0	1
502	//	FCVT	v	Half_precisio		Half_precisior		0	0	0	1	1	1	1	0	1
503	//	**Floating-point<->integer conv**						**sf**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**ty**
504	//	FCVTNS	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
505	//	FCVTNU	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
506	//	SCVTF	v	scalar_intege	scalar_integ	32_bit_to_sing		0	0	0	1	1	1	1	0	0
507	//	UCVTF	v	scalar_intege	scalar_integ	32_bit_to_sing		0	0	0	1	1	1	1	0	0
508	//	FCVTAS	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
509	//	FCVTAU	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
510	//	FMOV	v	general_Sing	general	Single_precisi		0	0	0	1	1	1	1	0	0
511	//	FMOV	v	general_32_b	general	32_bit_to_sing		0	0	0	1	1	1	1	0	0
512	//	FCVTPS	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
513	//	FCVTPU	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
514	//	FCVTMS	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0
515	//	FCVTMU	v	scalar_Single	scalar	Single_precisi		0	0	0	1	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
516	//	FCVTZS	v	scalar_intege	scalar_integ	Single_precisi		0	0	0	1	1	1	1	0	0
517	//	FCVTZU	v	scalar_intege	scalar_integ	Single_precisi		0	0	0	1	1	1	1	0	0
518	//	FCVTNS	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
519	//	FCVTNU	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
520	//	SCVTF	v	scalar_intege	scalar_integ	32_bit_to_dou		0	0	0	1	1	1	1	0	0
521	//	UCVTF	v	scalar_intege	scalar_integ	32_bit_to_dou		0	0	0	1	1	1	1	0	0
522	//	FCVTAS	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
523	//	FCVTAU	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
524	//	FCVTPS	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
525	//	FCVTPU	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
526	//	FCVTMS	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
527	//	FCVTMU	v	scalar_Doubl	scalar	Double_precis		0	0	0	1	1	1	1	0	0
528	//	FCVTZS	v	scalar_intege	scalar_integ	Double_precis		0	0	0	1	1	1	1	0	0
529	//	FCVTZU	v	scalar_intege	scalar_integ	Double_precis		0	0	0	1	1	1	1	0	0
530	//	FCVTNS	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
531	//	FCVTNU	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
532	//	SCVTF	v	scalar_intege	scalar_integ	64_bit_to_sing		1	0	0	1	1	1	1	0	0
533	//	UCVTF	v	scalar_intege	scalar_integ	64_bit_to_sing		1	0	0	1	1	1	1	0	0
534	//	FCVTAS	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
535	//	FCVTAU	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
536	//	FCVTPS	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
537	//	FCVTPU	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
538	//	FCVTMS	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
539	//	FCVTMU	v	scalar_Single	scalar	Single_precisi		1	0	0	1	1	1	1	0	0
540	//	FCVTZS	v	scalar_intege	scalar_integ	Single_precisi		1	0	0	1	1	1	1	0	0
541	//	FCVTZU	v	scalar_intege	scalar_integ	Single_precisi		1	0	0	1	1	1	1	0	0
542	//	FCVTNS	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
543	//	FCVTNU	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
544	//	SCVTF	v	scalar_intege	scalar_integ	64_bit_to_dou		1	0	0	1	1	1	1	0	0
545	//	UCVTF	v	scalar_intege	scalar_integ	64_bit_to_dou		1	0	0	1	1	1	1	0	0
546	//	FCVTAS	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
547	//	FCVTAU	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
548	//	FMOV	v	general_Doul	general	Double_precis		1	0	0	1	1	1	1	0	0
549	//	FMOV	v	general_64_b	general	64_bit_to_dou		1	0	0	1	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
550	//	FCVTPS	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
551	//	FCVTPU	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
552	//	FCVTMS	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
553	//	FCVTMU	v	scalar_Doubl	scalar	Double_precis		1	0	0	1	1	1	1	0	0
554	//	FCVTZS	v	scalar_intege	scalar_integ	Double_precis		1	0	0	1	1	1	1	0	0
555	//	FCVTZU	v	scalar_intege	scalar_integ	Double_precis		1	0	0	1	1	1	1	0	0
556	//	FMOV	v	general_Top_	general	Top_half_of_		1	0	0	1	1	1	1	0	1
557	//	FMOV	v	general_64_b	general	64_bit_to_top		1	0	0	1	1	1	1	0	1
558	//	**Floating-point data-processin**						**M**	**0**	**S**	**1**	**1**	**1**	**1**	**1**	**ty**
559	//	FMADD	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	1	0
560	//	FMSUB	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	1	0
561	//	FNMADD	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	1	0
562	//	FNMSUB	v	Single_precis		Single_precisi		0	0	0	1	1	1	1	1	0
563	//	FMADD	v	Double_preci		Double_precis		0	0	0	1	1	1	1	1	0
564	//	FMSUB	v	Double_preci		Double_precis		0	0	0	1	1	1	1	1	0
565	//	FNMADD	v	Double_preci		Double_precis		0	0	0	1	1	1	1	1	0
566	//	FNMSUB	v	Double_preci		Double_precis		0	0	0	1	1	1	1	1	0
567	//	**AdvSIMD scalar three same**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**si**
568	//	SQADD	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
569	//	SQSUB	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
570	//	CMGT	v	register_Scal	register	Scalar		0	1	0	1	1	1	1	0	-
571	//	CMGE	v	register_Scal	register	Scalar		0	1	0	1	1	1	1	0	-
572	//	SSHL	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
573	//	SQSHL	v	register_Scal	register	Scalar		0	1	0	1	1	1	1	0	-
574	//	SRSHL	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
575	//	SQRSHL	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
576	//	ADD	v	vector_Scala	vector	Scalar		0	1	0	1	1	1	1	0	-
577	//	CMTST	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
578	//	SQDMULH	v	vector_Scala	vector	Scalar		0	1	0	1	1	1	1	0	-
579	//	FMULX	v	Scalar		Scalar		0	1	0	1	1	1	1	0	0
580	//	FCMEQ	v	register_Scal	register	Scalar		0	1	0	1	1	1	1	0	0
581	//	FRECPS	v	Scalar		Scalar		0	1	0	1	1	1	1	0	0
582	//	FRSQRTS	v	Scalar		Scalar		0	1	0	1	1	1	1	0	1
583	//	UQADD	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
584	//	UQSUB	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
585	//	CMHI	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	-
586	//	CMHS	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	-
587	//	USHL	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
588	//	UQSHL	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	-
589	//	URSHL	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
590	//	UQRSHL	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
591	//	SUB	v	vector_Scala	vector	Scalar		0	1	1	1	1	1	1	0	-
592	//	CMEQ	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	-
593	//	SQRDMULH	v	vector_Scala	vector	Scalar		0	1	1	1	1	1	1	0	-
594	//	FCMGE	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	0
595	//	FACGE	v	Scalar		Scalar		0	1	1	1	1	1	1	0	0
596	//	FABD	v	Scalar		Scalar		0	1	1	1	1	1	1	0	1
597	//	FCMGT	v	register_Scal	register	Scalar		0	1	1	1	1	1	1	0	1
598	//	FACGT	v	Scalar		Scalar		0	1	1	1	1	1	1	0	1
599	//	**AdvSIMD scalar three differer**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**si**
600	//	SQDMLAL	v	vector_Scala	vector	Scalar	writes to low	0	1	0	1	1	1	1	0	siz
601	//	SQDMLAL2	v	vector_Scala	vector	Scalar	writes to hig	0	1	0	1	1	1	1	0	siz
602	//	SQDMLSL	v	vector_Scala	vector	Scalar	writes to low	0	1	0	1	1	1	1	0	siz
603	//	SQDMLSL2	v	vector_Scala	vector	Scalar	writes to hig	0	1	0	1	1	1	1	0	siz
604	//	SQDMULL	v	vector_Scala	vector	Scalar	writes to low	0	1	0	1	1	1	1	0	siz
605	//	SQDMULL2	v	vector_Scala	vector	Scalar	writes to hig	0	1	0	1	1	1	1	0	siz
606	//	**AdvSIMD scalar two-reg misc**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**si**
607	//	SUQADD	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
608	//	SQABS	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
609	//	CMGT	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	-
610	//	CMEQ	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	-
611	//	CMLT	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	-
612	//	ABS	v	Scalar		Scalar		0	1	0	1	1	1	1	0	-
613	//	SQXTN	v	Scalar		Scalar	writes to low	0	1	0	1	1	1	1	0	-
614	//	SQXTN2	v	Scalar		Scalar	writes to hig	0	1	0	1	1	1	1	0	-
615	//	FCVTNS	v	vector_Scala	vector	Scalar		0	1	0	1	1	1	1	0	0
616	//	FCVTMS	v	vector_Scala	vector	Scalar		0	1	0	1	1	1	1	0	0
617	//	FCVTAS	v	vector_Scala	vector	Scalar		0	1	0	1	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
618	//	SCVTF	v	vector_intege	vector_integ	Scalar		0	1	0	1	1	1	1	0	0
619	//	FCMGT	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	1
620	//	FCMEQ	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	1
621	//	FCMLT	v	zero_Scalar	zero	Scalar		0	1	0	1	1	1	1	0	1
622	//	FCVTPS	v	vector_Scalai	vector	Scalar		0	1	0	1	1	1	1	0	1
623	//	FCVTZS	v	vector_intege	vector_integ	Scalar		0	1	0	1	1	1	1	0	1
624	//	FRECPE	v	Scalar		Scalar		0	1	0	1	1	1	1	0	1
625	//	FRECPX						0	1	0	1	1	1	1	0	1
626	//	USQADD	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
627	//	SQNEG	v	Scalar		Scalar		0	1	1	1	1	1	1	0	-
628	//	CMGE	v	zero_Scalar	zero	Scalar		0	1	1	1	1	1	1	0	-
629	//	CMLE	v	zero_Scalar	zero	Scalar		0	1	1	1	1	1	1	0	-
630	//	NEG	v	vector_Scalai	vector	Scalar		0	1	1	1	1	1	1	0	-
631	//	SQXTUN	v	Scalar		Scalar	writes to low	0	1	1	1	1	1	1	0	-
632	//	SQXTUN2	v	Scalar		Scalar	writes to hig	0	1	1	1	1	1	1	0	-
633	//	UQXTN	v	Scalar		Scalar	writes to low	0	1	1	1	1	1	1	0	-
634	//	UQXTN2	v	Scalar		Scalar	writes to hig	0	1	1	1	1	1	1	0	-
635	//	FCVTXN	v	Scalar		Scalar	writes to low	0	1	1	1	1	1	1	0	0
636	//	FCVTXN2	v	Scalar		Scalar	writes to hig	0	1	1	1	1	1	1	0	0
637	//	FCVTNU	v	vector_Scalai	vector	Scalar		0	1	1	1	1	1	1	0	0
638	//	FCVTMU	v	vector_Scalai	vector	Scalar		0	1	1	1	1	1	1	0	0
639	//	FCVTAU	v	vector_Scalai	vector	Scalar		0	1	1	1	1	1	1	0	0
640	//	UCVTF	v	vector_intege	vector_integ	Scalar		0	1	1	1	1	1	1	0	0
641	//	FCMGE	v	zero_Scalar	zero	Scalar		0	1	1	1	1	1	1	0	1
642	//	FCMLE	v	zero_Scalar	zero	Scalar		0	1	1	1	1	1	1	0	1
643	//	FCVTPU	v	vector_Scalai	vector	Scalar		0	1	1	1	1	1	1	0	1
644	//	FCVTZU	v	vector_intege	vector_integ	Scalar		0	1	1	1	1	1	1	0	1
645	//	FRSQRTE	v	Scalar		Scalar		0	1	1	1	1	1	1	0	1
646	//	**AdvSIMD scalar pairwise**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**si**
647	//	ADDP		scalar	scalar			0	1	0	1	1	1	1	0	-
648	//	FMAXNMP		scalar	scalar			0	1	1	1	1	1	1	0	0
649	//	FADDP		scalar	scalar			0	1	1	1	1	1	1	0	0
650	//	FMAXP		scalar	scalar			0	1	1	1	1	1	1	0	0
651	//	FMINNMP		scalar	scalar			0	1	1	1	1	1	1	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
652	//	FMINP		scalar	scalar			0	1	1	1	1	1	1	0	1
653	//	**AdvSIMD scalar copy**						**0**	**1**	**op**	**1**	**1**	**1**	**1**	**0**	**0**
654	//	DUP	v	element_Scal	element	Scalar		0	1	0	1	1	1	1	0	0
655	//	**AdvSIMD scalar x indexed ele**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**1**	**si**
656	//	SQDMLAL	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
657	//	SQDMLAL2	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
658	//	SQDMLSL	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
659	//	SQDMLSL2	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
660	//	SQDMULL	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
661	//	SQDMULL2	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
662	//	SQDMULH	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
663	//	SQRDMULH	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	-
664	//	FMLA	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	1
665	//	FMLS	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	1
666	//	FMUL	v	by_element_S	by_element	Scalar		0	1	0	1	1	1	1	1	1
667	//	FMULX	v	by_element_S	by_element	Scalar		0	1	1	1	1	1	1	1	1
668	//	**AdvSIMD scalar shift by imme**						**0**	**1**	**U**	**1**	**1**	**1**	**1**	**1**	**0**
669	//	SSHR	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
670	//	SSRA	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
671	//	SRSHR	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
672	//	SRSRA	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
673	//	SHL	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
674	//	SQSHL	v	immediate_S	immediate	Scalar	immh != 000	0	1	0	1	1	1	1	1	0
675	//	SQSHRN	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
676	//	SQSHRN2	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
677	//	SQRSHRN	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
678	//	SQRSHRN2	v	Scalar		Scalar	immh != 000	0	1	0	1	1	1	1	1	0
679	//	SCVTF	v	vector_fixed_	vector_fixed	Scalar	immh != 000	0	1	0	1	1	1	1	1	0
680	//	FCVTZS	v	vector_fixed_	vector_fixed	Scalar	immh != 000	0	1	0	1	1	1	1	1	0
681	//	USHR	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
682	//	USRA	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
683	//	URSHR	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
684	//	URSRA	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
685	//	SRI	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
686	//	SLI	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
687	//	SQSHLU	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
688	//	UQSHL	v	immediate_S	immediate	Scalar	immh != 000	0	1	1	1	1	1	1	1	0
689	//	SQSHRUN	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
690	//	SQSHRUN2	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
691	//	SQRSHRUN	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
692	//	SQRSHRUN2	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
693	//	UQSHRN	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
694	//	UQRSHRN	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
695	//	UQRSHRN2	v	Scalar		Scalar	immh != 000	0	1	1	1	1	1	1	1	0
696	//	UCVTF	v	vector_fixed_	vector_fixed	Scalar	immh != 000	0	1	1	1	1	1	1	1	0
697	//	FCVTZU	v	vector_fixed_	vector_fixed	Scalar	immh != 000	0	1	1	1	1	1	1	1	0
698	//	**Crypto three-reg SHA**						**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**si**
699	//	SHA1C						0	1	0	1	1	1	1	0	0
700	//	SHA1P						0	1	0	1	1	1	1	0	0
701	//	SHA1M						0	1	0	1	1	1	1	0	0
702	//	SHA1SU0						0	1	0	1	1	1	1	0	0
703	//	SHA256H						0	1	0	1	1	1	1	0	0
704	//	SHA256H2						0	1	0	1	1	1	1	0	0
705	//	SHA256SU1						0	1	0	1	1	1	1	0	0
706	//	**Crypto two-reg SHA**						**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**si**
707	//	SHA1H						0	1	0	1	1	1	1	0	0
708	//	SHA1SU1						0	1	0	1	1	1	1	0	0
709	//	SHA256SU0						0	1	0	1	1	1	1	0	0
710	//	**Crypto AES**						**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**si**
711	//	AESE						0	1	0	0	1	1	1	0	0
712	//	AESD						0	1	0	0	1	1	1	0	0
713	//	AESMC						0	1	0	0	1	1	1	0	0
714	//	AESIMC						0	1	0	0	1	1	1	0	0
715	//	**AdvSIMD three same**						**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**si**
716	//	SHADD						0	Q	0	0	1	1	1	0	-
717	//	SQADD	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
718	//	SRHADD						0	Q	0	0	1	1	1	0	-
719	//	SHSUB						0	Q	0	0	1	1	1	0	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
720	//	SQSUB	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
721	//	CMGT	v	register_Vect	register	Vector		0	Q	0	0	1	1	1	0	-
722	//	CMGE	v	register_Vect	register	Vector		0	Q	0	0	1	1	1	0	-
723	//	SSHL Vector						0	Q	0	0	1	1	1	0	-
724	//	SQSHL	v	register_Vect	register	Vector		0	Q	0	0	1	1	1	0	-
725	//	SRSHL	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
726	//	SQRSHL	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
727	//	SMAX						0	Q	0	0	1	1	1	0	-
728	//	SMIN						0	Q	0	0	1	1	1	0	-
729	//	SABD						0	Q	0	0	1	1	1	0	-
730	//	SABA						0	Q	0	0	1	1	1	0	-
731	//	ADD	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	-
732	//	CMTST	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
733	//	MLA		vector	vector			0	Q	0	0	1	1	1	0	-
734	//	MUL		vector	vector			0	Q	0	0	1	1	1	0	-
735	//	SMAXP						0	Q	0	0	1	1	1	0	-
736	//	SMINP						0	Q	0	0	1	1	1	0	-
737	//	SQDMULH	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	-
738	//	ADDP		vector	vector			0	Q	0	0	1	1	1	0	-
739	//	FMAXNM		vector	vector			0	Q	0	0	1	1	1	0	0
740	//	FMLA		vector	vector			0	Q	0	0	1	1	1	0	0
741	//	FADD		vector	vector			0	Q	0	0	1	1	1	0	0
742	//	FMULX	v	Vector		Vector		0	Q	0	0	1	1	1	0	0
743	//	FCMEQ	v	register_Vect	register	Vector		0	Q	0	0	1	1	1	0	0
744	//	FMAX		vector	vector			0	Q	0	0	1	1	1	0	0
745	//	FRECPS	v	Vector		Vector		0	Q	0	0	1	1	1	0	0
746	//	AND		vector	vector			0	Q	0	0	1	1	1	0	0
747	//	BIC		vector_registe	vector_regis			0	Q	0	0	1	1	1	0	0
748	//	FMINNM		vector	vector			0	Q	0	0	1	1	1	0	1
749	//	FMLS		vector	vector			0	Q	0	0	1	1	1	0	1
750	//	FSUB		vector	vector			0	Q	0	0	1	1	1	0	1
751	//	FMIN		vector	vector			0	Q	0	0	1	1	1	0	1
752	//	FRSQRTS	v	Vector		Vector		0	Q	0	0	1	1	1	0	1
753	//	ORR		vector_registe	vector_regis			0	Q	0	0	1	1	1	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
754	//	ORN		vector	vector			0	Q	0	0	1	1	1	0	1
755	//	UHADD						0	Q	1	0	1	1	1	0	-
756	//	UQADD	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
757	//	URHADD						0	Q	1	0	1	1	1	0	-
758	//	UHSUB						0	Q	1	0	1	1	1	0	-
759	//	UQSUB	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
760	//	CMHI	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	-
761	//	CMHS	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	-
762	//	USHL	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
763	//	UQSHL	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	-
764	//	URSHL	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
765	//	UQRSHL	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
766	//	UMAX						0	Q	1	0	1	1	1	0	-
767	//	UMIN						0	Q	1	0	1	1	1	0	-
768	//	UABD						0	Q	1	0	1	1	1	0	-
769	//	UABA						0	Q	1	0	1	1	1	0	-
770	//	SUB	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	-
771	//	CMEQ	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	-
772	//	MLS		vector	vector			0	Q	1	0	1	1	1	0	-
773	//	PMUL						0	Q	1	0	1	1	1	0	-
774	//	UMAXP						0	Q	1	0	1	1	1	0	-
775	//	UMINP						0	Q	1	0	1	1	1	0	-
776	//	SQRDMULH	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	-
777	//	FMAXNMP		vector	vector			0	Q	1	0	1	1	1	0	0
778	//	FADDP		vector	vector			0	Q	1	0	1	1	1	0	0
779	//	FMUL		vector	vector			0	Q	1	0	1	1	1	0	0
780	//	FCMGE	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	0
781	//	FACGE	v	Vector		Vector		0	Q	1	0	1	1	1	0	0
782	//	FMAXP		vector	vector			0	Q	1	0	1	1	1	0	0
783	//	FDIV		vector	vector			0	Q	1	0	1	1	1	0	0
784	//	EOR		vector	vector			0	Q	1	0	1	1	1	0	0
785	//	BSL						0	Q	1	0	1	1	1	0	0
786	//	FMINNMP		vector	vector			0	Q	1	0	1	1	1	0	1
787	//	FABD	v	Vector		Vector		0	Q	1	0	1	1	1	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
788	//	FCMGT	v	register_Vect	register	Vector		0	Q	1	0	1	1	1	0	1
789	//	FACGT	v	Vector		Vector		0	Q	1	0	1	1	1	0	1
790	//	FMINP		vector	vector			0	Q	1	0	1	1	1	0	1
791	//	BIT						0	Q	1	0	1	1	1	0	1
792	//	BIF						0	Q	1	0	1	1	1	0	1
793	//	**AdvSIMD three different**						**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**siz**
794	//	SADDL					writes to low	0	0	0	0	1	1	1	0	siz
795	//	SADDL2					writes to high	0	1	0	0	1	1	1	0	siz
796	//	SADDW					writes to low	0	0	0	0	1	1	1	0	siz
797	//	SADDW2					writes to high	0	1	0	0	1	1	1	0	siz
798	//	SSUBL					writes to low	0	0	0	0	1	1	1	0	siz
799	//	SSUBL2					writes to high	0	1	0	0	1	1	1	0	siz
800	//	SSUBW					writes to low	0	0	0	0	1	1	1	0	siz
801	//	SSUBW2					writes to high	0	1	0	0	1	1	1	0	siz
802	//	ADDHN					writes to low	0	0	0	0	1	1	1	0	siz
803	//	ADDHN2					writes to high	0	1	0	0	1	1	1	0	siz
804	//	SABAL					writes to low	0	0	0	0	1	1	1	0	siz
805	//	SABAL2					writes to high	0	1	0	0	1	1	1	0	siz
806	//	SUBHN					writes to low	0	0	0	0	1	1	1	0	siz
807	//	SUBHN2					writes to high	0	1	0	0	1	1	1	0	siz
808	//	SABDL					writes to low	0	0	0	0	1	1	1	0	siz
809	//	SABDL2					writes to high	0	1	0	0	1	1	1	0	siz
810	//	SMLAL		vector	vector		writes to low	0	0	0	0	1	1	1	0	siz
811	//	SMLAL2		vector	vector		writes to high	0	1	0	0	1	1	1	0	siz
812	//	SQDMLAL	v	vector_Vecto	vector	Vector	writes to low	0	0	0	0	1	1	1	0	siz
813	//	SQDMLAL2	v	vector_Vecto	vector	Vector	writes to high	0	1	0	0	1	1	1	0	siz
814	//	SMLSL		vector	vector		writes to low	0	0	0	0	1	1	1	0	siz
815	//	SMLSL2		vector	vector		writes to high	0	1	0	0	1	1	1	0	siz
816	//	SQDMLSL	v	vector_Vecto	vector	Vector	writes to low	0	0	0	0	1	1	1	0	siz
817	//	SQDMLSL2	v	vector_Vecto	vector	Vector	writes to high	0	1	0	0	1	1	1	0	siz
818	//	SMULL		vector	vector		writes to low	0	0	0	0	1	1	1	0	siz
819	//	SMULL2		vector	vector		writes to high	0	1	0	0	1	1	1	0	siz
820	//	SQDMULL	v	vector_Vecto	vector	Vector	writes to low	0	0	0	0	1	1	1	0	siz
821	//	SQDMULL2	v	vector_Vecto	vector	Vector	writes to high	0	1	0	0	1	1	1	0	siz

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
822	//	PMULL					writes to low	0	0	0	0	1	1	1	0	siz
823	//	PMULL2					writes to high	0	1	0	0	1	1	1	0	siz
824	//	UADDL					writes to low	0	0	1	0	1	1	1	0	siz
825	//	UADDL2					writes to high	0	1	1	0	1	1	1	0	siz
826	//	UADDW					writes to low	0	0	1	0	1	1	1	0	siz
827	//	UADDW2					writes to high	0	1	1	0	1	1	1	0	siz
828	//	USUBL					writes to low	0	0	1	0	1	1	1	0	siz
829	//	USUBL2					writes to high	0	1	1	0	1	1	1	0	siz
830	//	USUBW					writes to low	0	0	1	0	1	1	1	0	siz
831	//	USUBW2					writes to high	0	1	1	0	1	1	1	0	siz
832	//	RADDHN					writes to low	0	0	1	0	1	1	1	0	siz
833	//	RADDHN2					writes to high	0	1	1	0	1	1	1	0	siz
834	//	UABAL					writes to low	0	0	1	0	1	1	1	0	siz
835	//	UABAL2					writes to high	0	1	1	0	1	1	1	0	siz
836	//	RSUBHN					writes to low	0	0	1	0	1	1	1	0	siz
837	//	RSUBHN2					writes to high	0	1	1	0	1	1	1	0	siz
838	//	UABDL					writes to low	0	0	1	0	1	1	1	0	siz
839	//	UABDL2					writes to high	0	1	1	0	1	1	1	0	siz
840	//	UMLAL		vector	vector		writes to low	0	0	1	0	1	1	1	0	siz
841	//	UMLAL2		vector	vector		writes to high	0	1	1	0	1	1	1	0	siz
842	//	UMLSL		vector	vector		writes to low	0	0	1	0	1	1	1	0	siz
843	//	UMLSL2		vector	vector		writes to high	0	1	1	0	1	1	1	0	siz
844	//	UMULL		vector	vector		writes to low	0	0	1	0	1	1	1	0	siz
845	//	UMULL2		vector	vector		writes to high	0	1	1	0	1	1	1	0	siz
846	//	**AdvSIMD two-reg misc**						**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**siz**
847	//	REV64						0	Q	0	0	1	1	1	0	-
848	//	REV16		vector	vector			0	Q	0	0	1	1	1	0	-
849	//	SADDLP						0	Q	0	0	1	1	1	0	-
850	//	SUQADD	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
851	//	CLS		vector	vector			0	Q	0	0	1	1	1	0	-
852	//	CNT						0	Q	0	0	1	1	1	0	-
853	//	SADALP						0	Q	0	0	1	1	1	0	-
854	//	SQABS	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
855	//	CMGT	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
856	//	CMEQ	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	-
857	//	CMLT	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	-
858	//	ABS	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
859	//	XTN						0	Q	0	0	1	1	1	0	-
860	//	XTN2						0	Q	0	0	1	1	1	0	-
861	//	SQXTN	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
862	//	SQXTN2	v	Vector		Vector		0	Q	0	0	1	1	1	0	-
863	//	FCVTN						0	Q	0	0	1	1	1	0	0
864	//	FCVTN2						0	Q	0	0	1	1	1	0	0
865	//	FCVTL						0	Q	0	0	1	1	1	0	0
866	//	FCVTL2						0	Q	0	0	1	1	1	0	0
867	//	FRINTN		vector	vector			0	Q	0	0	1	1	1	0	0
868	//	FRINTM		vector	vector			0	Q	0	0	1	1	1	0	0
869	//	FCVTNS	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	0
870	//	FCVTMS	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	0
871	//	FCVTAS	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	0
872	//	SCVTF	v	vector_intege	vector_integ	Vector		0	Q	0	0	1	1	1	0	0
873	//	FCMGT	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	1
874	//	FCMEQ	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	1
875	//	FCMLT	v	zero_Vector	zero	Vector		0	Q	0	0	1	1	1	0	1
876	//	FABS		vector	vector			0	Q	0	0	1	1	1	0	1
877	//	FRINTP		vector	vector			0	Q	0	0	1	1	1	0	1
878	//	FRINTZ		vector	vector			0	Q	0	0	1	1	1	0	1
879	//	FCVTPS	v	vector_Vecto	vector	Vector		0	Q	0	0	1	1	1	0	1
880	//	FCVTZS	v	vector_intege	vector_integ	Vector		0	Q	0	0	1	1	1	0	1
881	//	URECPE						0	Q	0	0	1	1	1	0	1
882	//	FRECPE	v	Vector		Vector		0	Q	0	0	1	1	1	0	1
883	//	REV32		vector	vector			0	Q	1	0	1	1	1	0	-
884	//	UADDLP						0	Q	1	0	1	1	1	0	-
885	//	USQADD	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
886	//	CLZ		vector	vector			0	Q	1	0	1	1	1	0	-
887	//	UADALP						0	Q	1	0	1	1	1	0	-
888	//	SQNEG	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
889	//	CMGE	v	zero_Vector	zero	Vector		0	Q	1	0	1	1	1	0	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
890	//	CMLE	v	zero_Vector	zero	Vector		0	Q	1	0	1	1	1	0	-
891	//	NEG	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	-
892	//	SQXTUN	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
893	//	SQXTUN2	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
894	//	SHLL						0	Q	1	0	1	1	1	0	-
895	//	SHLL2						0	Q	1	0	1	1	1	0	-
896	//	UQXTN	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
897	//	UQXTN2	v	Vector		Vector		0	Q	1	0	1	1	1	0	-
898	//	FCVTXN	v	Vector		Vector		0	Q	1	0	1	1	1	0	0
899	//	FCVTXN2	v	Vector		Vector		0	Q	1	0	1	1	1	0	0
900	//	FRINTA		vector	vector			0	Q	1	0	1	1	1	0	0
901	//	FRINTX		vector	vector			0	Q	1	0	1	1	1	0	0
902	//	FCVTNU	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	0
903	//	FCVTMU	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	0
904	//	FCVTAU	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	0
905	//	UCVTF	v	vector_intege	vector_integ	Vector		0	Q	1	0	1	1	1	0	0
906	//	NOT						0	Q	1	0	1	1	1	0	0
907	//	RBIT		vector	vector			0	Q	1	0	1	1	1	0	0
908	//	FCMGE	v	zero_Vector	zero	Vector		0	Q	1	0	1	1	1	0	1
909	//	FCMLE	v	zero_Vector	zero	Vector		0	Q	1	0	1	1	1	0	1
910	//	FNEG		vector	vector			0	Q	1	0	1	1	1	0	1
911	//	FRINTI		vector	vector			0	Q	1	0	1	1	1	0	1
912	//	FCVTPU	v	vector_Vecto	vector	Vector		0	Q	1	0	1	1	1	0	1
913	//	FCVTZU	v	vector_intege	vector_integ	Vector		0	Q	1	0	1	1	1	0	1
914	//	URSQRTE						0	Q	1	0	1	1	1	0	1
915	//	FRSQRTE	v	Vector		Vector		0	Q	1	0	1	1	1	0	1
916	//	FSQRT		vector	vector			0	Q	1	0	1	1	1	0	1
917	//	**AdvSIMD across lanes**						**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**si**
918	//	SADDLV						0	Q	0	0	1	1	1	0	-
919	//	SMAXV						0	Q	0	0	1	1	1	0	-
920	//	SMINV						0	Q	0	0	1	1	1	0	-
921	//	ADDV						0	Q	0	0	1	1	1	0	-
922	//	UADDLV						0	Q	1	0	1	1	1	0	-
923	//	UMAXV						0	Q	1	0	1	1	1	0	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
924	//	UMINV						0	Q	1	0	1	1	1	0	-
925	//	FMAXNMV						0	Q	1	0	1	1	1	0	0
926	//	FMAXV						0	Q	1	0	1	1	1	0	0
927	//	FMINNMV						0	Q	1	0	1	1	1	0	1
928	//	FMINV						0	Q	1	0	1	1	1	0	1
929	//	**AdvSIMD copy**						**0**	**Q**	**op**	**0**	**1**	**1**	**1**	**0**	**0**
930	//	DUP	v	element_Vec	element	Vector		0	-	0	0	1	1	1	0	0
931	//	DUP		general	general			0	-	0	0	1	1	1	0	0
932	//	SMOV	v	32_bit		32_bit		0	0	0	0	1	1	1	0	0
933	//	UMOV	v	32_bit		32_bit		0	0	0	0	1	1	1	0	0
934	//	INS		general	general			0	1	0	0	1	1	1	0	0
935	//	SMOV	v	64_bit		64_bit		0	1	0	0	1	1	1	0	0
936	//	UMOV	v	64_bit		64_bit		0	1	0	0	1	1	1	0	0
937	//	INS		element	element			0	1	1	0	1	1	1	0	0
938	//	**AdvSIMD vector x indexed ele**						**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**1**	**si**
939	//	SMLAL		by_element	by_element			0	Q	0	0	1	1	1	1	-
940	//	SMLAL2		by_element	by_element			0	Q	0	0	1	1	1	1	-
941	//	SQDMLAL	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
942	//	SQDMLAL2	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
943	//	SMLSL		by_element	by_element			0	Q	0	0	1	1	1	1	-
944	//	SMLSL2		by_element	by_element			0	Q	0	0	1	1	1	1	-
945	//	SQDMLSL	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
946	//	SQDMLSL2	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
947	//	MUL		by_element	by_element			0	Q	0	0	1	1	1	1	-
948	//	SMULL		by_element	by_element			0	Q	0	0	1	1	1	1	-
949	//	SMULL2		by_element	by_element			0	Q	0	0	1	1	1	1	-
950	//	SQDMULL	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
951	//	SQDMULL2	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
952	//	SQDMULH	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
953	//	SQRDMULH	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	-
954	//	FMLA	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	1
955	//	FMLS	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	1
956	//	FMUL	v	by_element_	by_element	Vector		0	Q	0	0	1	1	1	1	1
957	//	MLA		by_element	by_element			0	Q	1	0	1	1	1	1	-

1	in_use	Opcode	prepend	appendage	Specific	variant	comments		31	30	29	28	27	26	25	24	23
958	//	UMLAL		by_element	by_element				0	Q	1	0	1	1	1	1	-
959	//	UMLAL2		by_element	by_element				0	Q	1	0	1	1	1	1	-
960	//	MLS		by_element	by_element				0	Q	1	0	1	1	1	1	-
961	//	UMLSL		by_element	by_element				0	Q	1	0	1	1	1	1	-
962	//	UMLSL2		by_element	by_element				0	Q	1	0	1	1	1	1	-
963	//	UMULL		by_element	by_element				0	Q	1	0	1	1	1	1	-
964	//	UMULL2		by_element	by_element				0	Q	1	0	1	1	1	1	-
965	//	FMULX	v	by_element_\	by_element	Vector			0	Q	1	0	1	1	1	1	1
966	//	**AdvSIMD modified immediate**							**0**	**Q**	**op**	**0**	**1**	**1**	**1**	**1**	**0**
967	//	MOVI	v	32_bit_shifte		32_bit_shifted			0	-	0	0	1	1	1	1	0
968	//	ORR	v	vector_imme	vector_imme	32_bit			0	-	0	0	1	1	1	1	0
969	//	MOVI	v	16_bit_shifte		16_bit_shifted			0	-	0	0	1	1	1	1	0
970	//	ORR	v	vector_imme	vector_imme	16_bit			0	-	0	0	1	1	1	1	0
971	//	MOVI	v	32_bit_shiftin		32_bit_shiftin			0	-	0	0	1	1	1	1	0
972	//	MOVI	v	8_bit		8_bit			0	-	0	0	1	1	1	1	0
973	//	FMOV	v	vector_imme	vector_imme	Single_precisi			0	-	0	0	1	1	1	1	0
974	//	MVNI	v	32_bit_shifte		32_bit_shifted			0	-	1	0	1	1	1	1	0
975	//	BIC	v	vector_imme	vector_imme	32_bit			0	-	1	0	1	1	1	1	0
976	//	MVNI	v	16_bit_shifte		16_bit_shifted			0	-	1	0	1	1	1	1	0
977	//	BIC	v	vector_imme	vector_imme	16_bit			0	-	1	0	1	1	1	1	0
978	//	MVNI	v	32_bit_shiftin		32_bit_shiftin			0	-	1	0	1	1	1	1	0
979	//	MOVI	v	64_bit_scalar		64_bit_scalar			0	0	1	0	1	1	1	1	0
980	//	MOVI	v	64_bit_vector		64_bit_vector			0	1	1	0	1	1	1	1	0
981	//	FMOV	v	vector_imme	vector_imme	Double_precis			0	1	1	0	1	1	1	1	0
982	//	**AdvSIMD shift by immediate**							**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**1**	**0**
983	//	SSHR	v	Vector		Vector			0	Q	0	0	1	1	1	1	0
984	//	SSRA	v	Vector		Vector			0	Q	0	0	1	1	1	1	0
985	//	SRSHR	v	Vector		Vector			0	Q	0	0	1	1	1	1	0
986	//	SRSRA	v	Vector		Vector			0	Q	0	0	1	1	1	1	0
987	//	SHL	v	Vector		Vector			0	Q	0	0	1	1	1	1	0
988	//	SQSHL	v	immediate_V	immediate	Vector			0	Q	0	0	1	1	1	1	0
989	//	SHRN							0	Q	0	0	1	1	1	1	0
990	//	SHRN2							0	Q	0	0	1	1	1	1	0
991	//	RSHRN							0	Q	0	0	1	1	1	1	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
992	//	RSHRN2						0	Q	0	0	1	1	1	1	0
993	//	SQSHRN	v	Vector		Vector		0	Q	0	0	1	1	1	1	0
994	//	SQSHRN2	v	Vector		Vector		0	Q	0	0	1	1	1	1	0
995	//	SQRSHRN	v	Vector		Vector		0	Q	0	0	1	1	1	1	0
996	//	SQRSHRN2	v	Vector		Vector		0	Q	0	0	1	1	1	1	0
997	//	SSHLL						0	Q	0	0	1	1	1	1	0
998	//	SSHLL2						0	Q	0	0	1	1	1	1	0
999	//	SCVTF	v	vector_fixed_	vector_fixed	Vector		0	Q	0	0	1	1	1	1	0
1000	//	FCVTZS	v	vector_fixed_	vector_fixed	Vector		0	Q	0	0	1	1	1	1	0
1001	//	USHR	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1002	//	USRA	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1003	//	URSHR	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1004	//	URSRA	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1005	//	SRI	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1006	//	SLI	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1007	//	SQSHLU	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1008	//	UQSHL	v	immediate_V	immediate	Vector		0	Q	1	0	1	1	1	1	0
1009	//	SQSHRUN	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1010	//	SQSHRUN2	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1011	//	SQRSHRUN	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1012	//	SQRSHRUN2	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1013	//	UQSHRN	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1014	//	UQRSHRN	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1015	//	UQRSHRN2	v	Vector		Vector		0	Q	1	0	1	1	1	1	0
1016	//	USHLL						0	Q	1	0	1	1	1	1	0
1017	//	USHLL2						0	Q	1	0	1	1	1	1	0
1018	//	UCVTF	v	vector_fixed_	vector_fixed	Vector		0	Q	1	0	1	1	1	1	0
1019	//	FCVTZU	v	vector_fixed_	vector_fixed	Vector		0	Q	1	0	1	1	1	1	0
1020	//	**AdvSIMD TBL/TBX**						**0**	**Q**	**0**	**0**	**1**	**1**	**1**	**0**	**op**
1021	//	TBL	v	Single_regist		Single_registe		0	Q	0	0	1	1	1	0	0
1022	//	TBX	v	Single_regist		Single_registe		0	Q	0	0	1	1	1	0	0
1023	//	TBL	v	Two_register_		Two_register_		0	Q	0	0	1	1	1	0	0
1024	//	TBX	v	Two_register_		Two_register_		0	Q	0	0	1	1	1	0	0
1025	//	TBL	v	Three_registe		Three_registe		0	Q	0	0	1	1	1	0	0

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
102(	//	TBX	v	Three_registe		Three_registe		0	Q	0	0	1	1	1	0	0
1027	//	TBL	v	Four_register		Four_register_		0	Q	0	0	1	1	1	0	0
1028	//	TBX	v	Four_register		Four_register_		0	Q	0	0	1	1	1	0	0
1029	//	**AdvSIMD ZIP/UZP/TRN**						**0**	**Q**	**0**	**0**	**1**	**1**	**1**	**0**	**siz**
103(	//	UZP1						0	Q	0	0	1	1	1	0	siz
1031	//	TRN1						0	Q	0	0	1	1	1	0	siz
1032	//	ZIP1						0	Q	0	0	1	1	1	0	siz
1033	//	UZP2						0	Q	0	0	1	1	1	0	siz
1034	//	TRN2						0	Q	0	0	1	1	1	0	siz
1035	//	ZIP2						0	Q	0	0	1	1	1	0	siz
1036	//	**AdvSIMD EXT**						**0**	**Q**	**1**	**0**	**1**	**1**	**1**	**0**	**op**
1037	//	EXT						0	Q	1	0	1	1	1	0	0
1038	//	**Loads and stores**										**1**		**0**		
1039	//	**AdvSIMD load/store multiple**						**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**0**	**0**
104(	//	ST4	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
1041	//	ST1	v	multiple_struc	multiple_stru	Four_registers		0	Q	0	0	1	1	0	0	0
1042	//	ST3	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
1043	//	ST1	v	multiple_struc	multiple_stru	Three_registe		0	Q	0	0	1	1	0	0	0
1044	//	ST1	v	multiple_struc	multiple_stru	One_register		0	Q	0	0	1	1	0	0	0
1045	//	ST2	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
1046	//	ST1	v	multiple_struc	multiple_stru	Two_registers		0	Q	0	0	1	1	0	0	0
1047	//	LD4	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
1048	//	LD1	v	multiple_struc	multiple_stru	Four_registers		0	Q	0	0	1	1	0	0	0
1049	//	LD3	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
105(	//	LD1	v	multiple_struc	multiple_stru	Three_registe		0	Q	0	0	1	1	0	0	0
1051	//	LD1	v	multiple_struc	multiple_stru	One_register		0	Q	0	0	1	1	0	0	0
1052	//	LD2	v	multiple_struc	multiple_stru	No_offset		0	Q	0	0	1	1	0	0	0
1053	//	LD1	v	multiple_struc	multiple_stru	Two_registers		0	Q	0	0	1	1	0	0	0
1054	//	**AdvSIMD load/store multiple**						**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
1055	//	ST4	v	multiple_struc	multiple_stru	Register_offse	Rm != 1111	0	Q	0	0	1	1	0	0	1
1056	//	ST1	v	multiple_struc	multiple_stru	Four_registers	Rm != 1111	0	Q	0	0	1	1	0	0	1
1057	//	ST3	v	multiple_struc	multiple_stru	Register_offse	Rm != 1111	0	Q	0	0	1	1	0	0	1
1058	//	ST1	v	multiple_struc	multiple_stru	Three_registe	Rm != 1111	0	Q	0	0	1	1	0	0	1
1059	//	ST1	v	multiple_struc	multiple_stru	One_register_	Rm != 1111	0	Q	0	0	1	1	0	0	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
1060	//	ST2	v	multiple_struc	multiple_stru	Register_offse	Rm != 11111	0	Q	0	0	1	1	0	0	1
1061	//	ST1	v	multiple_struc	multiple_stru	Two_registers	Rm != 11111	0	Q	0	0	1	1	0	0	1
1062	//	ST4	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1063	//	ST1	v	multiple_struc	multiple_stru	Four_registers		0	Q	0	0	1	1	0	0	1
1064	//	ST3	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1065	//	ST1	v	multiple_struc	multiple_stru	Three_registe		0	Q	0	0	1	1	0	0	1
1066	//	ST1	v	multiple_struc	multiple_stru	One_register_		0	Q	0	0	1	1	0	0	1
1067	//	ST2	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1068	//	ST1	v	multiple_struc	multiple_stru	Two_registers		0	Q	0	0	1	1	0	0	1
1069	//	LD4	v	multiple_struc	multiple_stru	Register_offse	Rm != 11111	0	Q	0	0	1	1	0	0	1
1070	//	LD1	v	multiple_struc	multiple_stru	Four_registers	Rm != 11111	0	Q	0	0	1	1	0	0	1
1071	//	LD3	v	multiple_struc	multiple_stru	Register_offse	Rm != 11111	0	Q	0	0	1	1	0	0	1
1072	//	LD1	v	multiple_struc	multiple_stru	Three_registe	Rm != 11111	0	Q	0	0	1	1	0	0	1
1073	//	LD1	v	multiple_struc	multiple_stru	One_register_	Rm != 11111	0	Q	0	0	1	1	0	0	1
1074	//	LD2	v	multiple_struc	multiple_stru	Register_offse	Rm != 11111	0	Q	0	0	1	1	0	0	1
1075	//	LD1	v	multiple_struc	multiple_stru	Two_registers	Rm != 11111	0	Q	0	0	1	1	0	0	1
1076	//	LD4	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1077	//	LD1	v	multiple_struc	multiple_stru	Four_registers		0	Q	0	0	1	1	0	0	1
1078	//	LD3	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1079	//	LD1	v	multiple_struc	multiple_stru	Three_registe		0	Q	0	0	1	1	0	0	1
1080	//	LD1	v	multiple_struc	multiple_stru	One_register_		0	Q	0	0	1	1	0	0	1
1081	//	LD2	v	multiple_struc	multiple_stru	Immediate_of		0	Q	0	0	1	1	0	0	1
1082	//	LD1	v	multiple_struc	multiple_stru	Two_registers		0	Q	0	0	1	1	0	0	1
1083	//	**AdvSIMD load/store single st**						**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**1**	**0**
1084	//	ST1	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1085	//	ST3	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1086	//	ST1	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1087	//	ST3	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1088	//	ST1	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1089	//	ST1	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1090	//	ST3	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1091	//	ST3	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1092	//	ST2	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1093	//	ST4	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0

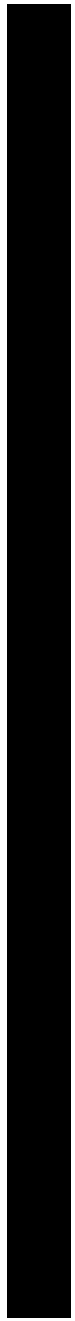
1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
1094	//	ST2	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1095	//	ST4	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1096	//	ST2	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1097	//	ST2	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1098	//	ST4	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1099	//	ST4	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1100	//	LD1	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1101	//	LD3	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1102	//	LD1	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1103	//	LD3	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1104	//	LD1	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1105	//	LD1	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1106	//	LD3	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1107	//	LD3	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1108	//	LD1R	v	No_offset		No_offset		0	Q	0	0	1	1	0	1	0
1109	//	LD3R	v	No_offset		No_offset		0	Q	0	0	1	1	0	1	0
1110	//	LD2	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1111	//	LD4	v	single_structu	single_struc	8_bit		0	Q	0	0	1	1	0	1	0
1112	//	LD2	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1113	//	LD4	v	single_structu	single_struc	16_bit		0	Q	0	0	1	1	0	1	0
1114	//	LD2	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1115	//	LD2	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1116	//	LD4	v	single_structu	single_struc	32_bit		0	Q	0	0	1	1	0	1	0
1117	//	LD4	v	single_structu	single_struc	64_bit		0	Q	0	0	1	1	0	1	0
1118	//	LD2R	v	No_offset		No_offset		0	Q	0	0	1	1	0	1	0
1119	//	LD4R	v	No_offset		No_offset		0	Q	0	0	1	1	0	1	0
1120	//	**AdvSIMD load/store single st**						**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1121	//	ST1	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1122	//	ST3	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1123	//	ST1	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1124	//	ST3	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1125	//	ST1	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1126	//	ST1	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1127	//	ST3	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
1128	//	ST3	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1129	//	ST1	v	single_structu	single_struc	8_bit_immedi		0	Q	0	0	1	1	0	1	1
1130	//	ST3	v	single_structu	single_struc	8_bit_immedi		0	Q	0	0	1	1	0	1	1
1131	//	ST1	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1132	//	ST3	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1133	//	ST1	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1134	//	ST1	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1135	//	ST3	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1136	//	ST3	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1137	//	ST2	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1138	//	ST4	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1139	//	ST2	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1140	//	ST4	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1141	//	ST2	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1142	//	ST2	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1143	//	ST4	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1144	//	ST4	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1145	//	ST2	v	single_structu	single_struc	8_bit_immedi		0	Q	0	0	1	1	0	1	1
1146	//	ST4	v	single_structu	single_struc	8_bit_immedi		0	Q	0	0	1	1	0	1	1
1147	//	ST2	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1148	//	ST4	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1149	//	ST2	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1150	//	ST2	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1151	//	ST4	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1152	//	ST4	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1153	//	LD1	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1154	//	LD3	v	single_structu	single_struc	8_bit_register	Rm != 11111	0	Q	0	0	1	1	0	1	1
1155	//	LD1	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1156	//	LD3	v	single_structu	single_struc	16_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1157	//	LD1	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1158	//	LD1	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1159	//	LD3	v	single_structu	single_struc	32_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1160	//	LD3	v	single_structu	single_struc	64_bit_registe	Rm != 11111	0	Q	0	0	1	1	0	1	1
1161	//	LD1R	v	Register_offs		Register_offse	Rm != 11111	0	Q	0	0	1	1	0	1	1

1	in_use	Opcode	prepend	appendage	Specific	variant	comments	31	30	29	28	27	26	25	24	23
1162	//	LD3R	v	Register_offs		Register_offse	Rm != 1111	0	Q	0	0	1	1	0	1	1
1163	//	LD1	v	single_structu	single_struc	8_bit_immedia		0	Q	0	0	1	1	0	1	1
1164	//	LD3	v	single_structu	single_struc	8_bit_immedia		0	Q	0	0	1	1	0	1	1
1165	//	LD1	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1166	//	LD3	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1167	//	LD1	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1168	//	LD1	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1169	//	LD3	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1170	//	LD3	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1171	//	LD1R	v	Immediate_of		Immediate_of		0	Q	0	0	1	1	0	1	1
1172	//	LD3R	v	Immediate_of		Immediate_of		0	Q	0	0	1	1	0	1	1
1173	//	LD2	v	single_structu	single_struc	8_bit_register	Rm != 1111	0	Q	0	0	1	1	0	1	1
1174	//	LD4	v	single_structu	single_struc	8_bit_register	Rm != 1111	0	Q	0	0	1	1	0	1	1
1175	//	LD2	v	single_structu	single_struc	16_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1176	//	LD4	v	single_structu	single_struc	16_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1177	//	LD2	v	single_structu	single_struc	32_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1178	//	LD2	v	single_structu	single_struc	64_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1179	//	LD4	v	single_structu	single_struc	32_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1180	//	LD4	v	single_structu	single_struc	64_bit_registe	Rm != 1111	0	Q	0	0	1	1	0	1	1
1181	//	LD2R	v	Register_offs		Register_offse	Rm != 1111	0	Q	0	0	1	1	0	1	1
1182	//	LD4R	v	Register_offs		Register_offse	Rm != 1111	0	Q	0	0	1	1	0	1	1
1183	//	LD2	v	single_structu	single_struc	8_bit_immedia		0	Q	0	0	1	1	0	1	1
1184	//	LD4	v	single_structu	single_struc	8_bit_immedia		0	Q	0	0	1	1	0	1	1
1185	//	LD2	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1186	//	LD4	v	single_structu	single_struc	16_bit_immec		0	Q	0	0	1	1	0	1	1
1187	//	LD2	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1188	//	LD2	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1189	//	LD4	v	single_structu	single_struc	32_bit_immec		0	Q	0	0	1	1	0	1	1
1190	//	LD4	v	single_structu	single_struc	64_bit_immec		0	Q	0	0	1	1	0	1	1
1191	//	LD2R	v	Immediate_of		Immediate_of		0	Q	0	0	1	1	0	1	1
1192	//	LD4R	v	Immediate_of		Immediate_of		0	Q	0	0	1	1	0	1	1

	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		UNALLOCATED																							
		BAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		Branch,exception generatic																							
		Compare _ Branch (immediat								**imm19**													**Rt**		
		CBZ								imm19													Rt		
		CBNZ								imm19													Rt		
		CBZ								imm19													Rt		
		CBNZ								imm19													Rt		
		Test bit & branch (immediate)		**b40**									**imm14**										**Rt**		
		TBZ		b40									imm14										Rt		
		TBNZ		b40									imm14										Rt		
		Conditional branch (immediat								**imm19**											**-**		**cond**		
		B_cond								imm19											0		cond		
		Exception generation	**-**	**-**								**imm16**									**-**	**-**	**-**	**-**	**-**
	//	SVC	0	0								imm16									0	0	0	0	1
	//	HVC	0	0								imm16									0	0	0	1	0
	//	SMC	0	0								imm16									0	0	0	1	1
		BRK	0	1								imm16									0	0	0	0	0
	//	HLT	1	0								imm16									0	0	0	0	0
	//	DCPS1	0	1								imm16									0	0	0	0	1
	//	DCPS2	0	1								imm16									0	0	0	1	0
	//	DCPS3	0	1								imm16									0	0	0	1	1
	//	**System**	**0**	**-**	**-**	**-**		**op1**			**CRn**				**CRm**				**op2**				**Rt**		
	//	MSR	0	0	0	0		op1		0	1	0	0		CR_m				op2		1	1	1	1	1
	//	HINT	0	0	0	0	0	1	1	0	0	1	0		CR_m				op2		1	1	1	1	1
	//	CLREX	0	0	0	0	0	1	1	0	0	1	1		CR_m			0	1	0	1	1	1	1	1
	//	DSB	0	0	0	0	0	1	1	0	0	1	1		CR_m			1	0	0	1	1	1	1	1
	//	DMB	0	0	0	0	0	1	1	0	0	1	1		CR_m			1	0	1	1	1	1	1	1
	//	ISB	0	0	0	0	0	1	1	0	0	1	1		CR_m			1	1	0	1	1	1	1	1
	//	SYS	0	0	0	1		op1			CR_n				CR_m				op2				Rt		
	//	MSR	0	0	1	-		op1			CR_n				CR_m				op2				Rt		
	//	SYSL	0	1	0	1		op1			CR_n				CR_m				op2				Rt		
	//	MRS	0	1	1	-		op1			CR_n				CR_m				op2				Rt		
		Unconditional branch (registe	**oc**				**op2**						**op3**					**Rn**					**op4**		
		BR	0	0	1	1	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0
		BLR	0	1	1	1	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0

	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38		RET	1	0	1	1	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0
39	//	ERET	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
40	//	DRPS	0	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
41	//	**Unconditional branch (immed**										**imm26**													
42		B										imm26													
43		BL										imm26													
44		**Loads and stores**																							
45		**Load/store exclusive**	**-**	**-**			**Rs**			**-**			**Rt2**					**Rn**					**Rt**		
46		STXRB	0	0			Rs			0			Rt2					Rn					Rt		
47		STLXRB	0	0			Rs			1			Rt2					Rn					Rt		
48		LDXRB	1	0			Rs			0			Rt2					Rn					Rt		
49		LDAXRB	1	0			Rs			1			Rt2					Rn					Rt		
50		STLRB	0	0			Rs			1			Rt2					Rn					Rt		
51		LDARB	1	0			Rs			1			Rt2					Rn					Rt		
52		STXRH	0	0			Rs			0			Rt2					Rn					Rt		
53		STLXRH	0	0			Rs			1			Rt2					Rn					Rt		
54		LDXRH	1	0			Rs			0			Rt2					Rn					Rt		
55		LDAXRH	1	0			Rs			1			Rt2					Rn					Rt		
56		STLRH	0	0			Rs			1			Rt2					Rn					Rt		
57		LDARH	1	0			Rs			1			Rt2					Rn					Rt		
58		STXR	0	0			Rs			0			Rt2					Rn					Rt		
59		STLXR	0	0			Rs			1			Rt2					Rn					Rt		
60		STXP	0	1			Rs			0			Rt2					Rn					Rt		
61		STLXP	0	1			Rs			1			Rt2					Rn					Rt		
62		LDXR	1	0			Rs			0			Rt2					Rn					Rt		
63		LDAXR	1	0			Rs			1			Rt2					Rn					Rt		
64		LDXP	1	1			Rs			0			Rt2					Rn					Rt		
65		LDAXP	1	1			Rs			1			Rt2					Rn					Rt		
66		STLR	0	0			Rs			1			Rt2					Rn					Rt		
67		LDAR	1	0			Rs			1			Rt2					Rn					Rt		
68		STXR	0	0			Rs			0			Rt2					Rn					Rt		
69		STLXR	0	0			Rs			1			Rt2					Rn					Rt		
70		STXP	0	1			Rs			0			Rt2					Rn					Rt		
71		STLXP	0	1			Rs			1			Rt2					Rn					Rt		
72		LDXR	1	0			Rs			0			Rt2					Rn					Rt		
73		LDAXR	1	0			Rs			1			Rt2					Rn					Rt		
74		LDXP	1	1			Rs			0			Rt2					Rn					Rt		



1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
75		LDAXP	1	1			Rs			1			Rt2					Rn					Rt		
76		STLR	0	0			Rs			1			Rt2					Rn					Rt		
77		LDAR	1	0			Rs			1			Rt2					Rn					Rt		
78		**Load register (literal)**								**imm19**													**Rt**		
79		LDR								imm19													Rt		
80		LDR								imm19													Rt		
81		LDR								imm19													Rt		
82		LDR								imm19													Rt		
83		LDRSW								imm19													Rt		
84		LDR								imm19													Rt		
85		PRFM								imm19													Rt		
86		**Load/store no-allocate pair (o**	**-**				**imm7**						**Rt2**					**Rn**					**Rt**		
87		STNP	0				imm7						Rt2					Rn					Rt		
88		LDNP	1				imm7						Rt2					Rn					Rt		
89		STNP	0				imm7						Rt2					Rn					Rt		
90		LDNP	1				imm7						Rt2					Rn					Rt		
91		STNP	0				imm7						Rt2					Rn					Rt		
92		LDNP	1				imm7						Rt2					Rn					Rt		
93		STNP	0				imm7						Rt2					Rn					Rt		
94		LDNP	1				imm7						Rt2					Rn					Rt		
95		STNP	0				imm7						Rt2					Rn					Rt		
96		LDNP	1				imm7						Rt2					Rn					Rt		
97		**Load/store register pair (post·**	**L**				**imm7**						**Rt2**					**Rn**					**Rt**		
98		STP	0				imm7						Rt2					Rn					Rt		
99		LDP	1				imm7						Rt2					Rn					Rt		
100		STP	0				imm7						Rt2					Rn					Rt		
101		LDP	1				imm7						Rt2					Rn					Rt		
102		LDPSW	1				imm7						Rt2					Rn					Rt		
103		STP	0				imm7						Rt2					Rn					Rt		
104		LDP	1				imm7						Rt2					Rn					Rt		
105		STP	0				imm7						Rt2					Rn					Rt		
106		LDP	1				imm7						Rt2					Rn					Rt		
107		STP	0				imm7						Rt2					Rn					Rt		
108		LDP	1				imm7						Rt2					Rn					Rt		
109		**Load/store register pair (offse**	**L**				**imm7**						**Rt2**					**Rn**					**Rt**		

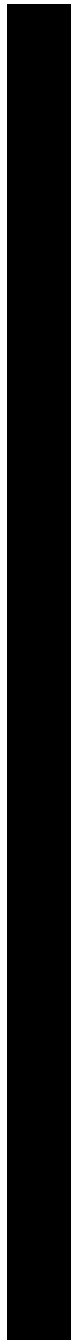
1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110		STP	0			imm7							Rt2					Rn					Rt		
111		LDP	1			imm7							Rt2					Rn					Rt		
112		STP	0			imm7							Rt2					Rn					Rt		
113		LDP	1			imm7							Rt2					Rn					Rt		
114		LDPSW	1			imm7							Rt2					Rn					Rt		
115		STP	0			imm7							Rt2					Rn					Rt		
116		LDP	1			imm7							Rt2					Rn					Rt		
117		STP	0			imm7							Rt2					Rn					Rt		
118		LDP	1			imm7							Rt2					Rn					Rt		
119		STP	0			imm7							Rt2					Rn					Rt		
120		LDP	1			imm7							Rt2					Rn					Rt		
121		**Load/store register pair (pre-i**	**L**			**imm7**							**Rt2**					**Rn**					**Rt**		
122		STP	0			imm7							Rt2					Rn					Rt		
123		LDP	1			imm7							Rt2					Rn					Rt		
124		STP	0			imm7							Rt2					Rn					Rt		
125		LDP	1			imm7							Rt2					Rn					Rt		
126		LDPSW	1			imm7							Rt2					Rn					Rt		
127		STP	0			imm7							Rt2					Rn					Rt		
128		LDP	1			imm7							Rt2					Rn					Rt		
129		STP	0			imm7							Rt2					Rn					Rt		
130		LDP	1			imm7							Rt2					Rn					Rt		
131		STP	0			imm7							Rt2					Rn					Rt		
132		LDP	1			imm7							Rt2					Rn					Rt		
133		**Load/store register (unscaled)c**		**0**					**imm9**					**0**	**0**			**Rn**					**Rt**		
134		STUR**B**	0	0					imm9					0	0			Rn					Rt		
135		LDUR**B**	1	0					imm9					0	0			Rn					Rt		
136		LDURS**B**	0	0					imm9					0	0			Rn					Rt		
137		LDURS**B**	1	0					imm9					0	0			Rn					Rt		
138		STUR	0	0					imm9					0	0			Rn					Rt		
139		LDUR	1	0					imm9					0	0			Rn					Rt		
140		STUR	0	0					imm9					0	0			Rn					Rt		
141		LDUR	1	0					imm9					0	0			Rn					Rt		
142		STURH	0	0					imm9					0	0			Rn					Rt		
143		LDURH	1	0					imm9					0	0			Rn					Rt		
144		LDURSH	0	0					imm9					0	0			Rn					Rt		
145		LDURSH	1	0					imm9					0	0			Rn					Rt		
146		STUR	0	0					imm9					0	0			Rn					Rt		
147		LDUR	1	0					imm9					0	0			Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
148		STUR	0	0					imm9					0	0			Rn					Rt			
149		LDUR	1	0					imm9					0	0			Rn					Rt			
150		LDURSW	0	0					imm9					0	0			Rn					Rt			
151		STUR	0	0					imm9					0	0			Rn					Rt			
152		LDUR	1	0					imm9					0	0			Rn					Rt			
153		STUR	0	0					imm9					0	0			Rn					Rt			
154		LDUR	1	0					imm9					0	0			Rn					Rt			
155		PRFUM	0	0					imm9					0	0			Rn					Rt			
156		STUR	0	0					imm9					0	0			Rn					Rt			
157		LDUR	1	0					imm9					0	0			Rn					Rt			
158		**Load/store register (immediatbc**		**0**					**imm9**					**0**	**1**			**Rn**					**Rt**			
159		STRB	0	0					imm9					0	1			Rn					Rt			
160		LDRB	1	0					imm9					0	1			Rn					Rt			
161		LDRSB	0	0					imm9					0	1			Rn					Rt			
162		LDRSB	1	0					imm9					0	1			Rn					Rt			
163		STR	0	0					imm9					0	1			Rn					Rt			
164		LDR	1	0					imm9					0	1			Rn					Rt			
165		STR	0	0					imm9					0	1			Rn					Rt			
166		LDR	1	0					imm9					0	1			Rn					Rt			
167		STRH	0	0					imm9					0	1			Rn					Rt			
168		LDRH	1	0					imm9					0	1			Rn					Rt			
169		LDRSH	0	0					imm9					0	1			Rn					Rt			
170		LDRSH	1	0					imm9					0	1			Rn					Rt			
171		STR	0	0					imm9					0	1			Rn					Rt			
172		LDR	1	0					imm9					0	1			Rn					Rt			
173		STR	0	0					imm9					0	1			Rn					Rt			
174		LDR	1	0					imm9					0	1			Rn					Rt			
175		LDRSW	0	0					imm9					0	1			Rn					Rt			
176		STR	0	0					imm9					0	1			Rn					Rt			
177		LDR	1	0					imm9					0	1			Rn					Rt			
178		STR	0	0					imm9					0	1			Rn					Rt			
179		LDR	1	0					imm9					0	1			Rn					Rt			
180		STR	0	0					imm9					0	1			Rn					Rt			
181		LDR	1	0					imm9					0	1			Rn					Rt			
182		**Load/store register (unprivileɔc**		**0**					**imm9**					**1**	**0**			**Rn**					**Rt**			
183		STTRB	0	0					imm9					1	0			Rn					Rt			
184		LDTRB	1	0					imm9					1	0			Rn					Rt			
185		LDTRSB	0	0					imm9					1	0			Rn					Rt			

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
186		LDTRSB	1	0					imm9					1	0			Rn					Rt		
187		STTRH	0	0					imm9					1	0			Rn					Rt		
188		LDTRH	1	0					imm9					1	0			Rn					Rt		
189		LDTRSH	0	0					imm9					1	0			Rn					Rt		
190		LDTRSH	1	0					imm9					1	0			Rn					Rt		
191		STTR	0	0					imm9					1	0			Rn					Rt		
192		LDTR	1	0					imm9					1	0			Rn					Rt		
193		LDTRSW	0	0					imm9					1	0			Rn					Rt		
194		STTR	0	0					imm9					1	0			Rn					Rt		
195		LDTR	1	0					imm9					1	0			Rn					Rt		
196		**Load/store register (immediatbc**		**0**					**imm9**					**1**	**1**			**Rn**					**Rt**		
197		STRB	0	0					imm9					1	1			Rn					Rt		
198		LDRB	1	0					imm9					1	1			Rn					Rt		
199		LDRSB	0	0					imm9					1	1			Rn					Rt		
200		LDRSB	1	0					imm9					1	1			Rn					Rt		
201		STR	0	0					imm9					1	1			Rn					Rt		
202		LDR	1	0					imm9					1	1			Rn					Rt		
203		STR	0	0					imm9					1	1			Rn					Rt		
204		LDR	1	0					imm9					1	1			Rn					Rt		
205		STRH	0	0					imm9					1	1			Rn					Rt		
206		LDRH	1	0					imm9					1	1			Rn					Rt		
207		LDRSH	0	0					imm9					1	1			Rn					Rt		
208		LDRSH	1	0					imm9					1	1			Rn					Rt		
209		STR	0	0					imm9					1	1			Rn					Rt		
210		LDR	1	0					imm9					1	1			Rn					Rt		
211		STR	0	0					imm9					1	1			Rn					Rt		
212		LDR	1	0					imm9					1	1			Rn					Rt		
213		LDRSW	0	0					imm9					1	1			Rn					Rt		
214		STR	0	0					imm9					1	1			Rn					Rt		
215		LDR	1	0					imm9					1	1			Rn					Rt		
216		STR	0	0					imm9					1	1			Rn					Rt		
217		LDR	1	0					imm9					1	1			Rn					Rt		
218		STR	0	0					imm9					1	1			Rn					Rt		
219		LDR	1	0					imm9					1	1			Rn					Rt		
220		**Load/store register (register oc**		**1**			**Rm**				**option**		**S**	**1**	**0**			**Rn**					**Rt**		
221		STRB	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
222		LDRB	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
223		LDRSB	0	1			Rm			-	-	-	S	1	0			Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
224		LDRSB	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
225		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
226		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
227		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
228		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
229		STRH	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
230		LDRH	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
231		LDRSH	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
232		LDRSH	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
233		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
234		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
235		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
236		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
237		LDRSW	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
238		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
239		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
240		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
241		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
243		STR	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
244		LDR	1	1			Rm			-	-	-	S	1	0			Rn					Rt		
242		PRFM	0	1			Rm			-	-	-	S	1	0			Rn					Rt		
245		**Load/store register (unsigned**	**oc**						**imm12**									**Rn**					**Rt**		
246		STRB	0						imm12									Rn					Rt		
247		LDRB	1						imm12									Rn					Rt		
248		LDRSB	0						imm12									Rn					Rt		
249		LDRSB	1						imm12									Rn					Rt		
250		STR	0						imm12									Rn					Rt		
251		LDR	1						imm12									Rn					Rt		
252		STR	0						imm12									Rn					Rt		
253		LDR	1						imm12									Rn					Rt		
254		STRH	0						imm12									Rn					Rt		
255		LDRH	1						imm12									Rn					Rt		
256		LDRSH	0						imm12									Rn					Rt		
257		LDRSH	1						imm12									Rn					Rt		
258		STR	0						imm12									Rn					Rt		
259		LDR	1						imm12									Rn					Rt		
260		STR	0						imm12									Rn					Rt		
261		LDR	1						imm12									Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
262		LDRSW	0						imm12									Rn					Rt		
263		STR	0						imm12									Rn					Rt		
264		LDR	1						imm12									Rn					Rt		
265		STR	0						imm12									Rn					Rt		
266		LDR	1						imm12									Rn					Rt		
268		STR	0						imm12									Rn					Rt		
269		LDR	1						imm12									Rn					Rt		
267		PRFM	0						imm12									Rn					Rt		
270	**Data processing – Immedia**																								
271		**PC-rel. addressing**								**immhi**													**Rd**		
272		ADR								immhi													Rd		
273		ADRP								immhi													Rd		
274		**Add/subtract (immediate)**	**ift**						**imm12**									**Rn**					**Rd**		
275		ADD	-						imm12									Rn					Rd		
276		ADDS	-						imm12									Rn					Rd		
277		SUB	-						imm12									Rn					Rd		
278		SUBS	-						imm12									Rn					Rd		
279		ADD	-						imm12									Rn					Rd		
280		ADDS	-						imm12									Rn					Rd		
281		SUB	-						imm12									Rn					Rd		
282		SUBS	-						imm12									Rn					Rd		
283		**Logical (immediate)**	**N**			**immr**					**imms**							**Rn**					**Rd**		
284		AND	0			immr					imms							Rn					Rd		
285		ORR	0			immr					imms							Rn					Rd		
286		EOR	0			immr					imms							Rn					Rd		
287		ANDS	0			immr					imms							Rn					Rd		
288		AND	-			immr					imms							Rn					Rd		
289		ORR	-			immr					imms							Rn					Rd		
290		EOR	-			immr					imms							Rn					Rd		
291		ANDS	-			immr					imms							Rn					Rd		
292		**Move wide (immediate)**	**hw**								**imm16**												**Rd**		
293		MOVN	-	-							imm16												Rd		
294		MOVZ	-	-							imm16												Rd		
295		MOVK	-	-							imm16												Rd		
296		MOVN	-	-							imm16												Rd		
297		MOVZ	-	-							imm16												Rd		
298		MOVK	-	-							imm16												Rd		
299		**Bitfield**	**N**			**immr**					**imms**							**Rn**					**Rd**		



1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
300		SBFM	0			immr						imms						Rn					Rd		
301		BFM	0			immr						imms						Rn					Rd		
302		UBFM	0			immr						imms						Rn					Rd		
303		SBFM	1			immr						imms						Rn					Rd		
304		BFM	1			immr						imms						Rn					Rd		
305		UBFM	1			immr						imms						Rn					Rd		
306		**Extract**	**N**	**o0**		**Rm**						**imms**						**Rn**					**Rd**		
307		EXTR	0	0		Rm				0	-	-	-	-	-			Rn					Rd		
308		EXTR	1	0		Rm				1	-	-	-	-	-			Rn					Rd		
309	**Data Processing – register**																								
310		**Logical (shifted register)**	**ift**	**N**		**Rm**						**imm6**						**Rn**					**Rd**		
311		AND	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
312		BIC	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
313		ORR	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
314		ORN	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
315		EOR	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
316		EON	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
317		ANDS	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
318		BICS	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
319		AND	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
320		BIC	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
321		ORR	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
322		ORN	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
323		EOR	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
324		EON	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
325		ANDS	ift	0		Rm				-	-	-	-	-	-			Rn					Rd		
326		BICS	ift	1		Rm				-	-	-	-	-	-			Rn					Rd		
327		**Add/subtract (shifted register**	**ift**	**0**		**Rm**						**imm6**						**Rn**					**Rd**		
328		ADD	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
329		ADDS	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
330		SUB	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
331		SUBS	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
332		ADD	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
333		ADDS	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
334		SUB	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
335		SUBS	-	0		Rm				-	-	-	-	-	-			Rn					Rd		
336		**Add/subtract (extended regist**	**ɒt**	**1**		**Rm**					**option**			**imm3**				**Rn**					**Rd**		
337		ADD	0	1		Rm					option		-	-	-			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
338		ADDS	0	1			Rm			option			-	-	-			Rn					Rd		
339		SUB	0	1			Rm			option			-	-	-			Rn					Rd		
340		SUBS	0	1			Rm			option			-	-	-			Rn					Rd		
341		ADD	0	1			Rm			option			-	-	-			Rn					Rd		
342		ADDS	0	1			Rm			option			-	-	-			Rn					Rd		
343		SUB	0	1			Rm			option			-	-	-			Rn					Rd		
344		SUBS	0	1			Rm			option			-	-	-			Rn					Rd		
345		**Add/subtract (with carry)**	**0**	**0**			Rm					**opcode2**						**Rn**					**Rd**		
346		ADC	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
347		ADCS	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
348		SBC	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
349		SBCS	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
350		ADC	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
351		ADCS	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
352		SBC	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
353		SBCS	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
354		**Conditional compare (registe**	**1**	**0**			**imm5**				**cond**			**0**	**o2**			**Rn**			**o3**		**nzcv**		
355		CCMN	1	0			imm5				cond			0	0			Rn			0		nzcv		
356		CCMN	1	0			imm5				cond			0	0			Rn			0		nzcv		
357		CCMP	1	0			imm5				cond			0	0			Rn			0		nzcv		
358		CCMP	1	0			imm5				cond			0	0			Rn			0		nzcv		
359		**Conditional compare (immedi**	**1**	**0**			**imm5**				**cond**			**1**	**o2**			**Rn**			**o3**		**nzcv**		
360		CCMN	1	0			imm5				cond			1	0			Rn			0		nzcv		
361		CCMN	1	0			imm5				cond			1	0			Rn			0		nzcv		
362		CCMP	1	0			imm5				cond			1	0			Rn			0		nzcv		
363		CCMP	1	0			imm5				cond			1	0			Rn			0		nzcv		
364		**Conditional select**	**0**	**0**			**Rm**				**cond**			**op2**				**Rn**					**Rd**		
365		CSEL	0	0			Rm				cond			0	0			Rn					Rd		
366		CSINC	0	0			Rm				cond			0	1			Rn					Rd		
367		CSINV	0	0			Rm				cond			0	0			Rn					Rd		
368		CSNEG	0	0			Rm				cond			0	1			Rn					Rd		
369		CSEL	0	0			Rm				cond			0	0			Rn					Rd		
370		CSINC	0	0			Rm				cond			0	1			Rn					Rd		
371		CSINV	0	0			Rm				cond			0	0			Rn					Rd		
372		CSNEG	0	0			Rm				cond			0	1			Rn					Rd		
373		**Data-processing (3 source)**	**op31**				**Rm**			**o0**			**Ra**					**Rn**					**Rd**		
374		MADD	0	0			Rm			0			Ra					Rn					Rd		
375		MADD	0	0			Rm			0			Ra					Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
376		SMADDL	0	1			Rm			0			Ra					Rn					Rd		
377		UMADDL	0	1			Rm			0			Ra					Rn					Rd		
378		MSUB	0	0			Rm			1			Ra					Rn					Rd		
379		MSUB	0	0			Rm			1			Ra					Rn					Rd		
380		SMSUBL	0	1			Rm			1			Ra					Rn					Rd		
381		UMSUBL	0	1			Rm			1			Ra					Rn					Rd		
382		SMULH	1	0			Rm			0			Ra					Rn					Rd		
383		UMULH	1	0			Rm			0			Ra					Rn					Rd		
384		**Data-processing (2 source)**	**1**	**0**			**Rm**					**opcode**						**Rn**					**Rd**		
385		CRC32X	1	0			Rm			0	1	0	0	1	1			Rn					Rd		
386		CRC32CX	1	0			Rm			0	1	0	1	1	1			Rn					Rd		
387		CRC32B	1	0			Rm			0	1	0	0	0	0			Rn					Rd		
388		CRC32CB	1	0			Rm			0	1	0	1	0	0			Rn					Rd		
389		CRC32H	1	0			Rm			0	1	0	0	0	1			Rn					Rd		
390		CRC32CH	1	0			Rm			0	1	0	1	0	1			Rn					Rd		
391		CRC32W	1	0			Rm			0	1	0	0	1	0			Rn					Rd		
392		CRC32CW	1	0			Rm			0	1	0	1	1	0			Rn					Rd		
393		UDIV	1	0			Rm			0	0	0	0	1	0			Rn					Rd		
394		UDIV	1	0			Rm			0	0	0	0	1	0			Rn					Rd		
395		SDIV	1	0			Rm			0	0	0	0	1	1			Rn					Rd		
396		SDIV	1	0			Rm			0	0	0	0	1	1			Rn					Rd		
397		LSLV	1	0			Rm			0	0	1	0	0	0			Rn					Rd		
398		LSLV	1	0			Rm			0	0	1	0	0	0			Rn					Rd		
399		LSRV	1	0			Rm			0	0	1	0	0	1			Rn					Rd		
400		LSRV	1	0			Rm			0	0	1	0	0	1			Rn					Rd		
401		ASRV	1	0			Rm			0	0	1	0	1	0			Rn					Rd		
402		ASRV	1	0			Rm			0	0	1	0	1	0			Rn					Rd		
403		RORV	1	0			Rm			0	0	1	0	1	1			Rn					Rd		
404		RORV	1	0			Rm			0	0	1	0	1	1			Rn					Rd		
405		**Data-processing (1 source)**	**1**	**0**		**opcode2**						**opcode**						**Rn**					**Rd**		
406		RBIT	1	0	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd		
407		RBIT	1	0	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd		
408		CLZ	1	0	0	0	0	0	0	0	0	0	1	0	0			Rn					Rd		
409		CLZ	1	0	0	0	0	0	0	0	0	0	1	0	0			Rn					Rd		
410		CLS	1	0	0	0	0	0	0	0	0	0	1	0	1			Rn					Rd		
411		CLS	1	0	0	0	0	0	0	0	0	0	1	0	1			Rn					Rd		
412		REV	1	0	0	0	0	0	0	0	0	0	0	1	0			Rn					Rd		
413		REV	1	0	0	0	0	0	0	0	0	0	0	1	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
414		REV16	1	0	0	0	0	0	0	0	0	0	0	0	1			Rn					Rd		
415		REV16	1	0	0	0	0	0	0	0	0	0	0	0	1			Rn					Rd		
416		REV32	1	0	0	0	0	0	0	0	0	0	0	1	0			Rn					Rd		
417	//	**Data Processing – SIMD an**																							
418	//	**Floating-point<->fixed-point c**	**pe**	**0**	**rmode**		**opcode**					**scale**						**Rn**					**Rd**		
419	//	SCVTF	0	0	0	0	0	1	0	-	-	-	-	-	-			Rn					Rd		
420	//	UCVTF	0	0	0	0	0	1	1	-	-	-	-	-	-			Rn					Rd		
421	//	FCVTZS	1	0	1	1	0	0	0	-	-	-	-	-	-			Rn					Rd		
422	//	FCVTZU	1	0	1	1	0	0	1	-	-	-	-	-	-			Rn					Rd		
423	//	SCVTF	0	0	0	0	0	1	0	-	-	-	-	-	-			Rn					Rd		
424	//	UCVTF	0	0	0	0	0	1	1	-	-	-	-	-	-			Rn					Rd		
425	//	FCVTZS	1	0	1	1	0	0	0	-	-	-	-	-	-			Rn					Rd		
426	//	FCVTZU	1	0	1	1	0	0	1	-	-	-	-	-	-			Rn					Rd		
427	//	SCVTF	0	0	0	0	0	1	0	-	-	-	-	-	-			Rn					Rd		
428	//	UCVTF	0	0	0	0	0	1	1	-	-	-	-	-	-			Rn					Rd		
429	//	FCVTZS	1	0	1	1	0	0	0	-	-	-	-	-	-			Rn					Rd		
430	//	FCVTZU	1	0	1	1	0	0	1	-	-	-	-	-	-			Rn					Rd		
431	//	SCVTF	0	0	0	0	0	1	0	-	-	-	-	-	-			Rn					Rd		
432	//	UCVTF	0	0	0	0	0	1	1	-	-	-	-	-	-			Rn					Rd		
433	//	FCVTZS	1	0	1	1	0	0	0	-	-	-	-	-	-			Rn					Rd		
434	//	FCVTZU	1	0	1	1	0	0	1	-	-	-	-	-	-			Rn					Rd		
435	//	**Floating-point conditional co**	**pe**	**1**			**Rm**				**cond**			**0**	**1**			**Rn**			**op**		**nzcv**		
436	//	FCCMP	0	1			Rm				cond			0	1			Rn			0		nzcv		
437	//	FCCMPE	0	1			Rm				cond			0	1			Rn			1		nzcv		
438	//	FCCMP	1	1			Rm				cond			0	1			Rn			0		nzcv		
439	//	FCCMPE	1	1			Rm				cond			0	1			Rn			1		nzcv		
440	//	**Floating-point data-processin**	**pe**	**1**			**Rm**				**opcode**			**1**	**0**			**Rn**					**Rd**		
441	//	FMUL	0	1			Rm			0	0	0	0	1	0			Rn					Rd		
442	//	FDIV	0	1			Rm			0	0	0	1	1	0			Rn					Rd		
443	//	FADD	0	1			Rm			0	0	1	0	1	0			Rn					Rd		
444	//	FSUB	0	1			Rm			0	0	1	1	1	0			Rn					Rd		
445	//	FMAX	0	1			Rm			0	1	0	0	1	0			Rn					Rd		
446	//	FMIN	0	1			Rm			0	1	0	1	1	0			Rn					Rd		
447	//	FMAXNM	0	1			Rm			0	1	1	0	1	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
448	//	FMINNM	0	1			Rm			0	1	1	1	1	0			Rn					Rd		
449	//	FNMUL	0	1			Rm			1	0	0	0	1	0			Rn					Rd		
450	//	FMUL	1	1			Rm			0	0	0	0	1	0			Rn					Rd		
451	//	FDIV	1	1			Rm			0	0	0	1	1	0			Rn					Rd		
452	//	FADD	1	1			Rm			0	0	1	0	1	0			Rn					Rd		
453	//	FSUB	1	1			Rm			0	0	1	1	1	0			Rn					Rd		
454	//	FMAX	1	1			Rm			0	1	0	0	1	0			Rn					Rd		
455	//	FMIN	1	1			Rm			0	1	0	1	1	0			Rn					Rd		
456	//	FMAXNM	1	1			Rm			0	1	1	0	1	0			Rn					Rd		
457	//	FMINNM	1	1			Rm			0	1	1	1	1	0			Rn					Rd		
458	//	FNMUL	1	1			Rm			1	0	0	0	1	0			Rn					Rd		
459	//	**Floating-point conditional sel**	**pe**	**1**			**Rm**				**cond**			**1**	**1**			**Rn**					**Rd**		
460	//	FCSEL	0	1			Rm				cond			1	1			Rn					Rd		
461	//	FCSEL	1	1			Rm				cond			1	1			Rn					Rd		
462	//	**Floating-point immediate**	**pe**	**1**				**imm8**					**1**	**0**	**0**			**imm5**					**Rd**		
463	//	FMOV	0	1				imm8					1	0	0	0	0	0	0	0			Rd		
464	//	FMOV	1	1				imm8					1	0	0	0	0	0	0	0			Rd		
465	//	**Floating-point compare**	**pe**	**1**			**Rm**			**op**		**1**	**0**	**0**	**0**			**Rn**				**opcode2**			
466	//	FCMP	0	1			Rm			0	0	1	0	0	0			Rn			0	0	0	0	0
467	//	FCMP	0	1			Rm			0	0	1	0	0	0			Rn			0	1	0	0	0
468	//	FCMPE	0	1			Rm			0	0	1	0	0	0			Rn			1	0	0	0	0
469	//	FCMPE	0	1			Rm			0	0	1	0	0	0			Rn			1	1	0	0	0
470	//	FCMP	1	1			Rm			0	0	1	0	0	0			Rn			0	0	0	0	0
471	//	FCMP	1	1			Rm			0	0	1	0	0	0			Rn			0	1	0	0	0
472	//	FCMPE	1	1			Rm			0	0	1	0	0	0			Rn			1	0	0	0	0
473	//	FCMPE	1	1			Rm			0	0	1	0	0	0			Rn			1	1	0	0	0
474	//	**Floating-point data-processin**	**pe**	**1**			**opcode**				**1**	**0**	**0**	**0**	**0**			**Rn**					**Rd**		
475	//	FMOV	0	1	0	0	0	0	0	0	1	0	0	0	0			Rn					Rd		
476	//	FABS	0	1	0	0	0	0	0	1	1	0	0	0	0			Rn					Rd		
477	//	FNEG	0	1	0	0	0	0	1	0	1	0	0	0	0			Rn					Rd		
478	//	FSQRT	0	1	0	0	0	0	1	1	1	0	0	0	0			Rn					Rd		
479	//	FCVT	0	1	0	0	0	1	0	1	1	0	0	0	0			Rn					Rd		
480	//	FCVT	0	1	0	0	0	1	1	1	1	0	0	0	0			Rn					Rd		
481	//	FRINTN	0	1	0	0	1	0	0	0	1	0	0	0	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
482	//	FRINTP	0	1	0	0	1	0	0	1	1	0	0	0	0			Rn					Rd		
483	//	FRINTM	0	1	0	0	1	0	1	0	1	0	0	0	0			Rn					Rd		
484	//	FRINTZ	0	1	0	0	1	0	1	1	1	0	0	0	0			Rn					Rd		
485	//	FRINTA	0	1	0	0	1	1	0	0	1	0	0	0	0			Rn					Rd		
486	//	FRINTX	0	1	0	0	1	1	1	0	1	0	0	0	0			Rn					Rd		
487	//	FRINTI	0	1	0	0	1	1	1	1	1	0	0	0	0			Rn					Rd		
488	//	FMOV	1	1	0	0	0	0	0	0	1	0	0	0	0			Rn					Rd		
489	//	FABS	1	1	0	0	0	0	0	1	1	0	0	0	0			Rn					Rd		
490	//	FNEG	1	1	0	0	0	0	1	0	1	0	0	0	0			Rn					Rd		
491	//	FSQRT	1	1	0	0	0	0	1	1	1	0	0	0	0			Rn					Rd		
492	//	FCVT	1	1	0	0	0	1	0	0	1	0	0	0	0			Rn					Rd		
493	//	FCVT	1	1	0	0	0	1	1	1	1	0	0	0	0			Rn					Rd		
494	//	FRINTN	1	1	0	0	1	0	0	0	1	0	0	0	0			Rn					Rd		
495	//	FRINTP	1	1	0	0	1	0	0	1	1	0	0	0	0			Rn					Rd		
496	//	FRINTM	1	1	0	0	1	0	1	0	1	0	0	0	0			Rn					Rd		
497	//	FRINTZ	1	1	0	0	1	0	1	1	1	0	0	0	0			Rn					Rd		
498	//	FRINTA	1	1	0	0	1	1	0	0	1	0	0	0	0			Rn					Rd		
499	//	FRINTX	1	1	0	0	1	1	1	0	1	0	0	0	0			Rn					Rd		
500	//	FRINTI	1	1	0	0	1	1	1	1	1	0	0	0	0			Rn					Rd		
501	//	FCVT	1	1	0	0	0	1	0	0	1	0	0	0	0			Rn					Rd		
502	//	FCVT	1	1	0	0	0	1	0	1	1	0	0	0	0			Rn					Rd		
503	//	**Floating-point<->integer conv**	**pe**	**1**	**rmode**		**opcode**			**0**	**0**	**0**	**0**	**0**	**0**			**Rn**					**Rd**		
504	//	FCVTNS	0	1	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd		
505	//	FCVTNU	0	1	0	0	0	0	1	0	0	0	0	0	0			Rn					Rd		
506	//	SCVTF	0	1	0	0	0	1	0	0	0	0	0	0	0			Rn					Rd		
507	//	UCVTF	0	1	0	0	0	1	1	0	0	0	0	0	0			Rn					Rd		
508	//	FCVTAS	0	1	0	0	1	0	0	0	0	0	0	0	0			Rn					Rd		
509	//	FCVTAU	0	1	0	0	1	0	1	0	0	0	0	0	0			Rn					Rd		
510	//	FMOV	0	1	0	0	1	1	0	0	0	0	0	0	0			Rn					Rd		
511	//	FMOV	0	1	0	0	1	1	1	0	0	0	0	0	0			Rn					Rd		
512	//	FCVTPS	0	1	0	1	0	0	0	0	0	0	0	0	0			Rn					Rd		
513	//	FCVTPU	0	1	0	1	0	0	1	0	0	0	0	0	0			Rn					Rd		
514	//	FCVTMS	0	1	1	0	0	0	0	0	0	0	0	0	0			Rn					Rd		
515	//	FCVTMU	0	1	1	0	0	0	1	0	0	0	0	0	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
516	//	FCVTZS	0	1	1	1	0	0	0	0	0	0	0	0	0			Rn					Rd			
517	//	FCVTZU	0	1	1	1	0	0	1	0	0	0	0	0	0			Rn					Rd			
518	//	FCVTNS	1	1	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd			
519	//	FCVTNU	1	1	0	0	0	0	1	0	0	0	0	0	0			Rn					Rd			
520	//	SCVTF	1	1	0	0	0	1	0	0	0	0	0	0	0			Rn					Rd			
521	//	UCVTF	1	1	0	0	0	1	1	0	0	0	0	0	0			Rn					Rd			
522	//	FCVTAS	1	1	0	0	1	0	0	0	0	0	0	0	0			Rn					Rd			
523	//	FCVTAU	1	1	0	0	1	0	1	0	0	0	0	0	0			Rn					Rd			
524	//	FCVTPS	1	1	0	1	0	0	0	0	0	0	0	0	0			Rn					Rd			
525	//	FCVTPU	1	1	0	1	0	0	1	0	0	0	0	0	0			Rn					Rd			
526	//	FCVTMS	1	1	1	0	0	0	0	0	0	0	0	0	0			Rn					Rd			
527	//	FCVTMU	1	1	1	0	0	0	1	0	0	0	0	0	0			Rn					Rd			
528	//	FCVTZS	1	1	1	1	0	0	0	0	0	0	0	0	0			Rn					Rd			
529	//	FCVTZU	1	1	1	1	0	0	1	0	0	0	0	0	0			Rn					Rd			
530	//	FCVTNS	0	1	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd			
531	//	FCVTNU	0	1	0	0	0	0	1	0	0	0	0	0	0			Rn					Rd			
532	//	SCVTF	0	1	0	0	0	1	0	0	0	0	0	0	0			Rn					Rd			
533	//	UCVTF	0	1	0	0	0	1	1	0	0	0	0	0	0			Rn					Rd			
534	//	FCVTAS	0	1	0	0	1	0	0	0	0	0	0	0	0			Rn					Rd			
535	//	FCVTAU	0	1	0	0	1	0	1	0	0	0	0	0	0			Rn					Rd			
536	//	FCVTPS	0	1	0	1	0	0	0	0	0	0	0	0	0			Rn					Rd			
537	//	FCVTPU	0	1	0	1	0	0	1	0	0	0	0	0	0			Rn					Rd			
538	//	FCVTMS	0	1	1	0	0	0	0	0	0	0	0	0	0			Rn					Rd			
539	//	FCVTMU	0	1	1	0	0	0	1	0	0	0	0	0	0			Rn					Rd			
540	//	FCVTZS	0	1	1	1	0	0	0	0	0	0	0	0	0			Rn					Rd			
541	//	FCVTZU	0	1	1	1	0	0	1	0	0	0	0	0	0			Rn					Rd			
542	//	FCVTNS	1	1	0	0	0	0	0	0	0	0	0	0	0			Rn					Rd			
543	//	FCVTNU	1	1	0	0	0	0	1	0	0	0	0	0	0			Rn					Rd			
544	//	SCVTF	1	1	0	0	0	1	0	0	0	0	0	0	0			Rn					Rd			
545	//	UCVTF	1	1	0	0	0	1	1	0	0	0	0	0	0			Rn					Rd			
546	//	FCVTAS	1	1	0	0	1	0	0	0	0	0	0	0	0			Rn					Rd			
547	//	FCVTAU	1	1	0	0	1	0	1	0	0	0	0	0	0			Rn					Rd			
548	//	FMOV	1	1	0	0	1	1	0	0	0	0	0	0	0			Rn					Rd			
549	//	FMOV	1	1	0	0	1	1	1	0	0	0	0	0	0			Rn					Rd			

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
550	//	FCVTPS	1	1	0	1	0	0	0	0	0	0	0	0	0			Rn					Rd		
551	//	FCVTPU	1	1	0	1	0	0	1	0	0	0	0	0	0			Rn					Rd		
552	//	FCVTMS	1	1	1	0	0	0	0	0	0	0	0	0	0			Rn					Rd		
553	//	FCVTMU	1	1	1	0	0	0	1	0	0	0	0	0	0			Rn					Rd		
554	//	FCVTZS	1	1	1	1	0	0	0	0	0	0	0	0	0			Rn					Rd		
555	//	FCVTZU	1	1	1	1	0	0	1	0	0	0	0	0	0			Rn					Rd		
556	//	FMOV	0	1	0	1	1	1	0	0	0	0	0	0	0			Rn					Rd		
557	//	FMOV	0	1	0	1	1	1	1	0	0	0	0	0	0			Rn					Rd		
558	//	**Floating-point data-processinpe**		**o1**			**Rm**			**o0**			**Ra**					**Rn**					**Rd**		
559	//	FMADD	0	0			Rm			0			Ra					Rn					Rd		
560	//	FMSUB	0	0			Rm			1			Ra					Rn					Rd		
561	//	FNMADD	0	1			Rm			0			Ra					Rn					Rd		
562	//	FNMSUB	0	1			Rm			1			Ra					Rn					Rd		
563	//	FMADD	1	0			Rm			0			Ra					Rn					Rd		
564	//	FMSUB	1	0			Rm			1			Ra					Rn					Rd		
565	//	FNMADD	1	1			Rm			0			Ra					Rn					Rd		
566	//	FNMSUB	1	1			Rm			1			Ra					Rn					Rd		
567	//	**AdvSIMD scalar three same**	**ze**	**1**			**Rm**					**opcode**			**1**			**Rn**					**Rd**		
568	//	SQADD	-	1			Rm			0	0	0	0	1	1			Rn					Rd		
569	//	SQSUB	-	1			Rm			0	0	1	0	1	1			Rn					Rd		
570	//	CMGT	-	1			Rm			0	0	1	1	0	1			Rn					Rd		
571	//	CMGE	-	1			Rm			0	0	1	1	1	1			Rn					Rd		
572	//	SSHL	-	1			Rm			0	1	0	0	0	1			Rn					Rd		
573	//	SQSHL	-	1			Rm			0	1	0	0	1	1			Rn					Rd		
574	//	SRSHL	-	1			Rm			0	1	0	1	0	1			Rn					Rd		
575	//	SQRSHL	-	1			Rm			0	1	0	1	1	1			Rn					Rd		
576	//	ADD	-	1			Rm			1	0	0	0	0	1			Rn					Rd		
577	//	CMTST	-	1			Rm			1	0	0	0	1	1			Rn					Rd		
578	//	SQDMULH	-	1			Rm			1	0	1	1	0	1			Rn					Rd		
579	//	FMULX	-	1			Rm			1	1	0	1	1	1			Rn					Rd		
580	//	FCMEQ	-	1			Rm			1	1	1	0	0	1			Rn					Rd		
581	//	FRECPS	-	1			Rm			1	1	1	1	1	1			Rn					Rd		
582	//	FRSQRTS	-	1			Rm			1	1	1	1	1	1			Rn					Rd		
583	//	UQADD	-	1			Rm			0	0	0	0	1	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
584	//	UQSUB	-	1			Rm			0	0	1	0	1	1			Rn					Rd		
585	//	CMHI	-	1			Rm			0	0	1	1	0	1			Rn					Rd		
586	//	CMHS	-	1			Rm			0	0	1	1	1	1			Rn					Rd		
587	//	USHL	-	1			Rm			0	1	0	0	0	1			Rn					Rd		
588	//	UQSHL	-	1			Rm			0	1	0	0	1	1			Rn					Rd		
589	//	URSHL	-	1			Rm			0	1	0	1	0	1			Rn					Rd		
590	//	UQRSHL	-	1			Rm			0	1	0	1	1	1			Rn					Rd		
591	//	SUB	-	1			Rm			1	0	0	0	0	1			Rn					Rd		
592	//	CMEQ	-	1			Rm			1	0	0	0	1	1			Rn					Rd		
593	//	SQRDMULH	-	1			Rm			1	0	1	1	0	1			Rn					Rd		
594	//	FCMGE	-	1			Rm			1	1	1	0	0	1			Rn					Rd		
595	//	FACGE	-	1			Rm			1	1	1	0	1	1			Rn					Rd		
596	//	FABD	-	1			Rm			1	1	0	1	0	1			Rn					Rd		
597	//	FCMGT	-	1			Rm			1	1	1	0	0	1			Rn					Rd		
598	//	FACGT	-	1			Rm			1	1	1	0	1	1			Rn					Rd		
599	//	**AdvSIMD scalar three differerze**		**1**			**Rm**				**opcode**			**0**	**0**			**Rn**					**Rd**		
600	//	SQDMLAL	ze	1			Rm			1	0	0	1	0	0			Rn					Rd		
601	//	SQDMLAL2	ze	1			Rm			1	0	0	1	0	0			Rn					Rd		
602	//	SQDMLSL	ze	1			Rm			1	0	1	1	0	0			Rn					Rd		
603	//	SQDMLSL2	ze	1			Rm			1	0	1	1	0	0			Rn					Rd		
604	//	SQDMULL	ze	1			Rm			1	1	0	1	0	0			Rn					Rd		
605	//	SQDMULL2	ze	1			Rm			1	1	0	1	0	0			Rn					Rd		
606	//	**AdvSIMD scalar two-reg miscze**		**1**	**0**	**0**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**		
607	//	SUQADD	-	1	0	0	0	0	0	0	0	1	1	1	0			Rn					Rd		
608	//	SQABS	-	1	0	0	0	0	0	0	1	1	1	1	0			Rn					Rd		
609	//	CMGT	-	1	0	0	0	0	0	1	0	0	0	1	0			Rn					Rd		
610	//	CMEQ	-	1	0	0	0	0	0	1	0	0	1	1	0			Rn					Rd		
611	//	CMLT	-	1	0	0	0	0	0	1	0	1	0	1	0			Rn					Rd		
612	//	ABS	-	1	0	0	0	0	0	1	0	1	1	1	0			Rn					Rd		
613	//	SQXTN	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
614	//	SQXTN2	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
615	//	FCVTNS	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
616	//	FCVTMS	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
617	//	FCVTAS	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
618	//	SCVTF	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
619	//	FCMGT	-	1	0	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
620	//	FCMEQ	-	1	0	0	0	0	0	1	1	0	1	1	0			Rn					Rd		
621	//	FCMLT	-	1	0	0	0	0	0	1	1	1	0	1	0			Rn					Rd		
622	//	FCVTPS	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
623	//	FCVTZS	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
624	//	FRECPE	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
625	//	FRECPX	-	1	0	0	0	0	1	1	1	1	1	1	0			Rn					Rd		
626	//	USQADD	-	1	0	0	0	0	0	0	0	1	1	1	0			Rn					Rd		
627	//	SQNEG	-	1	0	0	0	0	0	0	1	1	1	1	0			Rn					Rd		
628	//	CMGE	-	1	0	0	0	0	0	1	0	0	0	1	0			Rn					Rd		
629	//	CMLE	-	1	0	0	0	0	0	1	0	0	1	1	0			Rn					Rd		
630	//	NEG	-	1	0	0	0	0	0	1	0	1	1	1	0			Rn					Rd		
631	//	SQXTUN	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd		
632	//	SQXTUN2	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd		
633	//	UQXTN	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
634	//	UQXTN2	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
635	//	FCVTXN	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd		
636	//	FCVTXN2	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd		
637	//	FCVTNU	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
638	//	FCVTMU	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
639	//	FCVTAU	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd		
640	//	UCVTF	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
641	//	FCMGE	-	1	0	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
642	//	FCMLE	-	1	0	0	0	0	0	1	1	0	1	1	0			Rn					Rd		
643	//	FCVTPU	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
644	//	FCVTZU	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
645	//	FRSQRTE	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
646	//	**AdvSIMD scalar pairwise**	**ze**	**1**	**1**	**0**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**		
647	//	ADDP	-	1	1	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
648	//	FMAXNMP	-	1	1	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
649	//	FADDP	-	1	1	0	0	0	0	1	1	0	1	1	0			Rn					Rd		
650	//	FMAXP	-	1	1	0	0	0	0	1	1	1	1	1	0			Rn					Rd		
651	//	FMINNMP	-	1	1	0	0	0	0	1	1	0	0	1	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
652	//	FMINP	-	1	1	0	0	0	0	1	1	1	1	1	0			Rn					Rd		
653	//	**AdvSIMD scalar copy**	**0**	**0**			**imm5**			**0**			**imm4**		**1**			**Rn**					**Rd**		
654	//	DUP	0	0	-	-	-	-	-	0	0	0	0	0	1			Rn					Rd		
655	//	**AdvSIMD scalar x indexed ele**	**ze**	**L**	**M**		**Rm**				**opcode**			**H**	**0**			**Rn**					**Rd**		
656	//	SQDMLAL	-	L	M		Rm			0	0	1	1	H	0			Rn					Rd		
657	//	SQDMLAL2	-	L	M		Rm			0	0	1	1	H	0			Rn					Rd		
658	//	SQDMLSL	-	L	M		Rm			0	1	1	1	H	0			Rn					Rd		
659	//	SQDMLSL2	-	L	M		Rm			0	1	1	1	H	0			Rn					Rd		
660	//	SQDMULL	-	L	M		Rm			1	0	1	1	H	0			Rn					Rd		
661	//	SQDMULL2	-	L	M		Rm			1	0	1	1	H	0			Rn					Rd		
662	//	SQDMULH	-	L	M		Rm			1	1	0	0	H	0			Rn					Rd		
663	//	SQRDMULH	-	L	M		Rm			1	1	0	1	H	0			Rn					Rd		
664	//	FMLA	-	L	M		Rm			0	0	0	1	H	0			Rn					Rd		
665	//	FMLS	-	L	M		Rm			0	1	0	1	H	0			Rn					Rd		
666	//	FMUL	-	L	M		Rm			1	0	0	1	H	0			Rn					Rd		
667	//	FMULX	-	L	M		Rm			1	0	0	1	H	0			Rn					Rd		
668	//	**AdvSIMD scalar shift by imme**		**immh**			**immb**				**opcode**				**1**			**Rn**					**Rd**		
669	//	SSHR		immh			immb			0	0	0	0	0	1			Rn					Rd		
670	//	SSRA		immh			immb			0	0	0	1	0	1			Rn					Rd		
671	//	SRSHR		immh			immb			0	0	1	0	0	1			Rn					Rd		
672	//	SRSRA		immh			immb			0	0	1	1	0	1			Rn					Rd		
673	//	SHL		immh			immb			0	1	0	1	0	1			Rn					Rd		
674	//	SQSHL		immh			immb			0	1	1	1	0	1			Rn					Rd		
675	//	SQSHRN		immh			immb			1	0	0	1	0	1			Rn					Rd		
676	//	SQSHRN2		immh			immb			1	0	0	1	0	1			Rn					Rd		
677	//	SQRSHRN		immh			immb			1	0	0	1	1	1			Rn					Rd		
678	//	SQRSHRN2		immh			immb			1	0	0	1	1	1			Rn					Rd		
679	//	SCVTF		immh			immb			1	1	1	0	0	1			Rn					Rd		
680	//	FCVTZS		immh			immb			1	1	1	1	1	1			Rn					Rd		
681	//	USHR		immh			immb			0	0	0	0	0	1			Rn					Rd		
682	//	USRA		immh			immb			0	0	0	1	0	1			Rn					Rd		
683	//	URSHR		immh			immb			0	0	1	0	0	1			Rn					Rd		
684	//	URSRA		immh			immb			0	0	1	1	0	1			Rn					Rd		
685	//	SRI		immh			immb			0	1	0	0	0	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
686	//	SLI		immh				immb		0	1	0	1	0	1			Rn					Rd		
687	//	SQSHLU		immh				immb		0	1	1	0	0	1			Rn					Rd		
688	//	UQSHL		immh				immb		0	1	1	1	0	1			Rn					Rd		
689	//	SQSHRUN		immh				immb		1	0	0	0	0	1			Rn					Rd		
690	//	SQSHRUN2		immh				immb		1	0	0	0	0	1			Rn					Rd		
691	//	SQRSHRUN		immh				immb		1	0	0	0	1	1			Rn					Rd		
692	//	SQRSHRUN2		immh				immb		1	0	0	0	1	1			Rn					Rd		
693	//	UQSHRN		immh				immb		1	0	0	1	0	1			Rn					Rd		
694	//	UQRSHRN		immh				immb		1	0	0	1	1	1			Rn					Rd		
695	//	UQRSHRN2		immh				immb		1	0	0	1	1	1			Rn					Rd		
696	//	UCVTF		immh				immb		1	1	1	0	0	1			Rn					Rd		
697	//	FCVTZU		immh				immb		1	1	1	1	1	1			Rn					Rd		
698	//	**Crypto three-reg SHA**	**ze**	**0**			**Rm**			**0**	**opcode**			**0**	**0**			**Rn**					**Rd**		
699	//	SHA1C	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
700	//	SHA1P	0	0			Rm			0	0	0	1	0	0			Rn					Rd		
701	//	SHA1M	0	0			Rm			0	0	1	0	0	0			Rn					Rd		
702	//	SHA1SU0	0	0			Rm			0	0	1	1	0	0			Rn					Rd		
703	//	SHA256H	0	0			Rm			0	1	0	0	0	0			Rn					Rd		
704	//	SHA256H2	0	0			Rm			0	1	0	1	0	0			Rn					Rd		
705	//	SHA256SU1	0	0			Rm			0	1	1	0	0	0			Rn					Rd		
706	//	**Crypto two-reg SHA**	**ze**	**1**	**0**	**1**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**		
707	//	SHA1H	0	1	0	1	0	0	0	0	0	0	0	1	0			Rn					Rd		
708	//	SHA1SU1	0	1	0	1	0	0	0	0	0	0	1	1	0			Rn					Rd		
709	//	SHA256SU0	0	1	0	1	0	0	0	0	0	1	0	1	0			Rn					Rd		
710	//	**Crypto AES**	**ze**	**1**	**0**	**1**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**		
711	//	AESE	0	1	0	1	0	0	0	0	1	0	0	1	0			Rn					Rd		
712	//	AESD	0	1	0	1	0	0	0	0	1	0	1	1	0			Rn					Rd		
713	//	AESMC	0	1	0	1	0	0	0	0	1	1	0	1	0			Rn					Rd		
714	//	AESIMC	0	1	0	1	0	0	0	0	1	1	1	1	0			Rn					Rd		
715	//	**AdvSIMD three same**	**ze**	**1**			**Rm**				**opcode**				**1**			**Rn**					**Rd**		
716	//	SHADD	-	1			Rm			0	0	0	0	0	1			Rn					Rd		
717	//	SQADD	-	1			Rm			0	0	0	0	1	1			Rn					Rd		
718	//	SRHADD	-	1			Rm			0	0	0	1	0	1			Rn					Rd		
719	//	SHSUB	-	1			Rm			0	0	1	0	0	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
720	//	SQSUB	-	1			Rm			0	0	1	0	1	1			Rn					Rd		
721	//	CMGT	-	1			Rm			0	0	1	1	0	1			Rn					Rd		
722	//	CMGE	-	1			Rm			0	0	1	1	1	1			Rn					Rd		
723	//	SSHL Vector	-	1			Rm			0	1	0	0	0	1			Rn					Rd		
724	//	SQSHL	-	1			Rm			0	1	0	0	1	1			Rn					Rd		
725	//	SRSHL	-	1			Rm			0	1	0	1	0	1			Rn					Rd		
726	//	SQRSHL	-	1			Rm			0	1	0	1	1	1			Rn					Rd		
727	//	SMAX	-	1			Rm			0	1	1	0	0	1			Rn					Rd		
728	//	SMIN	-	1			Rm			0	1	1	0	1	1			Rn					Rd		
729	//	SABD	-	1			Rm			0	1	1	1	0	1			Rn					Rd		
730	//	SABA	-	1			Rm			0	1	1	1	1	1			Rn					Rd		
731	//	ADD	-	1			Rm			1	0	0	0	0	1			Rn					Rd		
732	//	CMTST	-	1			Rm			1	0	0	0	1	1			Rn					Rd		
733	//	MLA	-	1			Rm			1	0	0	1	0	1			Rn					Rd		
734	//	MUL	-	1			Rm			1	0	0	1	1	1			Rn					Rd		
735	//	SMAXP	-	1			Rm			1	0	1	0	0	1			Rn					Rd		
736	//	SMINP	-	1			Rm			1	0	1	0	1	1			Rn					Rd		
737	//	SQDMULH	-	1			Rm			1	0	1	1	0	1			Rn					Rd		
738	//	ADDP	-	1			Rm			1	0	1	1	1	1			Rn					Rd		
739	//	FMAXNM	-	1			Rm			1	1	0	0	0	1			Rn					Rd		
740	//	FMLA	-	1			Rm			1	1	0	0	1	1			Rn					Rd		
741	//	FADD	-	1			Rm			1	1	0	1	0	1			Rn					Rd		
742	//	FMULX	-	1			Rm			1	1	0	1	1	1			Rn					Rd		
743	//	FCMEQ	-	1			Rm			1	1	1	0	0	1			Rn					Rd		
744	//	FMAX	-	1			Rm			1	1	1	1	0	1			Rn					Rd		
745	//	FRECPS	-	1			Rm			1	1	1	1	1	1			Rn					Rd		
746	//	AND	0	1			Rm			0	0	0	1	1	1			Rn					Rd		
747	//	BIC	1	1			Rm			0	0	0	1	1	1			Rn					Rd		
748	//	FMINNM	-	1			Rm			1	1	0	0	0	1			Rn					Rd		
749	//	FMLS	-	1			Rm			1	1	0	0	1	1			Rn					Rd		
750	//	FSUB	-	1			Rm			1	1	0	1	0	1			Rn					Rd		
751	//	FMIN	-	1			Rm			1	1	1	1	0	1			Rn					Rd		
752	//	FRSQRTS	-	1			Rm			1	1	1	1	1	1			Rn					Rd		
753	//	ORR	0	1			Rm			0	0	0	1	1	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
754	//	ORN	1	1			Rm			0	0	0	1	1	1			Rn					Rd			
755	//	UHADD	-	1			Rm			0	0	0	0	0	1			Rn					Rd			
756	//	UQADD	-	1			Rm			0	0	0	0	1	1			Rn					Rd			
757	//	URHADD	-	1			Rm			0	0	0	1	0	1			Rn					Rd			
758	//	UHSUB	-	1			Rm			0	0	1	0	0	1			Rn					Rd			
759	//	UQSUB	-	1			Rm			0	0	1	0	1	1			Rn					Rd			
760	//	CMHI	-	1			Rm			0	0	1	1	0	1			Rn					Rd			
761	//	CMHS	-	1			Rm			0	0	1	1	1	1			Rn					Rd			
762	//	USHL	-	1			Rm			0	1	0	0	0	1			Rn					Rd			
763	//	UQSHL	-	1			Rm			0	1	0	0	1	1			Rn					Rd			
764	//	URSHL	-	1			Rm			0	1	0	1	0	1			Rn					Rd			
765	//	UQRSHL	-	1			Rm			0	1	0	1	1	1			Rn					Rd			
766	//	UMAX	-	1			Rm			0	1	1	0	0	1			Rn					Rd			
767	//	UMIN	-	1			Rm			0	1	1	0	1	1			Rn					Rd			
768	//	UABD	-	1			Rm			0	1	1	1	0	1			Rn					Rd			
769	//	UABA	-	1			Rm			0	1	1	1	1	1			Rn					Rd			
770	//	SUB	-	1			Rm			1	0	0	0	0	1			Rn					Rd			
771	//	CMEQ	-	1			Rm			1	0	0	0	1	1			Rn					Rd			
772	//	MLS	-	1			Rm			1	0	0	1	0	1			Rn					Rd			
773	//	PMUL	-	1			Rm			1	0	0	1	1	1			Rn					Rd			
774	//	UMAXP	-	1			Rm			1	0	1	0	0	1			Rn					Rd			
775	//	UMINP	-	1			Rm			1	0	1	0	1	1			Rn					Rd			
776	//	SQRDMULH	-	1			Rm			1	0	1	1	0	1			Rn					Rd			
777	//	FMAXNMP	-	1			Rm			1	1	0	0	0	1			Rn					Rd			
778	//	FADDP	-	1			Rm			1	1	0	1	0	1			Rn					Rd			
779	//	FMUL	-	1			Rm			1	1	0	1	1	1			Rn					Rd			
780	//	FCMGE	-	1			Rm			1	1	1	0	0	1			Rn					Rd			
781	//	FACGE	-	1			Rm			1	1	1	0	1	1			Rn					Rd			
782	//	FMAXP	-	1			Rm			1	1	1	1	0	1			Rn					Rd			
783	//	FDIV	-	1			Rm			1	1	1	1	1	1			Rn					Rd			
784	//	EOR	0	1			Rm			0	0	0	1	1	1			Rn					Rd			
785	//	BSL	1	1			Rm			0	0	0	1	1	1			Rn					Rd			
786	//	FMINNMP	-	1			Rm			1	1	0	0	0	1			Rn					Rd			
787	//	FABD	-	1			Rm			1	1	0	1	0	1			Rn					Rd			

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
788	//	FCMGT	-	1			Rm			1	1	1	0	0	1			Rn					Rd		
789	//	FACGT	-	1			Rm			1	1	1	0	1	1			Rn					Rd		
790	//	FMINP	-	1			Rm			1	1	1	1	0	1			Rn					Rd		
791	//	BIT	0	1			Rm			0	0	0	1	1	1			Rn					Rd		
792	//	BIF	1	1			Rm			0	0	0	1	1	1			Rn					Rd		
793	//	**AdvSIMD three different**	**ze**	**1**			**Rm**				**opcode**			**0**	**0**			**Rn**					**Rd**		
794	//	SADDL	ze	1			Rm			0	0	0	0	0	0			Rn					Rd		
795	//	SADDL2	ze	1			Rm			0	0	0	0	0	0			Rn					Rd		
796	//	SADDW	ze	1			Rm			0	0	0	1	0	0			Rn					Rd		
797	//	SADDW2	ze	1			Rm			0	0	0	1	0	0			Rn					Rd		
798	//	SSUBL	ze	1			Rm			0	0	1	0	0	0			Rn					Rd		
799	//	SSUBL2	ze	1			Rm			0	0	1	0	0	0			Rn					Rd		
800	//	SSUBW	ze	1			Rm			0	0	1	1	0	0			Rn					Rd		
801	//	SSUBW2	ze	1			Rm			0	0	1	1	0	0			Rn					Rd		
802	//	ADDHN	ze	1			Rm			0	1	0	0	0	0			Rn					Rd		
803	//	ADDHN2	ze	1			Rm			0	1	0	0	0	0			Rn					Rd		
804	//	SABAL	ze	1			Rm			0	1	0	1	0	0			Rn					Rd		
805	//	SABAL2	ze	1			Rm			0	1	0	1	0	0			Rn					Rd		
806	//	SUBHN	ze	1			Rm			0	1	1	0	0	0			Rn					Rd		
807	//	SUBHN2	ze	1			Rm			0	1	1	0	0	0			Rn					Rd		
808	//	SABDL	ze	1			Rm			0	1	1	1	0	0			Rn					Rd		
809	//	SABDL2	ze	1			Rm			0	1	1	1	0	0			Rn					Rd		
810	//	SMLAL	ze	1			Rm			1	0	0	0	0	0			Rn					Rd		
811	//	SMLAL2	ze	1			Rm			1	0	0	0	0	0			Rn					Rd		
812	//	SQDMLAL	ze	1			Rm			1	0	0	1	0	0			Rn					Rd		
813	//	SQDMLAL2	ze	1			Rm			1	0	0	1	0	0			Rn					Rd		
814	//	SMLSL	ze	1			Rm			1	0	1	0	0	0			Rn					Rd		
815	//	SMLSL2	ze	1			Rm			1	0	1	0	0	0			Rn					Rd		
816	//	SQDMLSL	ze	1			Rm			1	0	1	1	0	0			Rn					Rd		
817	//	SQDMLSL2	ze	1			Rm			1	0	1	1	0	0			Rn					Rd		
818	//	SMULL	ze	1			Rm			1	1	0	0	0	0			Rn					Rd		
819	//	SMULL2	ze	1			Rm			1	1	0	0	0	0			Rn					Rd		
820	//	SQDMULL	ze	1			Rm			1	1	0	1	0	0			Rn					Rd		
821	//	SQDMULL2	ze	1			Rm			1	1	0	1	0	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
822	//	PMULL	ze	1			Rm			1	1	1	0	0	0			Rn					Rd		
823	//	PMULL2	ze	1			Rm			1	1	1	0	0	0			Rn					Rd		
824	//	UADDL	ze	1			Rm			0	0	0	0	0	0			Rn					Rd		
825	//	UADDL2	ze	1			Rm			0	0	0	0	0	0			Rn					Rd		
826	//	UADDW	ze	1			Rm			0	0	0	1	0	0			Rn					Rd		
827	//	UADDW2	ze	1			Rm			0	0	0	1	0	0			Rn					Rd		
828	//	USUBL	ze	1			Rm			0	0	1	0	0	0			Rn					Rd		
829	//	USUBL2	ze	1			Rm			0	0	1	0	0	0			Rn					Rd		
830	//	USUBW	ze	1			Rm			0	0	1	1	0	0			Rn					Rd		
831	//	USUBW2	ze	1			Rm			0	0	1	1	0	0			Rn					Rd		
832	//	RADDHN	ze	1			Rm			0	1	0	0	0	0			Rn					Rd		
833	//	RADDHN2	ze	1			Rm			0	1	0	0	0	0			Rn					Rd		
834	//	UABAL	ze	1			Rm			0	1	0	1	0	0			Rn					Rd		
835	//	UABAL2	ze	1			Rm			0	1	0	1	0	0			Rn					Rd		
836	//	RSUBHN	ze	1			Rm			0	1	1	0	0	0			Rn					Rd		
837	//	RSUBHN2	ze	1			Rm			0	1	1	0	0	0			Rn					Rd		
838	//	UABDL	ze	1			Rm			0	1	1	1	0	0			Rn					Rd		
839	//	UABDL2	ze	1			Rm			0	1	1	1	0	0			Rn					Rd		
840	//	UMLAL	ze	1			Rm			1	0	0	0	0	0			Rn					Rd		
841	//	UMLAL2	ze	1			Rm			1	0	0	0	0	0			Rn					Rd		
842	//	UMLSL	ze	1			Rm			1	0	1	0	0	0			Rn					Rd		
843	//	UMLSL2	ze	1			Rm			1	0	1	0	0	0			Rn					Rd		
844	//	UMULL	ze	1			Rm			1	1	0	0	0	0			Rn					Rd		
845	//	UMULL2	ze	1			Rm			1	1	0	0	0	0			Rn					Rd		
846	//	**AdvSIMD two-reg misc**	**ze**	**1**	**0**	**0**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**		
847	//	REV64	-	1	0	0	0	0	0	0	0	0	0	1	0			Rn					Rd		
848	//	REV16	-	1	0	0	0	0	0	0	0	0	1	1	0			Rn					Rd		
849	//	SADDLP	-	1	0	0	0	0	0	0	0	1	0	1	0			Rn					Rd		
850	//	SUQADD	-	1	0	0	0	0	0	0	0	1	1	1	0			Rn					Rd		
851	//	CLS	-	1	0	0	0	0	0	0	1	0	0	1	0			Rn					Rd		
852	//	CNT	-	1	0	0	0	0	0	0	1	0	1	1	0			Rn					Rd		
853	//	SADALP	-	1	0	0	0	0	0	0	1	1	0	1	0			Rn					Rd		
854	//	SQABS	-	1	0	0	0	0	0	0	1	1	1	1	0			Rn					Rd		
855	//	CMGT	-	1	0	0	0	0	0	1	0	0	0	1	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
856	//	CMEQ	-	1	0	0	0	0	0	1	0	0	1	1	0			Rn					Rd			
857	//	CMLT	-	1	0	0	0	0	0	1	0	1	0	1	0			Rn					Rd			
858	//	ABS	-	1	0	0	0	0	0	1	0	1	1	1	0			Rn					Rd			
859	//	XTN	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd			
860	//	XTN2	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd			
861	//	SQXTN	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd			
862	//	SQXTN2	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd			
863	//	FCVTN	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd			
864	//	FCVTN2	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd			
865	//	FCVTL	-	1	0	0	0	0	1	0	1	1	1	1	0			Rn					Rd			
866	//	FCVTL2	-	1	0	0	0	0	1	0	1	1	1	1	0			Rn					Rd			
867	//	FRINTN	-	1	0	0	0	0	1	1	0	0	0	1	0			Rn					Rd			
868	//	FRINTM	-	1	0	0	0	0	1	1	0	0	1	1	0			Rn					Rd			
869	//	FCVTNS	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd			
870	//	FCVTMS	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd			
871	//	FCVTAS	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd			
872	//	SCVTF	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd			
873	//	FCMGT	-	1	0	0	0	0	0	1	1	0	0	1	0			Rn					Rd			
874	//	FCMEQ	-	1	0	0	0	0	0	1	1	0	1	1	0			Rn					Rd			
875	//	FCMLT	-	1	0	0	0	0	0	1	1	1	0	1	0			Rn					Rd			
876	//	FABS	-	1	0	0	0	0	0	1	1	1	1	1	0			Rn					Rd			
877	//	FRINTP	-	1	0	0	0	0	1	1	0	0	0	1	0			Rn					Rd			
878	//	FRINTZ	-	1	0	0	0	0	1	1	0	0	1	1	0			Rn					Rd			
879	//	FCVTPS	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd			
880	//	FCVTZS	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd			
881	//	URECPE	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd			
882	//	FRECPE	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd			
883	//	REV32	-	1	0	0	0	0	0	0	0	0	0	1	0			Rn					Rd			
884	//	UADDLP	-	1	0	0	0	0	0	0	0	1	0	1	0			Rn					Rd			
885	//	USQADD	-	1	0	0	0	0	0	0	0	1	1	1	0			Rn					Rd			
886	//	CLZ	-	1	0	0	0	0	0	0	1	0	0	1	0			Rn					Rd			
887	//	UADALP	-	1	0	0	0	0	0	0	1	1	0	1	0			Rn					Rd			
888	//	SQNEG	-	1	0	0	0	0	0	0	1	1	1	1	0			Rn					Rd			
889	//	CMGE	-	1	0	0	0	0	0	1	0	0	0	1	0			Rn					Rd			

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
890	//	CMLE	-	1	0	0	0	0	0	1	0	0	1	1	0			Rn					Rd		
891	//	NEG	-	1	0	0	0	0	0	1	0	1	1	1	0			Rn					Rd		
892	//	SQXTUN	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd		
893	//	SQXTUN2	-	1	0	0	0	0	1	0	0	1	0	1	0			Rn					Rd		
894	//	SHLL	-	1	0	0	0	0	1	0	0	1	1	1	0			Rn					Rd		
895	//	SHLL2	-	1	0	0	0	0	1	0	0	1	1	1	0			Rn					Rd		
896	//	UQXTN	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
897	//	UQXTN2	-	1	0	0	0	0	1	0	1	0	0	1	0			Rn					Rd		
898	//	FCVTXN	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd		
899	//	FCVTXN2	-	1	0	0	0	0	1	0	1	1	0	1	0			Rn					Rd		
900	//	FRINTA	-	1	0	0	0	0	1	1	0	0	0	1	0			Rn					Rd		
901	//	FRINTX	-	1	0	0	0	0	1	1	0	0	1	1	0			Rn					Rd		
902	//	FCVTNU	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
903	//	FCVTMU	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
904	//	FCVTAU	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd		
905	//	UCVTF	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
906	//	NOT	0	1	0	0	0	0	0	0	1	0	1	1	0			Rn					Rd		
907	//	RBIT	1	1	0	0	0	0	0	0	1	0	1	1	0			Rn					Rd		
908	//	FCMGE	-	1	0	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
909	//	FCMLE	-	1	0	0	0	0	0	1	1	0	1	1	0			Rn					Rd		
910	//	FNEG	-	1	0	0	0	0	0	1	1	1	1	1	0			Rn					Rd		
911	//	FRINTI	-	1	0	0	0	0	1	1	0	0	1	1	0			Rn					Rd		
912	//	FCVTPU	-	1	0	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
913	//	FCVTZU	-	1	0	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
914	//	URSQRTE	-	1	0	0	0	0	1	1	1	0	0	1	0			Rn					Rd		
915	//	FRSQRTE	-	1	0	0	0	0	1	1	1	0	1	1	0			Rn					Rd		
916	//	FSQRT	-	1	0	0	0	0	1	1	1	1	1	1	0			Rn					Rd		
917	//	**AdvSIMD across lanes**	**ze**	**1**	**1**	**0**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**		
918	//	SADDLV	-	1	1	0	0	0	0	0	0	1	1	1	0			Rn					Rd		
919	//	SMAXV	-	1	1	0	0	0	0	1	0	1	0	1	0			Rn					Rd		
920	//	SMINV	-	1	1	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
921	//	ADDV	-	1	1	0	0	0	1	1	0	1	1	1	0			Rn					Rd		
922	//	UADDLV	-	1	1	0	0	0	0	0	0	1	1	1	0			Rn					Rd		
923	//	UMAXV	-	1	1	0	0	0	0	1	0	1	0	1	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
924	//	UMINV	-	1	1	0	0	0	1	1	0	1	0	1	0			Rn					Rd		
925	//	FMAXNMV	-	1	1	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
926	//	FMAXV	-	1	1	0	0	0	0	1	1	1	1	1	0			Rn					Rd		
927	//	FMINNMV	-	1	1	0	0	0	0	1	1	0	0	1	0			Rn					Rd		
928	//	FMINV	-	1	1	0	0	0	0	1	1	1	1	1	0			Rn					Rd		
929	//	**AdvSIMD copy**	**0**	**0**			**imm5**			**0**			**imm4**		**1**			**Rn**					**Rd**		
930	//	DUP	0	0	-	-	-	-	-	0	0	0	0	0	1			Rn					Rd		
931	//	DUP	0	0	-	-	-	-	-	0	0	0	0	1	1			Rn					Rd		
932	//	SMOV	0	0	-	-	-	-	-	0	0	1	0	1	1			Rn					Rd		
933	//	UMOV	0	0	-	-	-	-	-	0	0	1	1	1	1			Rn					Rd		
934	//	INS	0	0	-	-	-	-	-	0	0	0	1	1	1			Rn					Rd		
935	//	SMOV	0	0	-	-	-	-	-	0	0	1	0	1	1			Rn					Rd		
936	//	UMOV	0	0	-	-	-	-	-	0	0	1	1	1	1			Rn					Rd		
937	//	INS	0	0	-	-	-	-	-	0	-	-	-	-	1			Rn					Rd		
938	//	**AdvSIMD vector x indexed elee**		**L**	**M**		**Rm**				**opcode**			**H**	**0**			**Rn**					**Rd**		
939	//	SMLAL	-	L	M		Rm			0	0	1	0	H	0			Rn					Rd		
940	//	SMLAL2	-	L	M		Rm			0	0	1	0	H	0			Rn					Rd		
941	//	SQDMLAL	-	L	M		Rm			0	0	1	1	H	0			Rn					Rd		
942	//	SQDMLAL2	-	L	M		Rm			0	0	1	1	H	0			Rn					Rd		
943	//	SMLSL	-	L	M		Rm			0	1	1	0	H	0			Rn					Rd		
944	//	SMLSL2	-	L	M		Rm			0	1	1	0	H	0			Rn					Rd		
945	//	SQDMLSL	-	L	M		Rm			0	1	1	1	H	0			Rn					Rd		
946	//	SQDMLSL2	-	L	M		Rm			0	1	1	1	H	0			Rn					Rd		
947	//	MUL	-	L	M		Rm			1	0	0	0	H	0			Rn					Rd		
948	//	SMULL	-	L	M		Rm			1	0	1	0	H	0			Rn					Rd		
949	//	SMULL2	-	L	M		Rm			1	0	1	0	H	0			Rn					Rd		
950	//	SQDMULL	-	L	M		Rm			1	0	1	1	H	0			Rn					Rd		
951	//	SQDMULL2	-	L	M		Rm			1	0	1	1	H	0			Rn					Rd		
952	//	SQDMULH	-	L	M		Rm			1	1	0	0	H	0			Rn					Rd		
953	//	SQRDMULH	-	L	M		Rm			1	1	0	1	H	0			Rn					Rd		
954	//	FMLA	-	L	M		Rm			0	0	0	1	H	0			Rn					Rd		
955	//	FMLS	-	L	M		Rm			0	1	0	1	H	0			Rn					Rd		
956	//	FMUL	-	L	M		Rm			1	0	0	1	H	0			Rn					Rd		
957	//	MLA	-	L	M		Rm			0	0	0	0	H	0			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
958	//	UMLAL	-	L	M		Rm			0	0	1	0	H	0			Rn					Rd		
959	//	UMLAL2	-	L	M		Rm			0	0	1	0	H	0			Rn					Rd		
960	//	MLS	-	L	M		Rm			0	1	0	0	H	0			Rn					Rd		
961	//	UMLSL	-	L	M		Rm			0	1	1	0	H	0			Rn					Rd		
962	//	UMLSL2	-	L	M		Rm			0	1	1	0	H	0			Rn					Rd		
963	//	UMULL	-	L	M		Rm			1	0	1	0	H	0			Rn					Rd		
964	//	UMULL2	-	L	M		Rm			1	0	1	0	H	0			Rn					Rd		
965	//	FMULX	-	L	M		Rm			1	0	0	1	H	0			Rn					Rd		
966	//	**AdvSIMD modified immediate**	**0**	**0**	**0**	**0**	**a**	**b**	**c**		**cmode**			**o2**	**1**	**d**	**e**	**f**	**g**	**h**			**Rd**		
967	//	MOVI	0	0	0	0	a	b	c	0	-	-	0	0	1	d	e	f	g	h			Rd		
968	//	ORR	0	0	0	0	a	b	c	0	-	-	1	0	1	d	e	f	g	h			Rd		
969	//	MOVI	0	0	0	0	a	b	c	1	0	-	0	0	1	d	e	f	g	h			Rd		
970	//	ORR	0	0	0	0	a	b	c	1	0	-	1	0	1	d	e	f	g	h			Rd		
971	//	MOVI	0	0	0	0	a	b	c	1	1	0	-	0	1	d	e	f	g	h			Rd		
972	//	MOVI	0	0	0	0	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd		
973	//	FMOV	0	0	0	0	a	b	c	1	1	1	1	0	1	d	e	f	g	h			Rd		
974	//	MVNI	0	0	0	0	a	b	c	0	-	-	0	0	1	d	e	f	g	h			Rd		
975	//	BIC	0	0	0	0	a	b	c	0	-	-	1	0	1	d	e	f	g	h			Rd		
976	//	MVNI	0	0	0	0	a	b	c	1	0	-	0	0	1	d	e	f	g	h			Rd		
977	//	BIC	0	0	0	0	a	b	c	1	0	-	1	0	1	d	e	f	g	h			Rd		
978	//	MVNI	0	0	0	0	a	b	c	1	1	0	-	0	1	d	e	f	g	h			Rd		
979	//	MOVI	0	0	0	0	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd		
980	//	MOVI	0	0	0	0	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd		
981	//	FMOV	0	0	0	0	a	b	c	1	1	1	1	0	1	d	e	f	g	h			Rd		
982	//	**AdvSIMD shift by immediate**		**immh**			**immb**				**opcode**				**1**			**Rn**					**Rd**		
983	//	SSHR		immh			immb			0	0	0	0	0	1			Rn					Rd		
984	//	SSRA		immh			immb			0	0	0	1	0	1			Rn					Rd		
985	//	SRSHR		immh			immb			0	0	1	0	0	1			Rn					Rd		
986	//	SRSRA		immh			immb			0	0	1	1	0	1			Rn					Rd		
987	//	SHL		immh			immb			0	1	0	1	0	1			Rn					Rd		
988	//	SQSHL		immh			immb			0	1	1	1	0	1			Rn					Rd		
989	//	SHRN		immh			immb			1	0	0	0	0	1			Rn					Rd		
990	//	SHRN2		immh			immb			1	0	0	0	0	1			Rn					Rd		
991	//	RSHRN		immh			immb			1	0	0	0	1	1			Rn					Rd		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
992	//	RSHRN2		immh				immb		1	0	0	0	1	1			Rn					Rd		
993	//	SQSHRN		immh				immb		1	0	0	1	0	1			Rn					Rd		
994	//	SQSHRN2		immh				immb		1	0	0	1	0	1			Rn					Rd		
995	//	SQRSHRN		immh				immb		1	0	0	1	1	1			Rn					Rd		
996	//	SQRSHRN2		immh				immb		1	0	0	1	1	1			Rn					Rd		
997	//	SSHLL		immh				immb		1	0	1	0	0	1			Rn					Rd		
998	//	SSHLL2		immh				immb		1	0	1	0	0	1			Rn					Rd		
999	//	SCVTF		immh				immb		1	1	1	0	0	1			Rn					Rd		
100(	//	FCVTZS		immh				immb		1	1	1	1	1	1			Rn					Rd		
1001	//	USHR		immh				immb		0	0	0	0	0	1			Rn					Rd		
1002	//	USRA		immh				immb		0	0	0	1	0	1			Rn					Rd		
1003	//	URSHR		immh				immb		0	0	1	0	0	1			Rn					Rd		
1004	//	URSRA		immh				immb		0	0	1	1	0	1			Rn					Rd		
1005	//	SRI		immh				immb		0	1	0	0	0	1			Rn					Rd		
1006	//	SLI		immh				immb		0	1	0	1	0	1			Rn					Rd		
1007	//	SQSHLU		immh				immb		0	1	1	0	0	1			Rn					Rd		
1008	//	UQSHL		immh				immb		0	1	1	1	0	1			Rn					Rd		
1009	//	SQSHRUN		immh				immb		1	0	0	0	0	1			Rn					Rd		
101(	//	SQSHRUN2		immh				immb		1	0	0	0	0	1			Rn					Rd		
1011	//	SQRSHRUN		immh				immb		1	0	0	0	1	1			Rn					Rd		
1012	//	SQRSHRUN2		immh				immb		1	0	0	0	1	1			Rn					Rd		
1013	//	UQSHRN		immh				immb		1	0	0	1	0	1			Rn					Rd		
1014	//	UQRSHRN		immh				immb		1	0	0	1	1	1			Rn					Rd		
1015	//	UQRSHRN2		immh				immb		1	0	0	1	1	1			Rn					Rd		
1016	//	USHLL		immh				immb		1	0	1	0	0	1			Rn					Rd		
1017	//	USHLL2		immh				immb		1	0	1	0	0	1			Rn					Rd		
1018	//	UCVTF		immh				immb		1	1	1	0	0	1			Rn					Rd		
1019	//	FCVTZU		immh				immb		1	1	1	1	1	1			Rn					Rd		
102(	//	**AdvSIMD TBL/TBX**	**›2**	**0**			**Rm**			**0**		**len**		**op**	**0**	**0**		**Rn**					**Rd**		
1021	//	TBL	0	0			Rm			0	0	0	0	0	0			Rn					Rd		
1022	//	TBX	0	0			Rm			0	0	0	1	0	0			Rn					Rd		
1023	//	TBL	0	0			Rm			0	0	1	0	0	0			Rn					Rd		
1024	//	TBX	0	0			Rm			0	0	1	1	0	0			Rn					Rd		
1025	//	TBL	0	0			Rm			0	1	0	0	0	0			Rn					Rd		

	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1026	//	TBX	0	0			Rm			0	1	0	1	0	0			Rn					Rd		
1027	//	TBL	0	0			Rm			0	1	1	0	0	0			Rn					Rd		
1028	//	TBX	0	0			Rm			0	1	1	1	0	0			Rn					Rd		
1029	//	**AdvSIMD ZIP/UZP/TRN**	**ze**	**0**			**Rm**			**0**	**opcode**			**1**	**0**			**Rn**					**Rd**		
1030	//	UZP1	ze	0			Rm			0	0	0	1	1	0			Rn					Rd		
1031	//	TRN1	ze	0			Rm			0	0	1	0	1	0			Rn					Rd		
1032	//	ZIP1	ze	0			Rm			0	0	1	1	1	0			Rn					Rd		
1033	//	UZP2	ze	0			Rm			0	1	0	1	1	0			Rn					Rd		
1034	//	TRN2	ze	0			Rm			0	1	1	0	1	0			Rn					Rd		
1035	//	ZIP2	ze	0			Rm			0	1	1	1	1	0			Rn					Rd		
1036	//	**AdvSIMD EXT**	**o2**	**0**			**Rm**			**0**		**imm4**			**0**			**Rn**					**Rd**		
1037	//	EXT	0	0			Rm			0		imm4			0			Rn					Rd		
1038	//	**Loads and stores**																							
1039	//	**AdvSIMD load/store multiple s**	**L**	**0**	**0**	**0**	**0**	**0**	**0**		**opcode**			**size**				**Rn**					**Rt**		
1040	//	ST4	0	0	0	0	0	0	0	0	0	0	0	size				Rn					Rt		
1041	//	ST1	0	0	0	0	0	0	0	0	0	1	0	size				Rn					Rt		
1042	//	ST3	0	0	0	0	0	0	0	0	1	0	0	size				Rn					Rt		
1043	//	ST1	0	0	0	0	0	0	0	0	1	1	0	size				Rn					Rt		
1044	//	ST1	0	0	0	0	0	0	0	0	1	1	1	size				Rn					Rt		
1045	//	ST2	0	0	0	0	0	0	0	1	0	0	0	size				Rn					Rt		
1046	//	ST1	0	0	0	0	0	0	0	1	0	1	0	size				Rn					Rt		
1047	//	LD4	1	0	0	0	0	0	0	0	0	0	0	size				Rn					Rt		
1048	//	LD1	1	0	0	0	0	0	0	0	0	1	0	size				Rn					Rt		
1049	//	LD3	1	0	0	0	0	0	0	0	1	0	0	size				Rn					Rt		
1050	//	LD1	1	0	0	0	0	0	0	0	1	1	0	size				Rn					Rt		
1051	//	LD1	1	0	0	0	0	0	0	0	1	1	1	size				Rn					Rt		
1052	//	LD2	1	0	0	0	0	0	0	1	0	0	0	size				Rn					Rt		
1053	//	LD1	1	0	0	0	0	0	0	1	0	1	0	size				Rn					Rt		
1054	//	**AdvSIMD load/store multiple s**	**L**	**0**			**Rm**				**opcode**			**size**				**Rn**					**Rt**		
1055	//	ST4	0	0			Rm			0	0	0	0	size				Rn					Rt		
1056	//	ST1	0	0			Rm			0	0	1	0	size				Rn					Rt		
1057	//	ST3	0	0			Rm			0	1	0	0	size				Rn					Rt		
1058	//	ST1	0	0			Rm			0	1	1	0	size				Rn					Rt		
1059	//	ST1	0	0			Rm			0	1	1	1	size				Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
106(	//	ST2	0	0			Rm			1	0	0	0	size				Rn					Rt		
1061	//	ST1	0	0			Rm			1	0	1	0	size				Rn					Rt		
1062	//	ST4	0	0	1	1	1	1	1	0	0	0	0	size				Rn					Rt		
1063	//	ST1	0	0	1	1	1	1	1	0	0	1	0	size				Rn					Rt		
1064	//	ST3	0	0	1	1	1	1	1	0	1	0	0	size				Rn					Rt		
1065	//	ST1	0	0	1	1	1	1	1	0	1	1	0	size				Rn					Rt		
1066	//	ST1	0	0	1	1	1	1	1	0	1	1	1	size				Rn					Rt		
1067	//	ST2	0	0	1	1	1	1	1	1	0	0	0	size				Rn					Rt		
1068	//	ST1	0	0	1	1	1	1	1	1	0	1	0	size				Rn					Rt		
1069	//	LD4	1	0			Rm			0	0	0	0	size				Rn					Rt		
107(	//	LD1	1	0			Rm			0	0	1	0	size				Rn					Rt		
1071	//	LD3	1	0			Rm			0	1	0	0	size				Rn					Rt		
1072	//	LD1	1	0			Rm			0	1	1	0	size				Rn					Rt		
1073	//	LD1	1	0			Rm			0	1	1	1	size				Rn					Rt		
1074	//	LD2	1	0			Rm			1	0	0	0	size				Rn					Rt		
1075	//	LD1	1	0			Rm			1	0	1	0	size				Rn					Rt		
1076	//	LD4	1	0	1	1	1	1	1	0	0	0	0	size				Rn					Rt		
1077	//	LD1	1	0	1	1	1	1	1	0	0	1	0	size				Rn					Rt		
1078	//	LD3	1	0	1	1	1	1	1	0	1	0	0	size				Rn					Rt		
1079	//	LD1	1	0	1	1	1	1	1	0	1	1	0	size				Rn					Rt		
108(	//	LD1	1	0	1	1	1	1	1	0	1	1	1	size				Rn					Rt		
1081	//	LD2	1	0	1	1	1	1	1	1	0	0	0	size				Rn					Rt		
1082	//	LD1	1	0	1	1	1	1	1	1	0	1	0	size				Rn					Rt		
1083	//	**AdvSIMD load/store single st**	**L**	**R**	**0**	**0**	**0**	**0**	**0**	**opcode**			**S**	**size**				**Rn**					**Rt**		
1084	//	ST1	0	0	0	0	0	0	0	0	0	0	-	-	-			Rn					Rt		
1085	//	ST3	0	0	0	0	0	0	0	0	0	1	-	-	-			Rn					Rt		
1086	//	ST1	0	0	0	0	0	0	0	0	1	0	-	-	0			Rn					Rt		
1087	//	ST3	0	0	0	0	0	0	0	0	1	1	-	-	0			Rn					Rt		
1088	//	ST1	0	0	0	0	0	0	0	1	0	0	-	0	0			Rn					Rt		
1089	//	ST1	0	0	0	0	0	0	0	1	0	0	0	0	1			Rn					Rt		
109(	//	ST3	0	0	0	0	0	0	0	1	0	1	-	0	0			Rn					Rt		
1091	//	ST3	0	0	0	0	0	0	0	1	0	1	0	0	1			Rn					Rt		
1092	//	ST2	0	1	0	0	0	0	0	0	0	0	-	-	-			Rn					Rt		
1093	//	ST4	0	1	0	0	0	0	0	0	0	1	-	-	-			Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1094	//	ST2	0	1	0	0	0	0	0	0	1	0	-	-	0			Rn					Rt		
1095	//	ST4	0	1	0	0	0	0	0	0	1	1	-	-	0			Rn					Rt		
1096	//	ST2	0	1	0	0	0	0	0	1	0	0	-	0	0			Rn					Rt		
1097	//	ST2	0	1	0	0	0	0	0	1	0	0	0	0	1			Rn					Rt		
1098	//	ST4	0	1	0	0	0	0	0	1	0	1	-	0	0			Rn					Rt		
1099	//	ST4	0	1	0	0	0	0	0	1	0	1	0	0	1			Rn					Rt		
1100	//	LD1	1	0	0	0	0	0	0	0	0	0	-	-	-			Rn					Rt		
1101	//	LD3	1	0	0	0	0	0	0	0	0	1	-	-	-			Rn					Rt		
1102	//	LD1	1	0	0	0	0	0	0	0	1	0	-	-	0			Rn					Rt		
1103	//	LD3	1	0	0	0	0	0	0	0	1	1	-	-	0			Rn					Rt		
1104	//	LD1	1	0	0	0	0	0	0	1	0	0	-	0	0			Rn					Rt		
1105	//	LD1	1	0	0	0	0	0	0	1	0	0	0	0	1			Rn					Rt		
1106	//	LD3	1	0	0	0	0	0	0	1	0	1	-	0	0			Rn					Rt		
1107	//	LD3	1	0	0	0	0	0	0	1	0	1	0	0	1			Rn					Rt		
1108	//	LD1R	1	0	0	0	0	0	0	1	1	0	0	-	-			Rn					Rt		
1109	//	LD3R	1	0	0	0	0	0	0	1	1	1	0	-	-			Rn					Rt		
1110	//	LD2	1	1	0	0	0	0	0	0	0	0	-	-	-			Rn					Rt		
1111	//	LD4	1	1	0	0	0	0	0	0	0	1	-	-	-			Rn					Rt		
1112	//	LD2	1	1	0	0	0	0	0	0	1	0	-	-	0			Rn					Rt		
1113	//	LD4	1	1	0	0	0	0	0	0	1	1	-	-	0			Rn					Rt		
1114	//	LD2	1	1	0	0	0	0	0	1	0	0	-	0	0			Rn					Rt		
1115	//	LD2	1	1	0	0	0	0	0	1	0	0	0	0	1			Rn					Rt		
1116	//	LD4	1	1	0	0	0	0	0	1	0	1	-	0	0			Rn					Rt		
1117	//	LD4	1	1	0	0	0	0	0	1	0	1	0	0	1			Rn					Rt		
1118	//	LD2R	1	1	0	0	0	0	0	1	1	0	0	-	-			Rn					Rt		
1119	//	LD4R	1	1	0	0	0	0	0	1	1	1	0	-	-			Rn					Rt		
1120	//	**AdvSIMD load/store single st**	**L**	**R**			Rm			**opcode**			**S**	**size**				**Rn**					**Rt**		
1121	//	ST1	0	0			Rm			0	0	0	-	-	-			Rn					Rt		
1122	//	ST3	0	0			Rm			0	0	1	-	-	-			Rn					Rt		
1123	//	ST1	0	0			Rm			0	1	0	-	-	0			Rn					Rt		
1124	//	ST3	0	0			Rm			0	1	1	-	-	0			Rn					Rt		
1125	//	ST1	0	0			Rm			1	0	0	-	0	0			Rn					Rt		
1126	//	ST1	0	0			Rm			1	0	0	0	0	1			Rn					Rt		
1127	//	ST3	0	0			Rm			1	0	1	-	0	0			Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1128	//	ST3	0	0			Rm			1	0	1	0	0	1			Rn					Rt		
1129	//	ST1	0	0	1	1	1	1	1	0	0	0	-	-	-			Rn					Rt		
1130	//	ST3	0	0	1	1	1	1	1	0	0	1	-	-	-			Rn					Rt		
1131	//	ST1	0	0	1	1	1	1	1	0	1	0	-	-	0			Rn					Rt		
1132	//	ST3	0	0	1	1	1	1	1	0	1	1	-	-	0			Rn					Rt		
1133	//	ST1	0	0	1	1	1	1	1	1	0	0	-	0	0			Rn					Rt		
1134	//	ST1	0	0	1	1	1	1	1	1	0	0	0	0	1			Rn					Rt		
1135	//	ST3	0	0	1	1	1	1	1	1	0	1	-	0	0			Rn					Rt		
1136	//	ST3	0	0	1	1	1	1	1	1	0	1	0	0	1			Rn					Rt		
1137	//	ST2	0	1			Rm			0	0	0	-	-	-			Rn					Rt		
1138	//	ST4	0	1			Rm			0	0	1	-	-	-			Rn					Rt		
1139	//	ST2	0	1			Rm			0	1	0	-	-	0			Rn					Rt		
1140	//	ST4	0	1			Rm			0	1	1	-	-	0			Rn					Rt		
1141	//	ST2	0	1			Rm			1	0	0	-	0	0			Rn					Rt		
1142	//	ST2	0	1			Rm			1	0	0	0	0	1			Rn					Rt		
1143	//	ST4	0	1			Rm			1	0	1	-	0	0			Rn					Rt		
1144	//	ST4	0	1			Rm			1	0	1	0	0	1			Rn					Rt		
1145	//	ST2	0	1	1	1	1	1	1	0	0	0	-	-	-			Rn					Rt		
1146	//	ST4	0	1	1	1	1	1	1	0	0	1	-	-	-			Rn					Rt		
1147	//	ST2	0	1	1	1	1	1	1	0	1	0	-	-	0			Rn					Rt		
1148	//	ST4	0	1	1	1	1	1	1	0	1	1	-	-	0			Rn					Rt		
1149	//	ST2	0	1	1	1	1	1	1	1	0	0	-	0	0			Rn					Rt		
1150	//	ST2	0	1	1	1	1	1	1	1	0	0	0	0	1			Rn					Rt		
1151	//	ST4	0	1	1	1	1	1	1	1	0	1	-	0	0			Rn					Rt		
1152	//	ST4	0	1	1	1	1	1	1	1	0	1	0	0	1			Rn					Rt		
1153	//	LD1	1	0			Rm			0	0	0	-	-	-			Rn					Rt		
1154	//	LD3	1	0			Rm			0	0	1	-	-	-			Rn					Rt		
1155	//	LD1	1	0			Rm			0	1	0	-	-	0			Rn					Rt		
1156	//	LD3	1	0			Rm			0	1	1	-	-	0			Rn					Rt		
1157	//	LD1	1	0			Rm			1	0	0	-	0	0			Rn					Rt		
1158	//	LD1	1	0			Rm			1	0	0	0	0	1			Rn					Rt		
1159	//	LD3	1	0			Rm			1	0	1	-	0	0			Rn					Rt		
1160	//	LD3	1	0			Rm			1	0	1	0	0	1			Rn					Rt		
1161	//	LD1R	1	0			Rm			1	1	0	0	-	-			Rn					Rt		

1	in_use	Opcode	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1162	//	LD3R	1	0			Rm			1	1	1	0	-	-			Rn					Rt		
1163	//	LD1	1	0	1	1	1	1	1	0	0	0	-	-	-			Rn					Rt		
1164	//	LD3	1	0	1	1	1	1	1	0	0	1	-	-	-			Rn					Rt		
1165	//	LD1	1	0	1	1	1	1	1	0	1	0	-	-	0			Rn					Rt		
1166	//	LD3	1	0	1	1	1	1	1	0	1	1	-	-	0			Rn					Rt		
1167	//	LD1	1	0	1	1	1	1	1	1	0	0	-	0	0			Rn					Rt		
1168	//	LD1	1	0	1	1	1	1	1	1	0	0	0	0	1			Rn					Rt		
1169	//	LD3	1	0	1	1	1	1	1	1	0	1	-	0	0			Rn					Rt		
1170	//	LD3	1	0	1	1	1	1	1	1	0	1	0	0	1			Rn					Rt		
1171	//	LD1R	1	0	1	1	1	1	1	1	1	0	0	-	-			Rn					Rt		
1172	//	LD3R	1	0	1	1	1	1	1	1	1	1	0	-	-			Rn					Rt		
1173	//	LD2	1	1			Rm			0	0	0	-	-	-			Rn					Rt		
1174	//	LD4	1	1			Rm			0	0	1	-	-	-			Rn					Rt		
1175	//	LD2	1	1			Rm			0	1	0	-	-	0			Rn					Rt		
1176	//	LD4	1	1			Rm			0	1	1	-	-	0			Rn					Rt		
1177	//	LD2	1	1			Rm			1	0	0	-	0	0			Rn					Rt		
1178	//	LD2	1	1			Rm			1	0	0	0	0	1			Rn					Rt		
1179	//	LD4	1	1			Rm			1	0	1	-	0	0			Rn					Rt		
1180	//	LD4	1	1			Rm			1	0	1	0	0	1			Rn					Rt		
1181	//	LD2R	1	1			Rm			1	1	0	0	-	-			Rn					Rt		
1182	//	LD4R	1	1			Rm			1	1	1	0	-	-			Rn					Rt		
1183	//	LD2	1	1	1	1	1	1	1	0	0	0	-	-	-			Rn					Rt		
1184	//	LD4	1	1	1	1	1	1	1	0	0	1	-	-	-			Rn					Rt		
1185	//	LD2	1	1	1	1	1	1	1	0	1	0	-	-	0			Rn					Rt		
1186	//	LD4	1	1	1	1	1	1	1	0	1	1	-	-	0			Rn					Rt		
1187	//	LD2	1	1	1	1	1	1	1	1	0	0	-	0	0			Rn					Rt		
1188	//	LD2	1	1	1	1	1	1	1	1	0	0	0	0	1			Rn					Rt		
1189	//	LD4	1	1	1	1	1	1	1	1	0	1	-	0	0			Rn					Rt		
1190	//	LD4	1	1	1	1	1	1	1	1	0	1	0	0	1			Rn					Rt		
1191	//	LD2R	1	1	1	1	1	1	1	1	1	0	0	-	-			Rn					Rt		
1192	//	LD4R	1	1	1	1	1	1	1	1	1	1	0	-	-			Rn					Rt		

	in_use	Opcode	31:30:29:28	Binary
		UNALLOCATED		
		BAD	-:-:-:-:-:-:-:-:-:-	0x00000000
		Branch,exception generatic		
		Compare _ Branch (immediat		
		CBZ	-:-:-:-:-:-:-:imn	0x34000000
		CBNZ	-:-:-:-:-:-:-:imn	0x35000000
		CBZ	-:-:-:-:-:-:-:imn	0xB4000000
		CBNZ	-:-:-:-:-:-:-:imn	0xB5000000
		Test bit & branch (immediate)		
		TBZ	b5:-:-:-:-:-:-:b4	0x36000000
		TBNZ	b5:-:-:-:-:-:-:b4	0x37000000
		Conditional branch (immediat		
		B_cond	-:-:-:-:-:-:-:imn	0x54000000
		Exception generation		
//		SVC	-:-:-:-:-:-:-:-:-:-	0xD4000001
//		HVC	-:-:-:-:-:-:-:-:-:-	0xD4000002
//		SMC	-:-:-:-:-:-:-:-:-:-	0xD4000003
		BRK	-:-:-:-:-:-:-:-:-:-	0xD4200000
//		HLT	-:-:-:-:-:-:-:-:-:-	0xD4400000
//		DCPS1	-:-:-:-:-:-:-:-:-:-	0xD4A00001
//		DCPS2	-:-:-:-:-:-:-:-:-:-	0xD4A00002
//		DCPS3	-:-:-:-:-:-:-:-:-:-	0xD4A00003
//		**System**		
//		MSR	-:-:-:-:-:-:-:-:-:-	0xD500401F
//		HINT	-:-:-:-:-:-:-:-:-:-	0xD503201F
//		CLREX	-:-:-:-:-:-:-:-:-:-	0xD503305F
//		DSB	-:-:-:-:-:-:-:-:-:-	0xD503309F
//		DMB	-:-:-:-:-:-:-:-:-:-	0xD50330BF
//		ISB	-:-:-:-:-:-:-:-:-:-	0xD50330DF
//		SYS	-:-:-:-:-:-:-:-:-:-	0xD5080000
//		MSR	-:-:-:-:-:-:-:-:-:-	0xD5100000
//		SYSL	-:-:-:-:-:-:-:-:-:-	0xD5280000
//		MRS	-:-:-:-:-:-:-:-:-:-	0xD5300000
		Unconditional branch (registe		
		BR	-:-:-:-:-:-:-:-:-:-	0xD61F0000
		BLR	-:-:-:-:-:-:-:-:-:-	0xD63F0000

	in_use	Opcode	31:30:29:28	Binary
38		RET	-:-:-:-:-:-:-:-:-:-	0xD65F0000
39	//	ERET	-:-:-:-:-:-:-:-:-:-	0xD69F03E0
40	//	DRPS	-:-:-:-:-:-:-:-:-:-	0xD6BF03E0
41	//	**Unconditional branch (immed**		
42		B	-:-:-:-:-:-:imm26	0x14000000
43		BL	-:-:-:-:-:-:imm26	0x94000000
44		**Loads and stores**		
45		**Load/store exclusive**		
46		STXRB	-:-:-:-:-:-:-:-:-:-	0x08000000
47		STLXRB	-:-:-:-:-:-:-:-:-:-	0x08008000
48		LDXRB	-:-:-:-:-:-:-:-:-:-	0x08400000
49		LDAXRB	-:-:-:-:-:-:-:-:-:-	0x08408000
50		STLRB	-:-:-:-:-:-:-:-:-:-	0x08808000
51		LDARB	-:-:-:-:-:-:-:-:-:-	0x08C08000
52		STXRH	-:-:-:-:-:-:-:-:-:-	0x48000000
53		STLXRH	-:-:-:-:-:-:-:-:-:-	0x48008000
54		LDXRH	-:-:-:-:-:-:-:-:-:-	0x48400000
55		LDAXRH	-:-:-:-:-:-:-:-:-:-	0x48408000
56		STLRH	-:-:-:-:-:-:-:-:-:-	0x48808000
57		LDARH	-:-:-:-:-:-:-:-:-:-	0x48C08000
58		STXR	-:-:-:-:-:-:-:-:-:-	0x88000000
59		STLXR	-:-:-:-:-:-:-:-:-:-	0x88008000
60		STXP	-:-:-:-:-:-:-:-:-:-	0x88200000
61		STLXP	-:-:-:-:-:-:-:-:-:-	0x88208000
62		LDXR	-:-:-:-:-:-:-:-:-:-	0x88400000
63		LDAXR	-:-:-:-:-:-:-:-:-:-	0x88408000
64		LDXP	-:-:-:-:-:-:-:-:-:-	0x88600000
65		LDAXP	-:-:-:-:-:-:-:-:-:-	0x88608000
66		STLR	-:-:-:-:-:-:-:-:-:-	0x88808000
67		LDAR	-:-:-:-:-:-:-:-:-:-	0x88C08000
68		STXR	-:-:-:-:-:-:-:-:-:-	0xC8000000
69		STLXR	-:-:-:-:-:-:-:-:-:-	0xC8008000
70		STXP	-:-:-:-:-:-:-:-:-:-	0xC8200000
71		STLXP	-:-:-:-:-:-:-:-:-:-	0xC8208000
72		LDXR	-:-:-:-:-:-:-:-:-:-	0xC8400000
73		LDAXR	-:-:-:-:-:-:-:-:-:-	0xC8408000
74		LDXP	-:-:-:-:-:-:-:-:-:-	0xC8600000

1	in_use	Opcode	31:30:29:28	Binary
75		LDAXP	-:-:-:-:-:-:-:-:-:-	0xC8608000
76		STLR	-:-:-:-:-:-:-:-:-:-	0xC8808000
77		LDAR	-:-:-:-:-:-:-:-:-:-	0xC8C08000
78		**Load register (literal)**		
79		LDR	-:-:-:-:-:-:-:-:imn	0x18000000
80		LDR	-:-:-:-:-:-:-:-:imn	0x1C000000
81		LDR	-:-:-:-:-:-:-:-:imn	0x58000000
82		LDR	-:-:-:-:-:-:-:-:imn	0x5C000000
83		LDRSW	-:-:-:-:-:-:-:-:imn	0x98000000
84		LDR	-:-:-:-:-:-:-:-:imn	0x9C000000
85		PRFM	-:-:-:-:-:-:-:-:imn	0xD8000000
86		**Load/store no-allocate pair (o**		
87		STNP	-:-:-:-:-:-:-:-:-:i	0x28000000
88		LDNP	-:-:-:-:-:-:-:-:-:i	0x28400000
89		STNP	-:-:-:-:-:-:-:-:-:i	0x2C000000
90		LDNP	-:-:-:-:-:-:-:-:-:i	0x2C400000
91		STNP	-:-:-:-:-:-:-:-:-:i	0x6C000000
92		LDNP	-:-:-:-:-:-:-:-:-:i	0x6C400000
93		STNP	-:-:-:-:-:-:-:-:-:i	0xA8000000
94		LDNP	-:-:-:-:-:-:-:-:-:i	0xA8400000
95		STNP	-:-:-:-:-:-:-:-:-:i	0xAC000000
96		LDNP	-:-:-:-:-:-:-:-:-:i	0xAC400000
97		**Load/store register pair (post**		
98		STP	-:-:-:-:-:-:-:-:-:i	0x28800000
99		LDP	-:-:-:-:-:-:-:-:-:i	0x28C00000
100		STP	-:-:-:-:-:-:-:-:-:i	0x2C800000
101		LDP	-:-:-:-:-:-:-:-:-:i	0x2CC00000
102		LDPSW	-:-:-:-:-:-:-:-:-:i	0x68C00000
103		STP	-:-:-:-:-:-:-:-:-:i	0x6C800000
104		LDP	-:-:-:-:-:-:-:-:-:i	0x6CC00000
105		STP	-:-:-:-:-:-:-:-:-:i	0xA8800000
106		LDP	-:-:-:-:-:-:-:-:-:i	0xA8C00000
107		STP	-:-:-:-:-:-:-:-:-:i	0xAC800000
108		LDP	-:-:-:-:-:-:-:-:-:i	0xACC00000
109		**Load/store register pair (offse**		

1	in_use	Opcode	31:30:29:28	Binary
110		STP	-:-:-:-:-:-:-:-:-:i	0x29000000
111		LDP	-:-:-:-:-:-:-:-:-:i	0x29400000
112		STP	-:-:-:-:-:-:-:-:-:i	0x2D000000
113		LDP	-:-:-:-:-:-:-:-:-:i	0x2D400000
114		LDPSW	-:-:-:-:-:-:-:-:-:i	0x69400000
115		STP	-:-:-:-:-:-:-:-:-:i	0x6D000000
116		LDP	-:-:-:-:-:-:-:-:-:i	0x6D400000
117		STP	-:-:-:-:-:-:-:-:-:i	0xA9000000
118		LDP	-:-:-:-:-:-:-:-:-:i	0xA9400000
119		STP	-:-:-:-:-:-:-:-:-:i	0xAD000000
120		LDP	-:-:-:-:-:-:-:-:-:i	0xAD400000
121		**Load/store register pair (pre-i**		
122		STP	-:-:-:-:-:-:-:-:-:i	0x29800000
123		LDP	-:-:-:-:-:-:-:-:-:i	0x29C00000
124		STP	-:-:-:-:-:-:-:-:-:i	0x2D800000
125		LDP	-:-:-:-:-:-:-:-:-:i	0x2DC00000
126		LDPSW	-:-:-:-:-:-:-:-:-:i	0x69C00000
127		STP	-:-:-:-:-:-:-:-:-:i	0x6D800000
128		LDP	-:-:-:-:-:-:-:-:-:i	0x6DC00000
129		STP	-:-:-:-:-:-:-:-:-:i	0xA9800000
130		LDP	-:-:-:-:-:-:-:-:-:i	0xA9C00000
131		STP	-:-:-:-:-:-:-:-:-:i	0xAD800000
132		LDP	-:-:-:-:-:-:-:-:-:i	0xADC00000
133		**Load/store register (unscaled**		
134		STURB	-:-:-:-:-:-:-:-:-:-	0x38000000
135		LDURB	-:-:-:-:-:-:-:-:-:-	0x38400000
136		LDURSB	-:-:-:-:-:-:-:-:-:-	0x38800000
137		LDURSB	-:-:-:-:-:-:-:-:-:-	0x38C00000
138		STUR	-:-:-:-:-:-:-:-:-:-	0x3C000000
139		LDUR	-:-:-:-:-:-:-:-:-:-	0x3C400000
140		STUR	-:-:-:-:-:-:-:-:-:-	0x3C800000
141		LDUR	-:-:-:-:-:-:-:-:-:-	0x3CC00000
142		STURH	-:-:-:-:-:-:-:-:-:-	0x78000000
143		LDURH	-:-:-:-:-:-:-:-:-:-	0x78400000
144		LDURSH	-:-:-:-:-:-:-:-:-:-	0x78800000
145		LDURSH	-:-:-:-:-:-:-:-:-:-	0x78C00000
146		STUR	-:-:-:-:-:-:-:-:-:-	0x7C000000
147		LDUR	-:-:-:-:-:-:-:-:-:-	0x7C400000

1	in_use	Opcode	31:30:29:28	Binary
148		STUR	-:-:-:-:-:-:-:-:-:-	0xB8000000
149		LDUR	-:-:-:-:-:-:-:-:-:-	0xB8400000
150		LDURSW	-:-:-:-:-:-:-:-:-:-	0xB8800000
151		STUR	-:-:-:-:-:-:-:-:-:-	0xBC000000
152		LDUR	-:-:-:-:-:-:-:-:-:-	0xBC400000
153		STUR	-:-:-:-:-:-:-:-:-:-	0xF8000000
154		LDUR	-:-:-:-:-:-:-:-:-:-	0xF8400000
155		PRFUM	-:-:-:-:-:-:-:-:-:-	0xF8800000
156		STUR	-:-:-:-:-:-:-:-:-:-	0xFC000000
157		LDUR	-:-:-:-:-:-:-:-:-:-	0xFC400000
158		**Load/store register (immediat**		
159		STRB	-:-:-:-:-:-:-:-:-:-	0x38000400
160		LDRB	-:-:-:-:-:-:-:-:-:-	0x38400400
161		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38800400
162		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38C00400
163		STR	-:-:-:-:-:-:-:-:-:-	0x3C000400
164		LDR	-:-:-:-:-:-:-:-:-:-	0x3C400400
165		STR	-:-:-:-:-:-:-:-:-:-	0x3C800400
166		LDR	-:-:-:-:-:-:-:-:-:-	0x3CC00400
167		STRH	-:-:-:-:-:-:-:-:-:-	0x78000400
168		LDRH	-:-:-:-:-:-:-:-:-:-	0x78400400
169		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78800400
170		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78C00400
171		STR	-:-:-:-:-:-:-:-:-:-	0x7C000400
172		LDR	-:-:-:-:-:-:-:-:-:-	0x7C400400
173		STR	-:-:-:-:-:-:-:-:-:-	0xB8000400
174		LDR	-:-:-:-:-:-:-:-:-:-	0xB8400400
175		LDRSW	-:-:-:-:-:-:-:-:-:-	0xB8800400
176		STR	-:-:-:-:-:-:-:-:-:-	0xBC000400
177		LDR	-:-:-:-:-:-:-:-:-:-	0xBC400400
178		STR	-:-:-:-:-:-:-:-:-:-	0xF8000400
179		LDR	-:-:-:-:-:-:-:-:-:-	0xF8400400
180		STR	-:-:-:-:-:-:-:-:-:-	0xFC000400
181		LDR	-:-:-:-:-:-:-:-:-:-	0xFC400400
182		**Load/store register (unprivile**		
183		STTRB	-:-:-:-:-:-:-:-:-:-	0x38000800
184		LDTRB	-:-:-:-:-:-:-:-:-:-	0x38400800
185		LDTRSB	-:-:-:-:-:-:-:-:-:-	0x38800800

1	in_use	Opcode	31:30:29:28	Binary
186		LDTRSB	-:-:-:-:-:-:-:-:-:-	0x38C00800
187		STTRH	-:-:-:-:-:-:-:-:-:-	0x78000800
188		LDTRH	-:-:-:-:-:-:-:-:-:-	0x78400800
189		LDTRSH	-:-:-:-:-:-:-:-:-:-	0x78800800
190		LDTRSH	-:-:-:-:-:-:-:-:-:-	0x78C00800
191		STTR	-:-:-:-:-:-:-:-:-:-	0xB8000800
192		LDTR	-:-:-:-:-:-:-:-:-:-	0xB8400800
193		LDTRSW	-:-:-:-:-:-:-:-:-:-	0xB8800800
194		STTR	-:-:-:-:-:-:-:-:-:-	0xF8000800
195		LDTR	-:-:-:-:-:-:-:-:-:-	0xF8400800
196		**Load/store register (immediat**		
197		STRB	-:-:-:-:-:-:-:-:-:-	0x38000C00
198		LDRB	-:-:-:-:-:-:-:-:-:-	0x38400C00
199		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38800C00
200		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38C00C00
201		STR	-:-:-:-:-:-:-:-:-:-	0x3C000C00
202		LDR	-:-:-:-:-:-:-:-:-:-	0x3C400C00
203		STR	-:-:-:-:-:-:-:-:-:-	0x3C800C00
204		LDR	-:-:-:-:-:-:-:-:-:-	0x3CC00C00
205		STRH	-:-:-:-:-:-:-:-:-:-	0x78000C00
206		LDRH	-:-:-:-:-:-:-:-:-:-	0x78400C00
207		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78800C00
208		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78C00C00
209		STR	-:-:-:-:-:-:-:-:-:-	0x7C000C00
210		LDR	-:-:-:-:-:-:-:-:-:-	0x7C400C00
211		STR	-:-:-:-:-:-:-:-:-:-	0xB8000C00
212		LDR	-:-:-:-:-:-:-:-:-:-	0xB8400C00
213		LDRSW	-:-:-:-:-:-:-:-:-:-	0xB8800C00
214		STR	-:-:-:-:-:-:-:-:-:-	0xBC000C00
215		LDR	-:-:-:-:-:-:-:-:-:-	0xBC400C00
216		STR	-:-:-:-:-:-:-:-:-:-	0xF8000C00
217		LDR	-:-:-:-:-:-:-:-:-:-	0xF8400C00
218		STR	-:-:-:-:-:-:-:-:-:-	0xFC000C00
219		LDR	-:-:-:-:-:-:-:-:-:-	0xFC400C00
220		**Load/store register (register c**		
221		STRB	-:-:-:-:-:-:-:-:-:-	0x38200800
222		LDRB	-:-:-:-:-:-:-:-:-:-	0x38600800
223		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38A00800

1	in_use	Opcode	31:30:29:28	Binary
224		LDRSB	-:-:-:-:-:-:-:-:-:-	0x38E00800
225		STR	-:-:-:-:-:-:-:-:-:-	0x3C200800
226		LDR	-:-:-:-:-:-:-:-:-:-	0x3C600800
227		STR	-:-:-:-:-:-:-:-:-:-	0x3CA00800
228		LDR	-:-:-:-:-:-:-:-:-:-	0x3CE00800
229		STRH	-:-:-:-:-:-:-:-:-:-	0x78200800
230		LDRH	-:-:-:-:-:-:-:-:-:-	0x78600800
231		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78A00800
232		LDRSH	-:-:-:-:-:-:-:-:-:-	0x78E00800
233		STR	-:-:-:-:-:-:-:-:-:-	0x7C200800
234		LDR	-:-:-:-:-:-:-:-:-:-	0x7C600800
235		STR	-:-:-:-:-:-:-:-:-:-	0xB8200800
236		LDR	-:-:-:-:-:-:-:-:-:-	0xB8600800
237		LDRSW	-:-:-:-:-:-:-:-:-:-	0xB8A00800
238		STR	-:-:-:-:-:-:-:-:-:-	0xBC200800
239		LDR	-:-:-:-:-:-:-:-:-:-	0xBC600800
240		STR	-:-:-:-:-:-:-:-:-:-	0xF8200800
241		LDR	-:-:-:-:-:-:-:-:-:-	0xF8600800
243		STR	-:-:-:-:-:-:-:-:-:-	0xFC200800
244		LDR	-:-:-:-:-:-:-:-:-:-	0xFC600800
242		PRFM	-:-:-:-:-:-:-:-:-:-	0xF8A00800
245		**Load/store register (unsigned**		
246		STRB	-:-:-:-:-:-:-:-:-:i	0x39000000
247		LDRB	-:-:-:-:-:-:-:-:-:i	0x39400000
248		LDRSB	-:-:-:-:-:-:-:-:-:i	0x39800000
249		LDRSB	-:-:-:-:-:-:-:-:-:i	0x39C00000
250		STR	-:-:-:-:-:-:-:-:-:i	0x3D000000
251		LDR	-:-:-:-:-:-:-:-:-:i	0x3D400000
252		STR	-:-:-:-:-:-:-:-:-:i	0x3D800000
253		LDR	-:-:-:-:-:-:-:-:-:i	0x3DC00000
254		STRH	-:-:-:-:-:-:-:-:-:i	0x79000000
255		LDRH	-:-:-:-:-:-:-:-:-:i	0x79400000
256		LDRSH	-:-:-:-:-:-:-:-:-:i	0x79800000
257		LDRSH	-:-:-:-:-:-:-:-:-:i	0x79C00000
258		STR	-:-:-:-:-:-:-:-:-:i	0x7D000000
259		LDR	-:-:-:-:-:-:-:-:-:i	0x7D400000
260		STR	-:-:-:-:-:-:-:-:-:i	0xB9000000
261		LDR	-:-:-:-:-:-:-:-:-:i	0xB9400000

1	in_use	Opcode	31:30:29:28	Binary
262		LDRSW	-:-:-:-:-:-:-:-:-:i	0xB9800000
263		STR	-:-:-:-:-:-:-:-:-:i	0xBD000000
264		LDR	-:-:-:-:-:-:-:-:-:i	0xBD400000
265		STR	-:-:-:-:-:-:-:-:-:i	0xF9000000
266		LDR	-:-:-:-:-:-:-:-:-:i	0xF9400000
268		STR	-:-:-:-:-:-:-:-:-:i	0xFD000000
269		LDR	-:-:-:-:-:-:-:-:-:i	0xFD400000
267		PRFM	-:-:-:-:-:-:-:-:-:i	0xF9800000
270	**Data processing – Immedia**			
271	**PC-rel. addressing**			
272		ADR	-:immlo::-:-:-:-:	0x10000000
273		ADRP	-:immlo::-:-:-:-:	0x90000000
274	**Add/subtract (immediate)**			
275		ADD	-:-:-:-:-:-:-:-:-:i	0x11000000
276		ADDS	-:-:-:-:-:-:-:-:-:i	0x31000000
277		SUB	-:-:-:-:-:-:-:-:-:i	0x51000000
278		SUBS	-:-:-:-:-:-:-:-:-:i	0x71000000
279		ADD	-:-:-:-:-:-:-:-:-:i	0x91000000
280		ADDS	-:-:-:-:-:-:-:-:-:i	0xB1000000
281		SUB	-:-:-:-:-:-:-:-:-:i	0xD1000000
282		SUBS	-:-:-:-:-:-:-:-:-:i	0xF1000000
283	**Logical (immediate)**			
284		AND	-:-:-:-:-:-:-:-:-:i	0x12000000
285		ORR	-:-:-:-:-:-:-:-:-:i	0x32000000
286		EOR	-:-:-:-:-:-:-:-:-:i	0x52000000
287		ANDS	-:-:-:-:-:-:-:-:-:i	0x72000000
288		AND	-:-:-:-:-:-:-:-:-:i	0x92000000
289		ORR	-:-:-:-:-:-:-:-:-:i	0xB2000000
290		EOR	-:-:-:-:-:-:-:-:-:i	0xD2000000
291		ANDS	-:-:-:-:-:-:-:-:-:i	0xF2000000
292	**Move wide (immediate)**			
293		MOVN	-:-:-:-:-:-:-:-:-:	0x12800000
294		MOVZ	-:-:-:-:-:-:-:-:-:	0x52800000
295		MOVK	-:-:-:-:-:-:-:-:-:	0x72800000
296		MOVN	-:-:-:-:-:-:-:-:-:	0x92800000
297		MOVZ	-:-:-:-:-:-:-:-:-:	0xD2800000
298		MOVK	-:-:-:-:-:-:-:-:-:	0xF2800000
299	**Bitfield**			

1	in_use	Opcode	31:30:29:28	Binary
300		SBFM	-:-:-:-:-:-:-:-:-:i	0x13000000
301		BFM	-:-:-:-:-:-:-:-:-:i	0x33000000
302		UBFM	-:-:-:-:-:-:-:-:-:i	0x53000000
303		SBFM	-:-:-:-:-:-:-:-:-:i	0x93400000
304		BFM	-:-:-:-:-:-:-:-:-:i	0xB3400000
305		UBFM	-:-:-:-:-:-:-:-:-:i	0xD3400000
306		**Extract**		
307		EXTR	-:-:-:-:-:-:-:-:-:-	0x13800000
308		EXTR	-:-:-:-:-:-:-:-:-:-	0x93C08000
309		**Data Processing – register**		
310		**Logical (shifted register)**		
311		AND	-:-:-:-:-:-:-:-:shif	0x0A000000
312		BIC	-:-:-:-:-:-:-:-:shif	0x0A200000
313		ORR	-:-:-:-:-:-:-:-:shif	0x2A000000
314		ORN	-:-:-:-:-:-:-:-:shif	0x2A200000
315		EOR	-:-:-:-:-:-:-:-:shif	0x4A000000
316		EON	-:-:-:-:-:-:-:-:shif	0x4A200000
317		ANDS	-:-:-:-:-:-:-:-:shif	0x6A000000
318		BICS	-:-:-:-:-:-:-:-:shif	0x6A200000
319		AND	-:-:-:-:-:-:-:-:shif	0x8A000000
320		BIC	-:-:-:-:-:-:-:-:shif	0x8A200000
321		ORR	-:-:-:-:-:-:-:-:shif	0xAA000000
322		ORN	-:-:-:-:-:-:-:-:shif	0xAA200000
323		EOR	-:-:-:-:-:-:-:-:shif	0xCA000000
324		EON	-:-:-:-:-:-:-:-:shif	0xCA200000
325		ANDS	-:-:-:-:-:-:-:-:shif	0xEA000000
326		BICS	-:-:-:-:-:-:-:-:shif	0xEA200000
327		**Add/subtract (shifted register**		
328		ADD	-:-:-:-:-:-:-:-:-:-	0x0B000000
329		ADDS	-:-:-:-:-:-:-:-:-:-	0x2B000000
330		SUB	-:-:-:-:-:-:-:-:-:-	0x4B000000
331		SUBS	-:-:-:-:-:-:-:-:-:-	0x6B000000
332		ADD	-:-:-:-:-:-:-:-:-:-	0x8B000000
333		ADDS	-:-:-:-:-:-:-:-:-:-	0xAB000000
334		SUB	-:-:-:-:-:-:-:-:-:-	0xCB000000
335		SUBS	-:-:-:-:-:-:-:-:-:-	0xEB000000
336		**Add/subtract (extended regist**		
337		ADD	-:-:-:-:-:-:-:-:-:-	0x0B200000

1	in_use	Opcode	31:30:29:28	Binary
338		ADDS	-:-:-:-:-:-:-:-:-:-	0x2B200000
339		SUB	-:-:-:-:-:-:-:-:-:-	0x4B200000
340		SUBS	-:-:-:-:-:-:-:-:-:-	0x6B200000
341		ADD	-:-:-:-:-:-:-:-:-:-	0x8B200000
342		ADDS	-:-:-:-:-:-:-:-:-:-	0xAB200000
343		SUB	-:-:-:-:-:-:-:-:-:-	0xCB200000
344		SUBS	-:-:-:-:-:-:-:-:-:-	0xEB200000
345		**Add/subtract (with carry)**		
346		ADC	-:-:-:-:-:-:-:-:-:-	0x1A000000
347		ADCS	-:-:-:-:-:-:-:-:-:-	0x3A000000
348		SBC	-:-:-:-:-:-:-:-:-:-	0x5A000000
349		SBCS	-:-:-:-:-:-:-:-:-:-	0x7A000000
350		ADC	-:-:-:-:-:-:-:-:-:-	0x9A000000
351		ADCS	-:-:-:-:-:-:-:-:-:-	0xBA000000
352		SBC	-:-:-:-:-:-:-:-:-:-	0xDA000000
353		SBCS	-:-:-:-:-:-:-:-:-:-	0xFA000000
354		**Conditional compare (registe**		
355		CCMN	-:-:-:-:-:-:-:-:-:-	0x3A400000
356		CCMN	-:-:-:-:-:-:-:-:-:-	0xBA400000
357		CCMP	-:-:-:-:-:-:-:-:-:-	0x7A400000
358		CCMP	-:-:-:-:-:-:-:-:-:-	0xFA400000
359		**Conditional compare (immedi**		
360		CCMN	-:-:-:-:-:-:-:-:-:-	0x3A400800
361		CCMN	-:-:-:-:-:-:-:-:-:-	0xBA400800
362		CCMP	-:-:-:-:-:-:-:-:-:-	0x7A400800
363		CCMP	-:-:-:-:-:-:-:-:-:-	0xFA400800
364		**Conditional select**		
365		CSEL	-:-:-:-:-:-:-:-:-:-	0x1A800000
366		CSINC	-:-:-:-:-:-:-:-:-:-	0x1A800400
367		CSINV	-:-:-:-:-:-:-:-:-:-	0x5A800000
368		CSNEG	-:-:-:-:-:-:-:-:-:-	0x5A800400
369		CSEL	-:-:-:-:-:-:-:-:-:-	0x9A800000
370		CSINC	-:-:-:-:-:-:-:-:-:-	0x9A800400
371		CSINV	-:-:-:-:-:-:-:-:-:-	0xDA800000
372		CSNEG	-:-:-:-:-:-:-:-:-:-	0xDA800400
373		**Data-processing (3 source)**		
374		MADD	-:-:-:-:-:-:-:-:-:-	0x1B000000
375		MADD	-:-:-:-:-:-:-:-:-:-	0x9B000000

1	in_use	Opcode	31:30:29:28	Binary
376		SMADDL	-:-:-:-:-:-:-:-:-:-	0x9B200000
377		UMADDL	-:-:-:-:-:-:-:-:-:-	0x9BA00000
378		MSUB	-:-:-:-:-:-:-:-:-:-	0x1B008000
379		MSUB	-:-:-:-:-:-:-:-:-:-	0x9B008000
380		SMSUBL	-:-:-:-:-:-:-:-:-:-	0x9B208000
381		UMSUBL	-:-:-:-:-:-:-:-:-:-	0x9BA08000
382		SMULH	-:-:-:-:-:-:-:-:-:-	0x9B400000
383		UMULH	-:-:-:-:-:-:-:-:-:-	0x9BC00000
384		**Data-processing (2 source)**		
385		CRC32X	-:-:-:-:-:-:-:-:-:-	0x9AC04C00
386		CRC32CX	-:-:-:-:-:-:-:-:-:-	0x9AC05C00
387		CRC32B	-:-:-:-:-:-:-:-:-:-	0x1AC04000
388		CRC32CB	-:-:-:-:-:-:-:-:-:-	0x1AC05000
389		CRC32H	-:-:-:-:-:-:-:-:-:-	0x1AC04400
390		CRC32CH	-:-:-:-:-:-:-:-:-:-	0x1AC05400
391		CRC32W	-:-:-:-:-:-:-:-:-:-	0x1AC04800
392		CRC32CW	-:-:-:-:-:-:-:-:-:-	0x1AC05800
393		UDIV	-:-:-:-:-:-:-:-:-:-	0x1AC00800
394		UDIV	-:-:-:-:-:-:-:-:-:-	0x9AC00800
395		SDIV	-:-:-:-:-:-:-:-:-:-	0x1AC00C00
396		SDIV	-:-:-:-:-:-:-:-:-:-	0x9AC00C00
397		LSLV	-:-:-:-:-:-:-:-:-:-	0x1AC02000
398		LSLV	-:-:-:-:-:-:-:-:-:-	0x9AC02000
399		LSRV	-:-:-:-:-:-:-:-:-:-	0x1AC02400
400		LSRV	-:-:-:-:-:-:-:-:-:-	0x9AC02400
401		ASRV	-:-:-:-:-:-:-:-:-:-	0x1AC02800
402		ASRV	-:-:-:-:-:-:-:-:-:-	0x9AC02800
403		RORV	-:-:-:-:-:-:-:-:-:-	0x1AC02C00
404		RORV	-:-:-:-:-:-:-:-:-:-	0x9AC02C00
405		**Data-processing (1 source)**		
406		RBIT	-:-:-:-:-:-:-:-:-:-	0x5AC00000
407		RBIT	-:-:-:-:-:-:-:-:-:-	0xDAC00000
408		CLZ	-:-:-:-:-:-:-:-:-:-	0x5AC01000
409		CLZ	-:-:-:-:-:-:-:-:-:-	0xDAC01000
410		CLS	-:-:-:-:-:-:-:-:-:-	0x5AC01400
411		CLS	-:-:-:-:-:-:-:-:-:-	0xDAC01400
412		REV	-:-:-:-:-:-:-:-:-:-	0x5AC00800
413		REV	-:-:-:-:-:-:-:-:-:-	0xDAC00C00

1	in_use	Opcode	31:30:29:28	Binary
414		REV16	-:-:-:-:-:-:-:-:-:-	0xDAC00400
415		REV16	-:-:-:-:-:-:-:-:-:-	0x5AC00400
416		REV32	-:-:-:-:-:-:-:-:-:-	0xDAC00800
417	//	**Data Processing – SIMD an**		
418	//	**Floating-point<->fixed-point c**		
419	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x1E020000
420	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x1E030000
421	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x1ED80000
422	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x1ED90000
423	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x1E020000
424	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x1E030000
425	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x1ED80000
426	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x1ED90000
427	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x9E020000
428	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x9E030000
429	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x9ED80000
430	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x9ED90000
431	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x9E020000
432	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x9E030000
433	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x9ED80000
434	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x9ED90000
435	//	**Floating-point conditional cor**		
436	//	FCCMP	-:-:-:-:-:-:-:-:-:-	0x1E200400
437	//	FCCMPE	-:-:-:-:-:-:-:-:-:-	0x1E200410
438	//	FCCMP	-:-:-:-:-:-:-:-:-:-	0x1E600400
439	//	FCCMPE	-:-:-:-:-:-:-:-:-:-	0x1E600410
440	//	**Floating-point data-processin**		
441	//	FMUL	-:-:-:-:-:-:-:-:-:-	0x1E200800
442	//	FDIV	-:-:-:-:-:-:-:-:-:-	0x1E201800
443	//	FADD	-:-:-:-:-:-:-:-:-:-	0x1E202800
444	//	FSUB	-:-:-:-:-:-:-:-:-:-	0x1E203800
445	//	FMAX	-:-:-:-:-:-:-:-:-:-	0x1E204800
446	//	FMIN	-:-:-:-:-:-:-:-:-:-	0x1E205800
447	//	FMAXNM	-:-:-:-:-:-:-:-:-:-	0x1E206800

1	in_use	Opcode	31:30:29:28	Binary
448	//	FMINNM	-:-:-:-:-:-:-:-:-:	0x1E207800
449	//	FNMUL	-:-:-:-:-:-:-:-:-:	0x1E208800
450	//	FMUL	-:-:-:-:-:-:-:-:-:	0x1E600800
451	//	FDIV	-:-:-:-:-:-:-:-:-:	0x1E601800
452	//	FADD	-:-:-:-:-:-:-:-:-:	0x1E602800
453	//	FSUB	-:-:-:-:-:-:-:-:-:	0x1E603800
454	//	FMAX	-:-:-:-:-:-:-:-:-:	0x1E604800
455	//	FMIN	-:-:-:-:-:-:-:-:-:	0x1E605800
456	//	FMAXNM	-:-:-:-:-:-:-:-:-:	0x1E606800
457	//	FMINNM	-:-:-:-:-:-:-:-:-:	0x1E607800
458	//	FNMUL	-:-:-:-:-:-:-:-:-:	0x1E608800
459	//	**Floating-point conditional sel**		
460	//	FCSEL	-:-:-:-:-:-:-:-:-:	0x1E200C00
461	//	FCSEL	-:-:-:-:-:-:-:-:-:	0x1E600C00
462	//	**Floating-point immediate**		
463	//	FMOV	-:-:-:-:-:-:-:-:-:	0x1E201000
464	//	FMOV	-:-:-:-:-:-:-:-:-:	0x1E601000
465	//	**Floating-point compare**		
466	//	FCMP	-:-:-:-:-:-:-:-:-:	0x1E202000
467	//	FCMP	-:-:-:-:-:-:-:-:-:	0x1E202008
468	//	FCMPE	-:-:-:-:-:-:-:-:-:	0x1E202010
469	//	FCMPE	-:-:-:-:-:-:-:-:-:	0x1E202018
470	//	FCMP	-:-:-:-:-:-:-:-:-:	0x1E602000
471	//	FCMP	-:-:-:-:-:-:-:-:-:	0x1E602008
472	//	FCMPE	-:-:-:-:-:-:-:-:-:	0x1E602010
473	//	FCMPE	-:-:-:-:-:-:-:-:-:	0x1E602018
474	//	**Floating-point data-processin**		
475	//	FMOV	-:-:-:-:-:-:-:-:-:	0x1E204000
476	//	FABS	-:-:-:-:-:-:-:-:-:	0x1E20C000
477	//	FNEG	-:-:-:-:-:-:-:-:-:	0x1E214000
478	//	FSQRT	-:-:-:-:-:-:-:-:-:	0x1E21C000
479	//	FCVT	-:-:-:-:-:-:-:-:-:	0x1E22C000
480	//	FCVT	-:-:-:-:-:-:-:-:-:	0x1E23C000
481	//	FRINTN	-:-:-:-:-:-:-:-:-:	0x1E244000

1	in_use	Opcode	31:30:29:28	Binary
482	//	FRINTP	-:-:-:-:-:-:-:-:-:-	0x1E24C000
483	//	FRINTM	-:-:-:-:-:-:-:-:-:-	0x1E254000
484	//	FRINTZ	-:-:-:-:-:-:-:-:-:-	0x1E25C000
485	//	FRINTA	-:-:-:-:-:-:-:-:-:-	0x1E264000
486	//	FRINTX	-:-:-:-:-:-:-:-:-:-	0x1E274000
487	//	FRINTI	-:-:-:-:-:-:-:-:-:-	0x1E27C000
488	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x1E604000
489	//	FABS	-:-:-:-:-:-:-:-:-:-	0x1E60C000
490	//	FNEG	-:-:-:-:-:-:-:-:-:-	0x1E614000
491	//	FSQRT	-:-:-:-:-:-:-:-:-:-	0x1E61C000
492	//	FCVT	-:-:-:-:-:-:-:-:-:-	0x1E624000
493	//	FCVT	-:-:-:-:-:-:-:-:-:-	0x1E63C000
494	//	FRINTN	-:-:-:-:-:-:-:-:-:-	0x1E644000
495	//	FRINTP	-:-:-:-:-:-:-:-:-:-	0x1E64C000
496	//	FRINTM	-:-:-:-:-:-:-:-:-:-	0x1E654000
497	//	FRINTZ	-:-:-:-:-:-:-:-:-:-	0x1E65C000
498	//	FRINTA	-:-:-:-:-:-:-:-:-:-	0x1E664000
499	//	FRINTX	-:-:-:-:-:-:-:-:-:-	0x1E674000
500	//	FRINTI	-:-:-:-:-:-:-:-:-:-	0x1E67C000
501	//	FCVT	-:-:-:-:-:-:-:-:-:-	0x1EE24000
502	//	FCVT	-:-:-:-:-:-:-:-:-:-	0x1EE2C000
503	//	**Floating-point<->integer conv**		
504	//	FCVTNS	-:-:-:-:-:-:-:-:-:-	0x1E200000
505	//	FCVTNU	-:-:-:-:-:-:-:-:-:-	0x1E210000
506	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x1E220000
507	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x1E230000
508	//	FCVTAS	-:-:-:-:-:-:-:-:-:-	0x1E240000
509	//	FCVTAU	-:-:-:-:-:-:-:-:-:-	0x1E250000
510	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x1E260000
511	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x1E270000
512	//	FCVTPS	-:-:-:-:-:-:-:-:-:-	0x1E280000
513	//	FCVTPU	-:-:-:-:-:-:-:-:-:-	0x1E290000
514	//	FCVTMS	-:-:-:-:-:-:-:-:-:-	0x1E300000
515	//	FCVTMU	-:-:-:-:-:-:-:-:-:-	0x1E310000

1	in_use	Opcode	31:30:29:28	Binary
516	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x1E380000
517	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x1E390000
518	//	FCVTNS	-:-:-:-:-:-:-:-:-:-	0x1E600000
519	//	FCVTNU	-:-:-:-:-:-:-:-:-:-	0x1E610000
520	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x1E620000
521	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x1E630000
522	//	FCVTAS	-:-:-:-:-:-:-:-:-:-	0x1E640000
523	//	FCVTAU	-:-:-:-:-:-:-:-:-:-	0x1E650000
524	//	FCVTPS	-:-:-:-:-:-:-:-:-:-	0x1E680000
525	//	FCVTPU	-:-:-:-:-:-:-:-:-:-	0x1E690000
526	//	FCVTMS	-:-:-:-:-:-:-:-:-:-	0x1E700000
527	//	FCVTMU	-:-:-:-:-:-:-:-:-:-	0x1E710000
528	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x1E780000
529	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x1E790000
530	//	FCVTNS	-:-:-:-:-:-:-:-:-:-	0x9E200000
531	//	FCVTNU	-:-:-:-:-:-:-:-:-:-	0x9E210000
532	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x9E220000
533	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x9E230000
534	//	FCVTAS	-:-:-:-:-:-:-:-:-:-	0x9E240000
535	//	FCVTAU	-:-:-:-:-:-:-:-:-:-	0x9E250000
536	//	FCVTPS	-:-:-:-:-:-:-:-:-:-	0x9E280000
537	//	FCVTPU	-:-:-:-:-:-:-:-:-:-	0x9E290000
538	//	FCVTMS	-:-:-:-:-:-:-:-:-:-	0x9E300000
539	//	FCVTMU	-:-:-:-:-:-:-:-:-:-	0x9E310000
540	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x9E380000
541	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x9E390000
542	//	FCVTNS	-:-:-:-:-:-:-:-:-:-	0x9E600000
543	//	FCVTNU	-:-:-:-:-:-:-:-:-:-	0x9E610000
544	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x9E620000
545	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x9E630000
546	//	FCVTAS	-:-:-:-:-:-:-:-:-:-	0x9E640000
547	//	FCVTAU	-:-:-:-:-:-:-:-:-:-	0x9E650000
548	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x9E660000
549	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x9E670000

1	in_use	Opcode	31:30:29:28	Binary
550	//	FCVTPS	-:-:-:-:-:-:-:-:-:-	0x9E680000
551	//	FCVTPU	-:-:-:-:-:-:-:-:-:-	0x9E690000
552	//	FCVTMS	-:-:-:-:-:-:-:-:-:-	0x9E700000
553	//	FCVTMU	-:-:-:-:-:-:-:-:-:-	0x9E710000
554	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x9E780000
555	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x9E790000
556	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x9EAE0000
557	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x9EAF0000
558	//	**Floating-point data-processin**		
559	//	FMADD	-:-:-:-:-:-:-:-:-:-	0x1F000000
560	//	FMSUB	-:-:-:-:-:-:-:-:-:-	0x1F008000
561	//	FNMADD	-:-:-:-:-:-:-:-:-:-	0x1F200000
562	//	FNMSUB	-:-:-:-:-:-:-:-:-:-	0x1F208000
563	//	FMADD	-:-:-:-:-:-:-:-:-:-	0x1F400000
564	//	FMSUB	-:-:-:-:-:-:-:-:-:-	0x1F408000
565	//	FNMADD	-:-:-:-:-:-:-:-:-:-	0x1F600000
566	//	FNMSUB	-:-:-:-:-:-:-:-:-:-	0x1F608000
567	//	**AdvSIMD scalar three same**		
568	//	SQADD	-:-:-:-:-:-:-:-:-:-	0x5E200C00
569	//	SQSUB	-:-:-:-:-:-:-:-:-:-	0x5E202C00
570	//	CMGT	-:-:-:-:-:-:-:-:-:-	0x5E203400
571	//	CMGE	-:-:-:-:-:-:-:-:-:-	0x5E203C00
572	//	SSHL	-:-:-:-:-:-:-:-:-:-	0x5E204400
573	//	SQSHL	-:-:-:-:-:-:-:-:-:-	0x5E204C00
574	//	SRSHL	-:-:-:-:-:-:-:-:-:-	0x5E205400
575	//	SQRSHL	-:-:-:-:-:-:-:-:-:-	0x5E205C00
576	//	ADD	-:-:-:-:-:-:-:-:-:-	0x5E208400
577	//	CMTST	-:-:-:-:-:-:-:-:-:-	0x5E208C00
578	//	SQDMULH	-:-:-:-:-:-:-:-:-:-	0x5E20B400
579	//	FMULX	-:-:-:-:-:-:-:-:-:-	0x5E20DC00
580	//	FCMEQ	-:-:-:-:-:-:-:-:-:-	0x5E20E400
581	//	FRECPS	-:-:-:-:-:-:-:-:-:-	0x5E20FC00
582	//	FRSQRTS	-:-:-:-:-:-:-:-:-:-	0x5EA0FC00
583	//	UQADD	-:-:-:-:-:-:-:-:-:-	0x7E200C00

1	in_use	Opcode	31:30:29:28	Binary
584	//	UQSUB	-:-:-:-:-:-:-:-:-:-	0x7E202C00
585	//	CMHI	-:-:-:-:-:-:-:-:-:-	0x7E203400
586	//	CMHS	-:-:-:-:-:-:-:-:-:-	0x7E203C00
587	//	USHL	-:-:-:-:-:-:-:-:-:-	0x7E204400
588	//	UQSHL	-:-:-:-:-:-:-:-:-:-	0x7E204C00
589	//	URSHL	-:-:-:-:-:-:-:-:-:-	0x7E205400
590	//	UQRSHL	-:-:-:-:-:-:-:-:-:-	0x7E205C00
591	//	SUB	-:-:-:-:-:-:-:-:-:-	0x7E208400
592	//	CMEQ	-:-:-:-:-:-:-:-:-:-	0x7E208C00
593	//	SQRDMULH	-:-:-:-:-:-:-:-:-:-	0x7E20B400
594	//	FCMGE	-:-:-:-:-:-:-:-:-:-	0x7E20E400
595	//	FACGE	-:-:-:-:-:-:-:-:-:-	0x7E20EC00
596	//	FABD	-:-:-:-:-:-:-:-:-:-	0x7EA0D400
597	//	FCMGT	-:-:-:-:-:-:-:-:-:-	0x7EA0E400
598	//	FACGT	-:-:-:-:-:-:-:-:-:-	0x7EA0EC00
599	//	**AdvSIMD scalar three differer**		
600	//	SQDMLAL	-:-:-:-:-:-:-:-:size	0x5E209000
601	//	SQDMLAL2	-:-:-:-:-:-:-:-:size	0x5E209000
602	//	SQDMLSL	-:-:-:-:-:-:-:-:size	0x5E20B000
603	//	SQDMLSL2	-:-:-:-:-:-:-:-:size	0x5E20B000
604	//	SQDMULL	-:-:-:-:-:-:-:-:size	0x5E20D000
605	//	SQDMULL2	-:-:-:-:-:-:-:-:size	0x5E20D000
606	//	**AdvSIMD scalar two-reg misc**		
607	//	SUQADD	-:-:-:-:-:-:-:-:-:-	0x5E203800
608	//	SQABS	-:-:-:-:-:-:-:-:-:-	0x5E207800
609	//	CMGT	-:-:-:-:-:-:-:-:-:-	0x5E208800
610	//	CMEQ	-:-:-:-:-:-:-:-:-:-	0x5E209800
611	//	CMLT	-:-:-:-:-:-:-:-:-:-	0x5E20A800
612	//	ABS	-:-:-:-:-:-:-:-:-:-	0x5E20B800
613	//	SQXTN	-:-:-:-:-:-:-:-:-:-	0x5E214800
614	//	SQXTN2	-:-:-:-:-:-:-:-:-:-	0x5E214800
615	//	FCVTNS	-:-:-:-:-:-:-:-:-:-	0x5E21A800
616	//	FCVTMS	-:-:-:-:-:-:-:-:-:-	0x5E21B800
617	//	FCVTAS	-:-:-:-:-:-:-:-:-:-	0x5E21C800

1	in_use	Opcode	31:30:29:28	Binary
618	//	SCVTF	-:-:-:-:-:-:-:-:-:-	0x5E21D800
619	//	FCMGT	-:-:-:-:-:-:-:-:-:-	0x5EA0C800
620	//	FCMEQ	-:-:-:-:-:-:-:-:-:-	0x5EA0D800
621	//	FCMLT	-:-:-:-:-:-:-:-:-:-	0x5EA0E800
622	//	FCVTPS	-:-:-:-:-:-:-:-:-:-	0x5EA1A800
623	//	FCVTZS	-:-:-:-:-:-:-:-:-:-	0x5EA1B800
624	//	FRECPE	-:-:-:-:-:-:-:-:-:-	0x5EA1D800
625	//	FRECPX	-:-:-:-:-:-:-:-:-:-	0x5EA1F800
626	//	USQADD	-:-:-:-:-:-:-:-:-:-	0x7E203800
627	//	SQNEG	-:-:-:-:-:-:-:-:-:-	0x7E207800
628	//	CMGE	-:-:-:-:-:-:-:-:-:-	0x7E208800
629	//	CMLE	-:-:-:-:-:-:-:-:-:-	0x7E209800
630	//	NEG	-:-:-:-:-:-:-:-:-:-	0x7E20B800
631	//	SQXTUN	-:-:-:-:-:-:-:-:-:-	0x7E212800
632	//	SQXTUN2	-:-:-:-:-:-:-:-:-:-	0x7E212800
633	//	UQXTN	-:-:-:-:-:-:-:-:-:-	0x7E214800
634	//	UQXTN2	-:-:-:-:-:-:-:-:-:-	0x7E214800
635	//	FCVTXN	-:-:-:-:-:-:-:-:-:-	0x7E216800
636	//	FCVTXN2	-:-:-:-:-:-:-:-:-:-	0x7E216800
637	//	FCVTNU	-:-:-:-:-:-:-:-:-:-	0x7E21A800
638	//	FCVTMU	-:-:-:-:-:-:-:-:-:-	0x7E21B800
639	//	FCVTAU	-:-:-:-:-:-:-:-:-:-	0x7E21C800
640	//	UCVTF	-:-:-:-:-:-:-:-:-:-	0x7E21D800
641	//	FCMGE	-:-:-:-:-:-:-:-:-:-	0x7EA0C800
642	//	FCMLE	-:-:-:-:-:-:-:-:-:-	0x7EA0D800
643	//	FCVTPU	-:-:-:-:-:-:-:-:-:-	0x7EA1A800
644	//	FCVTZU	-:-:-:-:-:-:-:-:-:-	0x7EA1B800
645	//	FRSQRTE	-:-:-:-:-:-:-:-:-:-	0x7EA1D800
646	//	**AdvSIMD scalar pairwise**		
647	//	ADDP	-:-:-:-:-:-:-:-:-:-	0x5E31B800
648	//	FMAXNMP	-:-:-:-:-:-:-:-:-:-	0x7E30C800
649	//	FADDP	-:-:-:-:-:-:-:-:-:-	0x7E30D800
650	//	FMAXP	-:-:-:-:-:-:-:-:-:-	0x7E30F800
651	//	FMINNMP	-:-:-:-:-:-:-:-:-:-	0x7EB0C800

1	in_use	Opcode	31:30:29:28	Binary
652	//	FMINP	-:-:-:-:-:-:-:-:-:-	0x7EB0F800
653	//	**AdvSIMD scalar copy**		
654	//	DUP	-:-:-:-:-:-:-:-:-:-	0x5E000400
655	//	**AdvSIMD scalar x indexed ele**		
656	//	SQDMLAL	-:-:-:-:-:-:-:-:-:L	0x5F003000
657	//	SQDMLAL2	-:-:-:-:-:-:-:-:-:L	0x5F003000
658	//	SQDMLSL	-:-:-:-:-:-:-:-:-:L	0x5F007000
659	//	SQDMLSL2	-:-:-:-:-:-:-:-:-:L	0x5F007000
660	//	SQDMULL	-:-:-:-:-:-:-:-:-:L	0x5F00B000
661	//	SQDMULL2	-:-:-:-:-:-:-:-:-:L	0x5F00B000
662	//	SQDMULH	-:-:-:-:-:-:-:-:-:L	0x5F00C000
663	//	SQRDMULH	-:-:-:-:-:-:-:-:-:L	0x5F00D000
664	//	FMLA	-:-:-:-:-:-:-:-:-:L	0x5F801000
665	//	FMLS	-:-:-:-:-:-:-:-:-:L	0x5F805000
666	//	FMUL	-:-:-:-:-:-:-:-:-:L	0x5F809000
667	//	FMULX	-:-:-:-:-:-:-:-:-:L	0x7F809000
668	//	**AdvSIMD scalar shift by imme**		
669	//	SSHR	-:-:-:-:-:-:-:-:-:im	0x5F000400
670	//	SSRA	-:-:-:-:-:-:-:-:-:im	0x5F001400
671	//	SRSHR	-:-:-:-:-:-:-:-:-:im	0x5F002400
672	//	SRSRA	-:-:-:-:-:-:-:-:-:im	0x5F003400
673	//	SHL	-:-:-:-:-:-:-:-:-:im	0x5F005400
674	//	SQSHL	-:-:-:-:-:-:-:-:-:im	0x5F007400
675	//	SQSHRN	-:-:-:-:-:-:-:-:-:im	0x5F009400
676	//	SQSHRN2	-:-:-:-:-:-:-:-:-:im	0x5F009400
677	//	SQRSHRN	-:-:-:-:-:-:-:-:-:im	0x5F009C00
678	//	SQRSHRN2	-:-:-:-:-:-:-:-:-:im	0x5F009C00
679	//	SCVTF	-:-:-:-:-:-:-:-:-:im	0x5F00E400
680	//	FCVTZS	-:-:-:-:-:-:-:-:-:im	0x5F00FC00
681	//	USHR	-:-:-:-:-:-:-:-:-:im	0x7F000400
682	//	USRA	-:-:-:-:-:-:-:-:-:im	0x7F001400
683	//	URSHR	-:-:-:-:-:-:-:-:-:im	0x7F002400
684	//	URSRA	-:-:-:-:-:-:-:-:-:im	0x7F003400
685	//	SRI	-:-:-:-:-:-:-:-:-:im	0x7F004400

1	in_use	Opcode	31:30:29:28	Binary
686	//	SLI	-:-:-:-:-:-:-:-:im	0x7F005400
687	//	SQSHLU	-:-:-:-:-:-:-:-:im	0x7F006400
688	//	UQSHL	-:-:-:-:-:-:-:-:im	0x7F007400
689	//	SQSHRUN	-:-:-:-:-:-:-:-:im	0x7F008400
690	//	SQSHRUN2	-:-:-:-:-:-:-:-:im	0x7F008400
691	//	SQRSHRUN	-:-:-:-:-:-:-:-:im	0x7F008C00
692	//	SQRSHRUN2	-:-:-:-:-:-:-:-:im	0x7F008C00
693	//	UQSHRN	-:-:-:-:-:-:-:-:im	0x7F009400
694	//	UQRSHRN	-:-:-:-:-:-:-:-:im	0x7F009C00
695	//	UQRSHRN2	-:-:-:-:-:-:-:-:im	0x7F009C00
696	//	UCVTF	-:-:-:-:-:-:-:-:im	0x7F00E400
697	//	FCVTZU	-:-:-:-:-:-:-:-:im	0x7F00FC00
698	//	**Crypto three-reg SHA**		
699	//	SHA1C	-:-:-:-:-:-:-:-:-:	0x5E000000
700	//	SHA1P	-:-:-:-:-:-:-:-:-:	0x5E001000
701	//	SHA1M	-:-:-:-:-:-:-:-:-:	0x5E002000
702	//	SHA1SU0	-:-:-:-:-:-:-:-:-:	0x5E003000
703	//	SHA256H	-:-:-:-:-:-:-:-:-:	0x5E004000
704	//	SHA256H2	-:-:-:-:-:-:-:-:-:	0x5E005000
705	//	SHA256SU1	-:-:-:-:-:-:-:-:-:	0x5E006000
706	//	**Crypto two-reg SHA**		
707	//	SHA1H	-:-:-:-:-:-:-:-:-:	0x5E280800
708	//	SHA1SU1	-:-:-:-:-:-:-:-:-:	0x5E281800
709	//	SHA256SU0	-:-:-:-:-:-:-:-:-:	0x5E282800
710	//	**Crypto AES**		
711	//	AESE	-:-:-:-:-:-:-:-:-:	0x4E284800
712	//	AESD	-:-:-:-:-:-:-:-:-:	0x4E285800
713	//	AESMC	-:-:-:-:-:-:-:-:-:	0x4E286800
714	//	AESIMC	-:-:-:-:-:-:-:-:-:	0x4E287800
715	//	**AdvSIMD three same**		
716	//	SHADD	-:Q:-:-:-:-:-:-:-	0x0E200400
717	//	SQADD	-:Q:-:-:-:-:-:-:-	0x0E200C00
718	//	SRHADD	-:Q:-:-:-:-:-:-:-	0x0E201400
719	//	SHSUB	-:Q:-:-:-:-:-:-:-	0x0E202400

1	in_use	Opcode	31:30:29:28	Binary
720	//	SQSUB	-:Q:-:-:-:-:-:-:-	0x0E202C00
721	//	CMGT	-:Q:-:-:-:-:-:-:-	0x0E203400
722	//	CMGE	-:Q:-:-:-:-:-:-:-	0x0E203C00
723	//	SSHL Vector	-:Q:-:-:-:-:-:-:-	0x0E204400
724	//	SQSHL	-:Q:-:-:-:-:-:-:-	0x0E204C00
725	//	SRSHL	-:Q:-:-:-:-:-:-:-	0x0E205400
726	//	SQRSHL	-:Q:-:-:-:-:-:-:-	0x0E205C00
727	//	SMAX	-:Q:-:-:-:-:-:-:-	0x0E206400
728	//	SMIN	-:Q:-:-:-:-:-:-:-	0x0E206C00
729	//	SABD	-:Q:-:-:-:-:-:-:-	0x0E207400
730	//	SABA	-:Q:-:-:-:-:-:-:-	0x0E207C00
731	//	ADD	-:Q:-:-:-:-:-:-:-	0x0E208400
732	//	CMTST	-:Q:-:-:-:-:-:-:-	0x0E208C00
733	//	MLA	-:Q:-:-:-:-:-:-:-	0x0E209400
734	//	MUL	-:Q:-:-:-:-:-:-:-	0x0E209C00
735	//	SMAXP	-:Q:-:-:-:-:-:-:-	0x0E20A400
736	//	SMINP	-:Q:-:-:-:-:-:-:-	0x0E20AC00
737	//	SQDMULH	-:Q:-:-:-:-:-:-:-	0x0E20B400
738	//	ADDP	-:Q:-:-:-:-:-:-:-	0x0E20BC00
739	//	FMAXNM	-:Q:-:-:-:-:-:-:-	0x0E20C400
740	//	FMLA	-:Q:-:-:-:-:-:-:-	0x0E20CC00
741	//	FADD	-:Q:-:-:-:-:-:-:-	0x0E20D400
742	//	FMULX	-:Q:-:-:-:-:-:-:-	0x0E20DC00
743	//	FCMEQ	-:Q:-:-:-:-:-:-:-	0x0E20E400
744	//	FMAX	-:Q:-:-:-:-:-:-:-	0x0E20F400
745	//	FRECPS	-:Q:-:-:-:-:-:-:-	0x0E20FC00
746	//	AND	-:Q:-:-:-:-:-:-:-	0x0E201C00
747	//	BIC	-:Q:-:-:-:-:-:-:-	0x0E601C00
748	//	FMINNM	-:Q:-:-:-:-:-:-:-	0x0EA0C400
749	//	FMLS	-:Q:-:-:-:-:-:-:-	0x0EA0CC00
750	//	FSUB	-:Q:-:-:-:-:-:-:-	0x0EA0D400
751	//	FMIN	-:Q:-:-:-:-:-:-:-	0x0EA0F400
752	//	FRSQRTS	-:Q:-:-:-:-:-:-:-	0x0EA0FC00
753	//	ORR	-:Q:-:-:-:-:-:-:-	0x0EA01C00

1	in_use	Opcode	31:30:29:28	Binary
754	//	ORN	-:Q:-:-:-:-:-:-:-:-	0x0EE01C00
755	//	UHADD	-:Q:-:-:-:-:-:-:-:-	0x2E200400
756	//	UQADD	-:Q:-:-:-:-:-:-:-:-	0x2E200C00
757	//	URHADD	-:Q:-:-:-:-:-:-:-:-	0x2E201400
758	//	UHSUB	-:Q:-:-:-:-:-:-:-:-	0x2E202400
759	//	UQSUB	-:Q:-:-:-:-:-:-:-:-	0x2E202C00
760	//	CMHI	-:Q:-:-:-:-:-:-:-:-	0x2E203400
761	//	CMHS	-:Q:-:-:-:-:-:-:-:-	0x2E203C00
762	//	USHL	-:Q:-:-:-:-:-:-:-:-	0x2E204400
763	//	UQSHL	-:Q:-:-:-:-:-:-:-:-	0x2E204C00
764	//	URSHL	-:Q:-:-:-:-:-:-:-:-	0x2E205400
765	//	UQRSHL	-:Q:-:-:-:-:-:-:-:-	0x2E205C00
766	//	UMAX	-:Q:-:-:-:-:-:-:-:-	0x2E206400
767	//	UMIN	-:Q:-:-:-:-:-:-:-:-	0x2E206C00
768	//	UABD	-:Q:-:-:-:-:-:-:-:-	0x2E207400
769	//	UABA	-:Q:-:-:-:-:-:-:-:-	0x2E207C00
770	//	SUB	-:Q:-:-:-:-:-:-:-:-	0x2E208400
771	//	CMEQ	-:Q:-:-:-:-:-:-:-:-	0x2E208C00
772	//	MLS	-:Q:-:-:-:-:-:-:-:-	0x2E209400
773	//	PMUL	-:Q:-:-:-:-:-:-:-:-	0x2E209C00
774	//	UMAXP	-:Q:-:-:-:-:-:-:-:-	0x2E20A400
775	//	UMINP	-:Q:-:-:-:-:-:-:-:-	0x2E20AC00
776	//	SQRDMULH	-:Q:-:-:-:-:-:-:-:-	0x2E20B400
777	//	FMAXNMP	-:Q:-:-:-:-:-:-:-:-	0x2E20C400
778	//	FADDP	-:Q:-:-:-:-:-:-:-:-	0x2E20D400
779	//	FMUL	-:Q:-:-:-:-:-:-:-:-	0x2E20DC00
780	//	FCMGE	-:Q:-:-:-:-:-:-:-:-	0x2E20E400
781	//	FACGE	-:Q:-:-:-:-:-:-:-:-	0x2E20EC00
782	//	FMAXP	-:Q:-:-:-:-:-:-:-:-	0x2E20F400
783	//	FDIV	-:Q:-:-:-:-:-:-:-:-	0x2E20FC00
784	//	EOR	-:Q:-:-:-:-:-:-:-:-	0x2E201C00
785	//	BSL	-:Q:-:-:-:-:-:-:-:-	0x2E601C00
786	//	FMINNMP	-:Q:-:-:-:-:-:-:-:-	0x2EA0C400
787	//	FABD	-:Q:-:-:-:-:-:-:-:-	0x2EA0D400

1	in_use	Opcode	31:30:29:28	Binary
788	//	FCMGT	-:Q:-:-:-:-:-:-:-	0x2EA0E400
789	//	FACGT	-:Q:-:-:-:-:-:-:-	0x2EA0EC00
790	//	FMINP	-:Q:-:-:-:-:-:-:-	0x2EA0F400
791	//	BIT	-:Q:-:-:-:-:-:-:-	0x2EA01C00
792	//	BIF	-:Q:-:-:-:-:-:-:-	0x2EE01C00
793	//	**AdvSIMD three different**		
794	//	SADDL	-:-:-:-:-:-:-:size	0x0E200000
795	//	SADDL2	-:-:-:-:-:-:-:size	0x4E200000
796	//	SADDW	-:-:-:-:-:-:-:size	0x0E201000
797	//	SADDW2	-:-:-:-:-:-:-:size	0x4E201000
798	//	SSUBL	-:-:-:-:-:-:-:size	0x0E202000
799	//	SSUBL2	-:-:-:-:-:-:-:size	0x4E202000
800	//	SSUBW	-:-:-:-:-:-:-:size	0x0E203000
801	//	SSUBW2	-:-:-:-:-:-:-:size	0x4E203000
802	//	ADDHN	-:-:-:-:-:-:-:size	0x0E204000
803	//	ADDHN2	-:-:-:-:-:-:-:size	0x4E204000
804	//	SABAL	-:-:-:-:-:-:-:size	0x0E205000
805	//	SABAL2	-:-:-:-:-:-:-:size	0x4E205000
806	//	SUBHN	-:-:-:-:-:-:-:size	0x0E206000
807	//	SUBHN2	-:-:-:-:-:-:-:size	0x4E206000
808	//	SABDL	-:-:-:-:-:-:-:size	0x0E207000
809	//	SABDL2	-:-:-:-:-:-:-:size	0x4E207000
810	//	SMLAL	-:-:-:-:-:-:-:size	0x0E208000
811	//	SMLAL2	-:-:-:-:-:-:-:size	0x4E208000
812	//	SQDMLAL	-:-:-:-:-:-:-:size	0x0E209000
813	//	SQDMLAL2	-:-:-:-:-:-:-:size	0x4E209000
814	//	SMLSL	-:-:-:-:-:-:-:size	0x0E20A000
815	//	SMLSL2	-:-:-:-:-:-:-:size	0x4E20A000
816	//	SQDMLSL	-:-:-:-:-:-:-:size	0x0E20B000
817	//	SQDMLSL2	-:-:-:-:-:-:-:size	0x4E20B000
818	//	SMULL	-:-:-:-:-:-:-:size	0x0E20C000
819	//	SMULL2	-:-:-:-:-:-:-:size	0x4E20C000
820	//	SQDMULL	-:-:-:-:-:-:-:size	0x0E20D000
821	//	SQDMULL2	-:-:-:-:-:-:-:size	0x4E20D000

1	in_use	Opcode	31:30:29:28	Binary
822	//	PMULL	-:-:-:-:-:-:-:size	0x0E20E000
823	//	PMULL2	-:-:-:-:-:-:-:size	0x4E20E000
824	//	UADDL	-:-:-:-:-:-:-:size	0x2E200000
825	//	UADDL2	-:-:-:-:-:-:-:size	0x6E200000
826	//	UADDW	-:-:-:-:-:-:-:size	0x2E201000
827	//	UADDW2	-:-:-:-:-:-:-:size	0x6E201000
828	//	USUBL	-:-:-:-:-:-:-:size	0x2E202000
829	//	USUBL2	-:-:-:-:-:-:-:size	0x6E202000
830	//	USUBW	-:-:-:-:-:-:-:size	0x2E203000
831	//	USUBW2	-:-:-:-:-:-:-:size	0x6E203000
832	//	RADDHN	-:-:-:-:-:-:-:size	0x2E204000
833	//	RADDHN2	-:-:-:-:-:-:-:size	0x6E204000
834	//	UABAL	-:-:-:-:-:-:-:size	0x2E205000
835	//	UABAL2	-:-:-:-:-:-:-:size	0x6E205000
836	//	RSUBHN	-:-:-:-:-:-:-:size	0x2E206000
837	//	RSUBHN2	-:-:-:-:-:-:-:size	0x6E206000
838	//	UABDL	-:-:-:-:-:-:-:size	0x2E207000
839	//	UABDL2	-:-:-:-:-:-:-:size	0x6E207000
840	//	UMLAL	-:-:-:-:-:-:-:size	0x2E208000
841	//	UMLAL2	-:-:-:-:-:-:-:size	0x6E208000
842	//	UMLSL	-:-:-:-:-:-:-:size	0x2E20A000
843	//	UMLSL2	-:-:-:-:-:-:-:size	0x6E20A000
844	//	UMULL	-:-:-:-:-:-:-:size	0x2E20C000
845	//	UMULL2	-:-:-:-:-:-:-:size	0x6E20C000
846	//	**AdvSIMD two-reg misc**		
847	//	REV64	-:Q:-:-:-:-:-:-:-	0x0E200800
848	//	REV16	-:Q:-:-:-:-:-:-:-	0x0E201800
849	//	SADDLP	-:Q:-:-:-:-:-:-:-	0x0E202800
850	//	SUQADD	-:Q:-:-:-:-:-:-:-	0x0E203800
851	//	CLS	-:Q:-:-:-:-:-:-:-	0x0E204800
852	//	CNT	-:Q:-:-:-:-:-:-:-	0x0E205800
853	//	SADALP	-:Q:-:-:-:-:-:-:-	0x0E206800
854	//	SQABS	-:Q:-:-:-:-:-:-:-	0x0E207800
855	//	CMGT	-:Q:-:-:-:-:-:-:-	0x0E208800

1	in_use	Opcode	31:30:29:28	Binary
856	//	CMEQ	-:Q:-:-:-:-:-:-:-	0x0E209800
857	//	CMLT	-:Q:-:-:-:-:-:-:-	0x0E20A800
858	//	ABS	-:Q:-:-:-:-:-:-:-	0x0E20B800
859	//	XTN	-:Q:-:-:-:-:-:-:-	0x0E212800
860	//	XTN2	-:Q:-:-:-:-:-:-:-	0x0E212800
861	//	SQXTN	-:Q:-:-:-:-:-:-:-	0x0E214800
862	//	SQXTN2	-:Q:-:-:-:-:-:-:-	0x0E214800
863	//	FCVTN	-:Q:-:-:-:-:-:-:-	0x0E216800
864	//	FCVTN2	-:Q:-:-:-:-:-:-:-	0x0E216800
865	//	FCVTL	-:Q:-:-:-:-:-:-:-	0x0E217800
866	//	FCVTL2	-:Q:-:-:-:-:-:-:-	0x0E217800
867	//	FRINTN	-:Q:-:-:-:-:-:-:-	0x0E218800
868	//	FRINTM	-:Q:-:-:-:-:-:-:-	0x0E219800
869	//	FCVTNS	-:Q:-:-:-:-:-:-:-	0x0E21A800
870	//	FCVTMS	-:Q:-:-:-:-:-:-:-	0x0E21B800
871	//	FCVTAS	-:Q:-:-:-:-:-:-:-	0x0E21C800
872	//	SCVTF	-:Q:-:-:-:-:-:-:-	0x0E21D800
873	//	FCMGT	-:Q:-:-:-:-:-:-:-	0x0EA0C800
874	//	FCMEQ	-:Q:-:-:-:-:-:-:-	0x0EA0D800
875	//	FCMLT	-:Q:-:-:-:-:-:-:-	0x0EA0E800
876	//	FABS	-:Q:-:-:-:-:-:-:-	0x0EA0F800
877	//	FRINTP	-:Q:-:-:-:-:-:-:-	0x0EA18800
878	//	FRINTZ	-:Q:-:-:-:-:-:-:-	0x0EA19800
879	//	FCVTPS	-:Q:-:-:-:-:-:-:-	0x0EA1A800
880	//	FCVTZS	-:Q:-:-:-:-:-:-:-	0x0EA1B800
881	//	URECPE	-:Q:-:-:-:-:-:-:-	0x0EA1C800
882	//	FRECPE	-:Q:-:-:-:-:-:-:-	0x0EA1D800
883	//	REV32	-:Q:-:-:-:-:-:-:-	0x2E200800
884	//	UADDLP	-:Q:-:-:-:-:-:-:-	0x2E202800
885	//	USQADD	-:Q:-:-:-:-:-:-:-	0x2E203800
886	//	CLZ	-:Q:-:-:-:-:-:-:-	0x2E204800
887	//	UADALP	-:Q:-:-:-:-:-:-:-	0x2E206800
888	//	SQNEG	-:Q:-:-:-:-:-:-:-	0x2E207800
889	//	CMGE	-:Q:-:-:-:-:-:-:-	0x2E208800

1	in_use	Opcode	31:30:29:28	Binary
890	//	CMLE	-:Q:-:-:-:-:-:-:-	0x2E209800
891	//	NEG	-:Q:-:-:-:-:-:-:-	0x2E20B800
892	//	SQXTUN	-:Q:-:-:-:-:-:-:-	0x2E212800
893	//	SQXTUN2	-:Q:-:-:-:-:-:-:-	0x2E212800
894	//	SHLL	-:Q:-:-:-:-:-:-:-	0x2E213800
895	//	SHLL2	-:Q:-:-:-:-:-:-:-	0x2E213800
896	//	UQXTN	-:Q:-:-:-:-:-:-:-	0x2E214800
897	//	UQXTN2	-:Q:-:-:-:-:-:-:-	0x2E214800
898	//	FCVTXN	-:Q:-:-:-:-:-:-:-	0x2E216800
899	//	FCVTXN2	-:Q:-:-:-:-:-:-:-	0x2E216800
900	//	FRINTA	-:Q:-:-:-:-:-:-:-	0x2E218800
901	//	FRINTX	-:Q:-:-:-:-:-:-:-	0x2E219800
902	//	FCVTNU	-:Q:-:-:-:-:-:-:-	0x2E21A800
903	//	FCVTMU	-:Q:-:-:-:-:-:-:-	0x2E21B800
904	//	FCVTAU	-:Q:-:-:-:-:-:-:-	0x2E21C800
905	//	UCVTF	-:Q:-:-:-:-:-:-:-	0x2E21D800
906	//	NOT	-:Q:-:-:-:-:-:-:-	0x2E205800
907	//	RBIT	-:Q:-:-:-:-:-:-:-	0x2E605800
908	//	FCMGE	-:Q:-:-:-:-:-:-:-	0x2EA0C800
909	//	FCMLE	-:Q:-:-:-:-:-:-:-	0x2EA0D800
910	//	FNEG	-:Q:-:-:-:-:-:-:-	0x2EA0F800
911	//	FRINTI	-:Q:-:-:-:-:-:-:-	0x2EA19800
912	//	FCVTPU	-:Q:-:-:-:-:-:-:-	0x2EA1A800
913	//	FCVTZU	-:Q:-:-:-:-:-:-:-	0x2EA1B800
914	//	URSQRTE	-:Q:-:-:-:-:-:-:-	0x2EA1C800
915	//	FRSQRTE	-:Q:-:-:-:-:-:-:-	0x2EA1D800
916	//	FSQRT	-:Q:-:-:-:-:-:-:-	0x2EA1F800
917	//	**AdvSIMD across lanes**		
918	//	SADDLV	-:Q:-:-:-:-:-:-:-	0x0E303800
919	//	SMAXV	-:Q:-:-:-:-:-:-:-	0x0E30A800
920	//	SMINV	-:Q:-:-:-:-:-:-:-	0x0E31A800
921	//	ADDV	-:Q:-:-:-:-:-:-:-	0x0E31B800
922	//	UADDLV	-:Q:-:-:-:-:-:-:-	0x2E303800
923	//	UMAXV	-:Q:-:-:-:-:-:-:-	0x2E30A800

1	in_use	Opcode	31:30:29:28	Binary
924	//	UMINV	-:Q:-:-:-:-:-:-:-:-	0x2E31A800
925	//	FMAXNMV	-:Q:-:-:-:-:-:-:-:-	0x2E30C800
926	//	FMAXV	-:Q:-:-:-:-:-:-:-:-	0x2E30F800
927	//	FMINNMV	-:Q:-:-:-:-:-:-:-:-	0x2EB0C800
928	//	FMINV	-:Q:-:-:-:-:-:-:-:-	0x2EB0F800
929	//	**AdvSIMD copy**		
930	//	DUP	-:-:-:-:-:-:-:-:-:-	0x0E000400
931	//	DUP	-:-:-:-:-:-:-:-:-:-	0x0E000C00
932	//	SMOV	-:-:-:-:-:-:-:-:-:-	0x0E002C00
933	//	UMOV	-:-:-:-:-:-:-:-:-:-	0x0E003C00
934	//	INS	-:-:-:-:-:-:-:-:-:-	0x4E001C00
935	//	SMOV	-:-:-:-:-:-:-:-:-:-	0x4E002C00
936	//	UMOV	-:-:-:-:-:-:-:-:-:-	0x4E003C00
937	//	INS	-:-:-:-:-:-:-:-:-:-	0x6E000400
938	//	**AdvSIMD vector x indexed ele**		
939	//	SMLAL	-:Q:-:-:-:-:-:-:-:-	0x0F002000
940	//	SMLAL2	-:Q:-:-:-:-:-:-:-:-	0x0F002000
941	//	SQDMLAL	-:Q:-:-:-:-:-:-:-:-	0x0F003000
942	//	SQDMLAL2	-:Q:-:-:-:-:-:-:-:-	0x0F003000
943	//	SMLSL	-:Q:-:-:-:-:-:-:-:-	0x0F006000
944	//	SMLSL2	-:Q:-:-:-:-:-:-:-:-	0x0F006000
945	//	SQDMLSL	-:Q:-:-:-:-:-:-:-:-	0x0F007000
946	//	SQDMLSL2	-:Q:-:-:-:-:-:-:-:-	0x0F007000
947	//	MUL	-:Q:-:-:-:-:-:-:-:-	0x0F008000
948	//	SMULL	-:Q:-:-:-:-:-:-:-:-	0x0F00A000
949	//	SMULL2	-:Q:-:-:-:-:-:-:-:-	0x0F00A000
950	//	SQDMULL	-:Q:-:-:-:-:-:-:-:-	0x0F00B000
951	//	SQDMULL2	-:Q:-:-:-:-:-:-:-:-	0x0F00B000
952	//	SQDMULH	-:Q:-:-:-:-:-:-:-:-	0x0F00C000
953	//	SQRDMULH	-:Q:-:-:-:-:-:-:-:-	0x0F00D000
954	//	FMLA	-:Q:-:-:-:-:-:-:-:-	0x0F801000
955	//	FMLS	-:Q:-:-:-:-:-:-:-:-	0x0F805000
956	//	FMUL	-:Q:-:-:-:-:-:-:-:-	0x0F809000
957	//	MLA	-:Q:-:-:-:-:-:-:-:-	0x2F000000

1	in_use	Opcode	31:30:29:28	Binary
958	//	UMLAL	-:Q:-:-:-:-:-:-:-:	0x2F002000
959	//	UMLAL2	-:Q:-:-:-:-:-:-:-:	0x2F002000
960	//	MLS	-:Q:-:-:-:-:-:-:-:	0x2F004000
961	//	UMLSL	-:Q:-:-:-:-:-:-:-:	0x2F006000
962	//	UMLSL2	-:Q:-:-:-:-:-:-:-:	0x2F006000
963	//	UMULL	-:Q:-:-:-:-:-:-:-:	0x2F00A000
964	//	UMULL2	-:Q:-:-:-:-:-:-:-:	0x2F00A000
965	//	FMULX	-:Q:-:-:-:-:-:-:-:	0x2F809000
966	//	**AdvSIMD modified immediate**		
967	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x0F000400
968	//	ORR	-:-:-:-:-:-:-:-:-:-	0x0F001400
969	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x0F008400
970	//	ORR	-:-:-:-:-:-:-:-:-:-	0x0F009400
971	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x0F00C400
972	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x0F00E400
973	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x0F00F400
974	//	MVNI	-:-:-:-:-:-:-:-:-:-	0x2F000400
975	//	BIC	-:-:-:-:-:-:-:-:-:-	0x2F001400
976	//	MVNI	-:-:-:-:-:-:-:-:-:-	0x2F008400
977	//	BIC	-:-:-:-:-:-:-:-:-:-	0x2F009400
978	//	MVNI	-:-:-:-:-:-:-:-:-:-	0x2F00C400
979	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x2F00E400
980	//	MOVI	-:-:-:-:-:-:-:-:-:-	0x6F00E400
981	//	FMOV	-:-:-:-:-:-:-:-:-:-	0x6F00F400
982	//	**AdvSIMD shift by immediate**		
983	//	SSHR	-:Q:-:-:-:-:-:-:ir	0x0F000400
984	//	SSRA	-:Q:-:-:-:-:-:-:ir	0x0F001400
985	//	SRSHR	-:Q:-:-:-:-:-:-:ir	0x0F002400
986	//	SRSRA	-:Q:-:-:-:-:-:-:ir	0x0F003400
987	//	SHL	-:Q:-:-:-:-:-:-:ir	0x0F005400
988	//	SQSHL	-:Q:-:-:-:-:-:-:ir	0x0F007400
989	//	SHRN	-:Q:-:-:-:-:-:-:ir	0x0F008400
990	//	SHRN2	-:Q:-:-:-:-:-:-:ir	0x0F008400
991	//	RSHRN	-:Q:-:-:-:-:-:-:ir	0x0F008C00

1	in_use	Opcode	31:30:29:28	Binary
992	//	RSHRN2	-:Q:-:-:-:-:-:-:-:ir	0x0F008C00
993	//	SQSHRN	-:Q:-:-:-:-:-:-:-:ir	0x0F009400
994	//	SQSHRN2	-:Q:-:-:-:-:-:-:-:ir	0x0F009400
995	//	SQRSHRN	-:Q:-:-:-:-:-:-:-:ir	0x0F009C00
996	//	SQRSHRN2	-:Q:-:-:-:-:-:-:-:ir	0x0F009C00
997	//	SSHLL	-:Q:-:-:-:-:-:-:-:ir	0x0F00A400
998	//	SSHLL2	-:Q:-:-:-:-:-:-:-:ir	0x0F00A400
999	//	SCVTF	-:Q:-:-:-:-:-:-:-:ir	0x0F00E400
1000	//	FCVTZS	-:Q:-:-:-:-:-:-:-:ir	0x0F00FC00
1001	//	USHR	-:Q:-:-:-:-:-:-:-:ir	0x2F000400
1002	//	USRA	-:Q:-:-:-:-:-:-:-:ir	0x2F001400
1003	//	URSHR	-:Q:-:-:-:-:-:-:-:ir	0x2F002400
1004	//	URSRA	-:Q:-:-:-:-:-:-:-:ir	0x2F003400
1005	//	SRI	-:Q:-:-:-:-:-:-:-:ir	0x2F004400
1006	//	SLI	-:Q:-:-:-:-:-:-:-:ir	0x2F005400
1007	//	SQSHLU	-:Q:-:-:-:-:-:-:-:ir	0x2F006400
1008	//	UQSHL	-:Q:-:-:-:-:-:-:-:ir	0x2F007400
1009	//	SQSHRUN	-:Q:-:-:-:-:-:-:-:ir	0x2F008400
1010	//	SQSHRUN2	-:Q:-:-:-:-:-:-:-:ir	0x2F008400
1011	//	SQRSHRUN	-:Q:-:-:-:-:-:-:-:ir	0x2F008C00
1012	//	SQRSHRUN2	-:Q:-:-:-:-:-:-:-:ir	0x2F008C00
1013	//	UQSHRN	-:Q:-:-:-:-:-:-:-:ir	0x2F009400
1014	//	UQRSHRN	-:Q:-:-:-:-:-:-:-:ir	0x2F009C00
1015	//	UQRSHRN2	-:Q:-:-:-:-:-:-:-:ir	0x2F009C00
1016	//	USHLL	-:Q:-:-:-:-:-:-:-:ir	0x2F00A400
1017	//	USHLL2	-:Q:-:-:-:-:-:-:-:ir	0x2F00A400
1018	//	UCVTF	-:Q:-:-:-:-:-:-:-:ir	0x2F00E400
1019	//	FCVTZU	-:Q:-:-:-:-:-:-:-:ir	0x2F00FC00
1020	//	**AdvSIMD TBL/TBX**		
1021	//	TBL	-:Q:-:-:-:-:-:-:-:-	0x0E000000
1022	//	TBX	-:Q:-:-:-:-:-:-:-:-	0x0E001000
1023	//	TBL	-:Q:-:-:-:-:-:-:-:-	0x0E002000
1024	//	TBX	-:Q:-:-:-:-:-:-:-:-	0x0E003000
1025	//	TBL	-:Q:-:-:-:-:-:-:-:-	0x0E004000

1	in_use	Opcode	31:30:29:28	Binary
1026	//	TBX	-:Q:-:-:-:-:-:-:-:-	0x0E005000
1027	//	TBL	-:Q:-:-:-:-:-:-:-:-	0x0E006000
1028	//	TBX	-:Q:-:-:-:-:-:-:-:-	0x0E007000
1029	//	**AdvSIMD ZIP/UZP/TRN**		
1030	//	UZP1	-:Q:-:-:-:-:-:-:siz	0x0E001800
1031	//	TRN1	-:Q:-:-:-:-:-:-:siz	0x0E002800
1032	//	ZIP1	-:Q:-:-:-:-:-:-:siz	0x0E003800
1033	//	UZP2	-:Q:-:-:-:-:-:-:siz	0x0E005800
1034	//	TRN2	-:Q:-:-:-:-:-:-:siz	0x0E006800
1035	//	ZIP2	-:Q:-:-:-:-:-:-:siz	0x0E007800
1036	//	**AdvSIMD EXT**		
1037	//	EXT	-:Q:-:-:-:-:-:-:-:-	0x2E000000
1038	//	**Loads and stores**		
1039	//	**AdvSIMD load/store multiple s**		
1040	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0C000000
1041	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C002000
1042	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0C004000
1043	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C006000
1044	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C007000
1045	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0C008000
1046	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C00A000
1047	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0C400000
1048	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0C402000
1049	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0C404000
1050	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0C406000
1051	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0C407000
1052	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0C408000
1053	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0C40A000
1054	//	**AdvSIMD load/store multiple s**		
1055	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0C800000
1056	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C802000
1057	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0C804000
1058	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C806000
1059	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C807000

1	in_use	Opcode	31:30:29:28	Binary
1060	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0C808000
1061	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C80A000
1062	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0C9F0000
1063	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C9F2000
1064	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0C9F4000
1065	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C9F6000
1066	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C9F7000
1067	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0C9F8000
1068	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0C9FA000
1069	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0CC00000
1070	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CC02000
1071	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0CC04000
1072	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CC06000
1073	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CC07000
1074	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0CC08000
1075	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CC0A000
1076	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0CDF0000
1077	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CDF2000
1078	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0CDF4000
1079	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CDF6000
1080	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CDF7000
1081	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0CDF8000
1082	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0CDFA000
1083	//	**AdvSIMD load/store single st**		
1084	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D000000
1085	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D002000
1086	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D004000
1087	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D006000
1088	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D008000
1089	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D008400
1090	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D00A000
1091	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D00A400
1092	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0D200000
1093	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0D202000

1	in_use	Opcode	31:30:29:28	Binary
1094	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0D204000
1095	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0D206000
1096	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0D208000
1097	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0D208400
1098	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0D20A000
1099	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0D20A400
1100	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0D400000
1101	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0D402000
1102	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0D404000
1103	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0D406000
1104	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0D408000
1105	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0D408400
1106	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0D40A000
1107	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0D40A400
1108	//	LD1R	-:Q:-:-:-:-:-:-:-:-	0x0D40C000
1109	//	LD3R	-:Q:-:-:-:-:-:-:-:-	0x0D40E000
1110	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0D600000
1111	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0D602000
1112	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0D604000
1113	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0D606000
1114	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0D608000
1115	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0D608400
1116	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0D60A000
1117	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0D60A400
1118	//	LD2R	-:Q:-:-:-:-:-:-:-:-	0x0D60C000
1119	//	LD4R	-:Q:-:-:-:-:-:-:-:-	0x0D60E000
1120	//	**AdvSIMD load/store single st**		
1121	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D800000
1122	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D802000
1123	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D804000
1124	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D806000
1125	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D808000
1126	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D808400
1127	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D80A000

1	in_use	Opcode	31:30:29:28	Binary
1128	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D80A400
1129	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D9F0000
1130	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D9F2000
1131	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D9F4000
1132	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D9F6000
1133	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D9F8000
1134	//	ST1	-:Q:-:-:-:-:-:-:-:-	0x0D9F8400
1135	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D9FA000
1136	//	ST3	-:Q:-:-:-:-:-:-:-:-	0x0D9FA400
1137	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DA00000
1138	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DA02000
1139	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DA04000
1140	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DA06000
1141	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DA08000
1142	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DA08400
1143	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DA0A000
1144	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DA0A400
1145	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DBF0000
1146	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DBF2000
1147	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DBF4000
1148	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DBF6000
1149	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DBF8000
1150	//	ST2	-:Q:-:-:-:-:-:-:-:-	0x0DBF8400
1151	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DBFA000
1152	//	ST4	-:Q:-:-:-:-:-:-:-:-	0x0DBFA400
1153	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DC00000
1154	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DC02000
1155	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DC04000
1156	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DC06000
1157	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DC08000
1158	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DC08400
1159	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DC0A000
1160	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DC0A400
1161	//	LD1R	-:Q:-:-:-:-:-:-:-:-	0x0DC0C000

1	in_use	Opcode	31:30:29:28	Binary
1162	//	LD3R	-:Q:-:-:-:-:-:-:-:-	0x0DC0E000
1163	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DDF0000
1164	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DDF2000
1165	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DDF4000
1166	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DDF6000
1167	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DDF8000
1168	//	LD1	-:Q:-:-:-:-:-:-:-:-	0x0DDF8400
1169	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DDFA000
1170	//	LD3	-:Q:-:-:-:-:-:-:-:-	0x0DDFA400
1171	//	LD1R	-:Q:-:-:-:-:-:-:-:-	0x0DDFC000
1172	//	LD3R	-:Q:-:-:-:-:-:-:-:-	0x0DDFE000
1173	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DE00000
1174	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DE02000
1175	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DE04000
1176	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DE06000
1177	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DE08000
1178	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DE08400
1179	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DE0A000
1180	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DE0A400
1181	//	LD2R	-:Q:-:-:-:-:-:-:-:-	0x0DE0C000
1182	//	LD4R	-:Q:-:-:-:-:-:-:-:-	0x0DE0E000
1183	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DFF0000
1184	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DFF2000
1185	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DFF4000
1186	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DFF6000
1187	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DFF8000
1188	//	LD2	-:Q:-:-:-:-:-:-:-:-	0x0DFF8400
1189	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DFFA000
1190	//	LD4	-:Q:-:-:-:-:-:-:-:-	0x0DFFA400
1191	//	LD2R	-:Q:-:-:-:-:-:-:-:-	0x0DFFC000
1192	//	LD4R	-:Q:-:-:-:-:-:-:-:-	0x0DFFE000

1	**in_use**	**Opcode**	**NAME**
2		**UNALLOCATED**	
3		BAD	bad,
4		**Branch,exception generatic**	
5		**Compare _ Branch (immediat**	
6		CBZ	cbzw,
7		CBNZ	cbnzw,
8		CBZ	cbzx,
9		CBNZ	cbnzx,
10		**Test bit & branch (immediate)**	
11		TBZ	tbz,
12		TBNZ	tbnz,
13		**Conditional branch (immediat**	
14		B_cond	b_cond,
15		**Exception generation**	
16	//	SVC	svc,
17	//	HVC	hvc,
18	//	SMC	smc,
19		BRK	brkarm64,
20	//	HLT	hlt,
21	//	DCPS1	dcps1,
22	//	DCPS2	dcps2,
23	//	DCPS3	dcps3,
24	//	**System**	
25	//	MSR	msrimm,
26	//	HINT	hint,
27	//	CLREX	clrex,
28	//	DSB	dsb,
29	//	DMB	dmb,
30	//	ISB	isb,
31	//	SYS	sys,
32	//	MSR	msr,
33	//	SYSL	sysl,
34	//	MRS	mrs,
35		**Unconditional branch (registe**	
36		BR	br,
37		BLR	blr,

	in_use	Opcode	NAME
38		RET	ret,
39	//	ERET	eret,
40	//	DRPS	drps,
41	//	**Unconditional branch (immed**	
42		B	b,
43		BL	bl,
44		**Loads and stores**	
45		**Load/store exclusive**	
46		STXRB	stxrb,
47		STLXRB	stlxrb,
48		LDXRB	ldxrb,
49		LDAXRB	ldaxrb,
50		STLRB	stlrb,
51		LDARB	ldarb,
52		STXRH	stxrh,
53		STLXRH	stlxrh,
54		LDXRH	ldxrh,
55		LDAXRH	ldaxrh,
56		STLRH	stlrh,
57		LDARH	ldarh,
58		STXR	stxrw,
59		STLXR	stlxrw,
60		STXP	stxpw,
61		STLXP	stlxpw,
62		LDXR	ldxrw,
63		LDAXR	ldaxrw,
64		LDXP	ldxpw,
65		LDAXP	ldaxpw,
66		STLR	stlrw,
67		LDAR	ldarw,
68		STXR	stxrx,
69		STLXR	stlxrx,
70		STXP	stxpx,
71		STLXP	stlxpx,
72		LDXR	ldxrx,
73		LDAXR	ldaxrx,
74		LDXP	ldxpx,

	in_use	Opcode	NAME
75		LDAXP	ldaxpx,
76		STLR	stlrx,
77		LDAR	ldarx,
78		**Load register (literal)**	
79		LDR	ldrw,
80		LDR	vldrs,
81		LDR	ldrx,
82		LDR	vldrd,
83		LDRSW	ldrsw,
84		LDR	vldrq,
85		PRFM	prfm,
86		**Load/store no-allocate pair (o**	
87		STNP	stnpw,
88		LDNP	ldnpw,
89		STNP	vstnps,
90		LDNP	vldnps,
91		STNP	vstnpd,
92		LDNP	vldnpd,
93		STNP	stnpx,
94		LDNP	ldnpx,
95		STNP	vstnpq,
96		LDNP	vldnpq,
97		**Load/store register pair (post**	
98		STP	stppostw,
99		LDP	ldppostw,
100		STP	vstpposts,
101		LDP	vldpposts,
102		LDPSW	ldpswpost,
103		STP	vstppostd,
104		LDP	vldppostd,
105		STP	stppostx,
106		LDP	ldppostx,
107		STP	vstppostq,
108		LDP	vldppostq,
109		**Load/store register pair (offse**	

1	in_use	Opcode	NAME
110		STP	stpoffw,
111		LDP	ldpoffw,
112		STP	vstpoffs,
113		LDP	vldpoffs,
114		LDPSW	ldpswoff,
115		STP	vstpoffd,
116		LDP	vldpoffd,
117		STP	stpoffx,
118		LDP	ldpoffx,
119		STP	vstpoffq,
120		LDP	vldpoffq,
121		**Load/store register pair (pre-i**	
122		STP	stpprew,
123		LDP	ldpprew,
124		STP	vstppres,
125		LDP	vldppres,
126		LDPSW	ldpswpre,
127		STP	vstppred,
128		LDP	vldppred,
129		STP	stpprex,
130		LDP	ldpprex,
131		STP	vstppreq,
132		LDP	vldppreq,
133		**Load/store register (unscaled**	
134		STURB	sturb,
135		LDURB	ldurb,
136		LDURSB	ldursbx,
137		LDURSB	ldursbw,
138		STUR	vsturb,
139		LDUR	vldurb,
140		STUR	vsturq,
141		LDUR	vldurq,
142		STURH	sturh,
143		LDURH	ldurh,
144		LDURSH	ldurshx,
145		LDURSH	ldurshw,
146		STUR	vsturh,
147		LDUR	vldurh,

1	in_use	Opcode	NAME
148		STUR	sturw,
149		LDUR	ldurw,
150		LDURSW	ldursw,
151		STUR	vsturs,
152		LDUR	vldurs,
153		STUR	sturx,
154		LDUR	ldurx,
155		PRFUM	prfum,
156		STUR	vsturd,
157		LDUR	vldurd,
158		**Load/store register (immediat**	
159		STRB	strbpost,
160		LDRB	ldrbpost,
161		LDRSB	ldrsbpostx,
162		LDRSB	ldrsbpostw,
163		STR	vstrpostb,
164		LDR	vldrpostb,
165		STR	vstrpostq,
166		LDR	vldrpostq,
167		STRH	strhpost,
168		LDRH	ldrhpost,
169		LDRSH	ldrshpostx,
170		LDRSH	ldrshpostw,
171		STR	vstrposth,
172		LDR	vldrposth,
173		STR	strpostw,
174		LDR	ldrpostw,
175		LDRSW	ldrswpost,
176		STR	vstrposts,
177		LDR	vldrposts,
178		STR	strpostx,
179		LDR	ldrpostx,
180		STR	vstrpostd,
181		LDR	vldrpostd,
182		**Load/store register (unprivile**	
183		STTRB	sttrb,
184		LDTRB	ldtrb,
185		LDTRSB	ldtrsbx,

1	in_use	Opcode	NAME
186		LDTRSB	ldtrsbw,
187		STTRH	sttrh,
188		LDTRH	ldtrh,
189		LDTRSH	ldtrshx,
190		LDTRSH	ldtrshw,
191		STTR	sttrw,
192		LDTR	ldtrw,
193		LDTRSW	ldtrsw,
194		STTR	sttrx,
195		LDTR	ldtrx,
196		**Load/store register (immediat**	
197		STRB	strbpre,
198		LDRB	ldrbpre,
199		LDRSB	ldrsbprex,
200		LDRSB	ldrsbprew,
201		STR	vstrpreb,
202		LDR	vldrpreb,
203		STR	vstrpreq,
204		LDR	vldrpreq,
205		STRH	strhpre,
206		LDRH	ldrhpre,
207		LDRSH	ldrshprex,
208		LDRSH	ldrshprew,
209		STR	vstrpreh,
210		LDR	vldrpreh,
211		STR	strprew,
212		LDR	ldrprew,
213		LDRSW	ldrswpre,
214		STR	vstrpres,
215		LDR	vldrpres,
216		STR	strprex,
217		LDR	ldrprex,
218		STR	vstrpred,
219		LDR	vldrpred,
220		**Load/store register (register o**	
221		STRB	strboff,
222		LDRB	ldrboff,
223		LDRSB	ldrsboffx,

1	in_use	Opcode	NAME
224		LDRSB	ldrsboffw,
225		STR	vstroffb,
226		LDR	vldroffb,
227		STR	vstroffq,
228		LDR	vldroffq,
229		STRH	strhoff,
230		LDRH	ldrhoff,
231		LDRSH	ldrshoffx,
232		LDRSH	ldrshoffw,
233		STR	vstroffh,
234		LDR	vldroffh,
235		STR	stroffw,
236		LDR	ldroffw,
237		LDRSW	ldrswoff,
238		STR	vstroffs,
239		LDR	vldroffs,
240		STR	stroffx,
241		LDR	ldroffx,
243		STR	vstroffd,
244		LDR	vldroffd,
242		PRFM	prfmoff,
245		**Load/store register (unsigned**	
246		STRB	strbimm,
247		LDRB	ldrbimm,
248		LDRSB	ldrsbimmx,
249		LDRSB	ldrsbimmw,
250		STR	vstrimmb,
251		LDR	vldrimmb,
252		STR	vstrimmq,
253		LDR	vldrimmq,
254		STRH	strhimm,
255		LDRH	ldrhimm,
256		LDRSH	ldrshimmx,
257		LDRSH	ldrshimmw,
258		STR	vstrimmh,
259		LDR	vldrimmh,
260		STR	strimmw,
261		LDR	ldrimmw,

1	in_use	Opcode	NAME
262		LDRSW	ldrswimm,
263		STR	vstrimms,
264		LDR	vldrimms,
265		STR	strimmx,
266		LDR	ldrimmx,
268		STR	vstrimmd,
269		LDR	vldrimmd,
267		PRFM	prfmimm,
270	**Data processing – Immedia**		
271	**PC-rel. addressing**		
272		ADR	adr,
273		ADRP	adrp,
274	**Add/subtract (immediate)**		
275		ADD	addimmw,
276		ADDS	addsimmw,
277		SUB	subimmw,
278		SUBS	subsimmw,
279		ADD	addimmx,
280		ADDS	addsimmx,
281		SUB	subimmx,
282		SUBS	subsimmx,
283	**Logical (immediate)**		
284		AND	andimmw,
285		ORR	orrimmw,
286		EOR	eorimmw,
287		ANDS	andsimmw,
288		AND	andimmx,
289		ORR	orrimmx,
290		EOR	eorimmx,
291		ANDS	andsimmx,
292	**Move wide (immediate)**		
293		MOVN	movnw,
294		MOVZ	movzw,
295		MOVK	movkw,
296		MOVN	movnx,
297		MOVZ	movzx,
298		MOVK	movkx,
299	**Bitfield**		

1	in_use	Opcode	NAME
300		SBFM	sbfmw,
301		BFM	bfmw,
302		UBFM	ubfmw,
303		SBFM	sbfmx,
304		BFM	bfmx,
305		UBFM	ubfmx,
306		**Extract**	
307		EXTR	extrw,
308		EXTR	extrx,
309	**Data Processing – register**		
310		**Logical (shifted register)**	
311		AND	andw,
312		BIC	bicw,
313		ORR	orrw,
314		ORN	ornw,
315		EOR	eorw,
316		EON	eonw,
317		ANDS	andsw,
318		BICS	bicsw,
319		AND	andx,
320		BIC	bicx,
321		ORR	orrx,
322		ORN	ornx,
323		EOR	eorx,
324		EON	eonx,
325		ANDS	andsx,
326		BICS	bicsx,
327		**Add/subtract (shifted register**	
328		ADD	addw,
329		ADDS	addsw,
330		SUB	subw,
331		SUBS	subsw,
332		ADD	addx,
333		ADDS	addsx,
334		SUB	subx,
335		SUBS	subsx,
336		**Add/subtract (extended regist**	
337		ADD	addextw,

1	in_use	Opcode	NAME
338		ADDS	addsextw,
339		SUB	subextw,
340		SUBS	subsextw,
341		ADD	addextx,
342		ADDS	addsextx,
343		SUB	subextx,
344		SUBS	subsextx,
345		**Add/subtract (with carry)**	
346		ADC	adcw,
347		ADCS	adcsw,
348		SBC	sbcw,
349		SBCS	sbcsw,
350		ADC	adcx,
351		ADCS	adcsx,
352		SBC	sbcx,
353		SBCS	sbcsx,
354		**Conditional compare (registe**	
355		CCMN	ccmnw,
356		CCMN	ccmnx,
357		CCMP	ccmpw,
358		CCMP	ccmpx,
359		**Conditional compare (immedi**	
360		CCMN	ccmnimmw,
361		CCMN	ccmnimmx,
362		CCMP	ccmpimmw,
363		CCMP	ccmpimmx,
364		**Conditional select**	
365		CSEL	cselw,
366		CSINC	csincw,
367		CSINV	csinvw,
368		CSNEG	csnegw,
369		CSEL	cselx,
370		CSINC	csincx,
371		CSINV	csinvx,
372		CSNEG	csnegx,
373		**Data-processing (3 source)**	
374		MADD	maddw,
375		MADD	maddx,

1	in_use	Opcode	NAME
376		SMADDL	smaddl,
377		UMADDL	umaddl,
378		MSUB	msubw,
379		MSUB	msubx,
380		SMSUBL	smsubl,
381		UMSUBL	umsubl,
382		SMULH	smulh,
383		UMULH	umulh,
384		**Data-processing (2 source)**	
385		CRC32X	crc32x,
386		CRC32CX	crc32cx,
387		CRC32B	crc32b,
388		CRC32CB	crc32cb,
389		CRC32H	crc32h,
390		CRC32CH	crc32ch,
391		CRC32W	crc32w,
392		CRC32CW	crc32cw,
393		UDIV	udivw,
394		UDIV	udivx,
395		SDIV	sdivw,
396		SDIV	sdivx,
397		LSLV	lslvw,
398		LSLV	lslvx,
399		LSRV	lsrvw,
400		LSRV	lsrvx,
401		ASRV	asrvw,
402		ASRV	asrvx,
403		RORV	rorvw,
404		RORV	rorvx,
405		**Data-processing (1 source)**	
406		RBIT	rbitw,
407		RBIT	rbitx,
408		CLZ	clzw,
409		CLZ	clzx,
410		CLS	clsw,
411		CLS	clsx,
412		REV	revw,
413		REV	revx,

```
1   in_use   Opcode                NAME
414          REV16                 rev16w,
415          REV16                 rev16x,
416          REV32                 rev32,
417 //   Data Processing – SIMD an
418 //       Floating-point<->fixed-point c
419 //           SCVTF             vscvtfscalar_fixed_point_32_bit_to_single_precision,
420 //           UCVTF             vucvtfscalar_fixed_point_32_bit_to_single_precision,
421 //           FCVTZS            vfcvtzsscalar_fixed_point_Single_precision_to_32_bit,
422 //           FCVTZU            vfcvtzuscalar_fixed_point_Single_precision_to_32_bit,
423 //           SCVTF             vscvtfscalar_fixed_point_32_bit_to_double_precision,
424 //           UCVTF             vucvtfscalar_fixed_point_32_bit_to_double_precision,
425 //           FCVTZS            vfcvtzsscalar_fixed_point_Double_precision_to_32_bit,
426 //           FCVTZU            vfcvtzuscalar_fixed_point_Double_precision_to_32_bit,
427 //           SCVTF             vscvtfscalar_fixed_point_64_bit_to_single_precision,
428 //           UCVTF             vucvtfscalar_fixed_point_64_bit_to_single_precision,
429 //           FCVTZS            vfcvtzsscalar_fixed_point_Single_precision_to_64_bit,
430 //           FCVTZU            vfcvtzuscalar_fixed_point_Single_precision_to_64_bit,
431 //           SCVTF             vscvtfscalar_fixed_point_64_bit_to_double_precision,
432 //           UCVTF             vucvtfscalar_fixed_point_64_bit_to_double_precision,
433 //           FCVTZS            vfcvtzsscalar_fixed_point_Double_precision_to_64_bit,
434 //           FCVTZU            vfcvtzuscalar_fixed_point_Double_precision_to_64_bit,
435 //       Floating-point conditional cor
436 //           FCCMP             vfccmpSingle_precision,
437 //           FCCMPE            vfccmpeSingle_precision,
438 //           FCCMP             vfccmpDouble_precision,
439 //           FCCMPE            vfccmpeDouble_precision,
440 //       Floating-point data-processin
441 //           FMUL              vfmulscalar_Single_precision,
442 //           FDIV              vfdivscalar_Single_precision,
443 //           FADD              vfaddscalar_Single_precision,
444 //           FSUB              vfsubscalar_Single_precision,
445 //           FMAX              vfmaxscalar_Single_precision,
446 //           FMIN              vfminscalar_Single_precision,
447 //           FMAXNM            vfmaxnmscalar_Single_precision,
```

1	in_use	Opcode	NAME
448	//	FMINNM	vfminnmscalar_Single_precision,
449	//	FNMUL	vfnmulSingle_precision,
450	//	FMUL	vfmulscalar_Double_precision,
451	//	FDIV	vfdivscalar_Double_precision,
452	//	FADD	vfaddscalar_Double_precision,
453	//	FSUB	vfsubscalar_Double_precision,
454	//	FMAX	vfmaxscalar_Double_precision,
455	//	FMIN	vfminscalar_Double_precision,
456	//	FMAXNM	vfmaxnmscalar_Double_precision,
457	//	FMINNM	vfminnmscalar_Double_precision,
458	//	FNMUL	vfnmulDouble_precision,
459	//	**Floating-point conditional sel**	
460	//	FCSEL	vfcselSingle_precision,
461	//	FCSEL	vfcselDouble_precision,
462	//	**Floating-point immediate**	
463	//	FMOV	vfmovscalar_immediate_Single_precision,
464	//	FMOV	vfmovscalar_immediate_Double_precision,
465	//	**Floating-point compare**	
466	//	FCMP	vfcmpSingle_precision,
467	//	FCMP	vfcmpSingle_precision_zero,
468	//	FCMPE	vfcmpeSingle_precision,
469	//	FCMPE	vfcmpeSingle_precision_zero,
470	//	FCMP	vfcmpDouble_precision,
471	//	FCMP	vfcmpDouble_precision_zero,
472	//	FCMPE	vfcmpeDouble_precision,
473	//	FCMPE	vfcmpeDouble_precision_zero,
474	//	**Floating-point data-processin**	
475	//	FMOV	vfmovregister_Single_precision,
476	//	FABS	vfabsscalar_Single_precision,
477	//	FNEG	vfnegscalar_Single_precision,
478	//	FSQRT	vfsqrtscalar_Single_precision,
479	//	FCVT	vfcvtSingle_precision_to_double_precision,
480	//	FCVT	vfcvtSingle_precision_to_half_precision,
481	//	FRINTN	vfrintnscalar_Single_precision,

1	in_use	Opcode	NAME
482	//	FRINTP	vfrintpscalar_Single_precision,
483	//	FRINTM	vfrintmscalar_Single_precision,
484	//	FRINTZ	vfrintzscalar_Single_precision,
485	//	FRINTA	vfrintascalar_Single_precision,
486	//	FRINTX	vfrintxscalar_Single_precision,
487	//	FRINTI	vfrintiscalar_Single_precision,
488	//	FMOV	vfmovregister_Double_precision,
489	//	FABS	vfabsscalar_Double_precision,
490	//	FNEG	vfnegscalar_Double_precision,
491	//	FSQRT	vfsqrtscalar_Double_precision,
492	//	FCVT	vfcvtDouble_precision_to_single_precision,
493	//	FCVT	vfcvtDouble_precision_to_half_precision,
494	//	FRINTN	vfrintnscalar_Double_precision,
495	//	FRINTP	vfrintpscalar_Double_precision,
496	//	FRINTM	vfrintmscalar_Double_precision,
497	//	FRINTZ	vfrintzscalar_Double_precision,
498	//	FRINTA	vfrintascalar_Double_precision,
499	//	FRINTX	vfrintxscalar_Double_precision,
500	//	FRINTI	vfrintiscalar_Double_precision,
501	//	FCVT	vfcvtHalf_precision_to_single_precision,
502	//	FCVT	vfcvtHalf_precision_to_double_precision,
503	//	**Floating-point<->integer conv**	
504	//	FCVTNS	vfcvtnsscalar_Single_precision_to_32_bit,
505	//	FCVTNU	vfcvtnuscalar_Single_precision_to_32_bit,
506	//	SCVTF	vscvtfscalar_integer_32_bit_to_single_precision,
507	//	UCVTF	vucvtfscalar_integer_32_bit_to_single_precision,
508	//	FCVTAS	vfcvtasscalar_Single_precision_to_32_bit,
509	//	FCVTAU	vfcvtauscalar_Single_precision_to_32_bit,
510	//	FMOV	vfmovgeneral_Single_precision_to_32_bit,
511	//	FMOV	vfmovgeneral_32_bit_to_single_precision,
512	//	FCVTPS	vfcvtpsscalar_Single_precision_to_32_bit,
513	//	FCVTPU	vfcvtpuscalar_Single_precision_to_32_bit,
514	//	FCVTMS	vfcvtmsscalar_Single_precision_to_32_bit,
515	//	FCVTMU	vfcvtmuscalar_Single_precision_to_32_bit,

1	in_use	Opcode	NAME
516	//	FCVTZS	vfcvtzsscalar_integer_Single_precision_to_32_bit,
517	//	FCVTZU	vfcvtzuscalar_integer_Single_precision_to_32_bit,
518	//	FCVTNS	vfcvtnsscalar_Double_precision_to_32_bit,
519	//	FCVTNU	vfcvtnuscalar_Double_precision_to_32_bit,
520	//	SCVTF	vscvtfscalar_integer_32_bit_to_double_precision,
521	//	UCVTF	vucvtfscalar_integer_32_bit_to_double_precision,
522	//	FCVTAS	vfcvtasscalar_Double_precision_to_32_bit,
523	//	FCVTAU	vfcvtauscalar_Double_precision_to_32_bit,
524	//	FCVTPS	vfcvtpsscalar_Double_precision_to_32_bit,
525	//	FCVTPU	vfcvtpuscalar_Double_precision_to_32_bit,
526	//	FCVTMS	vfcvtmsscalar_Double_precision_to_32_bit,
527	//	FCVTMU	vfcvtmuscalar_Double_precision_to_32_bit,
528	//	FCVTZS	vfcvtzsscalar_integer_Double_precision_to_32_bit,
529	//	FCVTZU	vfcvtzuscalar_integer_Double_precision_to_32_bit,
530	//	FCVTNS	vfcvtnsscalar_Single_precision_to_64_bit,
531	//	FCVTNU	vfcvtnuscalar_Single_precision_to_64_bit,
532	//	SCVTF	vscvtfscalar_integer_64_bit_to_single_precision,
533	//	UCVTF	vucvtfscalar_integer_64_bit_to_single_precision,
534	//	FCVTAS	vfcvtasscalar_Single_precision_to_64_bit,
535	//	FCVTAU	vfcvtauscalar_Single_precision_to_64_bit,
536	//	FCVTPS	vfcvtpsscalar_Single_precision_to_64_bit,
537	//	FCVTPU	vfcvtpuscalar_Single_precision_to_64_bit,
538	//	FCVTMS	vfcvtmsscalar_Single_precision_to_64_bit,
539	//	FCVTMU	vfcvtmuscalar_Single_precision_to_64_bit,
540	//	FCVTZS	vfcvtzsscalar_integer_Single_precision_to_64_bit,
541	//	FCVTZU	vfcvtzuscalar_integer_Single_precision_to_64_bit,
542	//	FCVTNS	vfcvtnsscalar_Double_precision_to_64_bit,
543	//	FCVTNU	vfcvtnuscalar_Double_precision_to_64_bit,
544	//	SCVTF	vscvtfscalar_integer_64_bit_to_double_precision,
545	//	UCVTF	vucvtfscalar_integer_64_bit_to_double_precision,
546	//	FCVTAS	vfcvtasscalar_Double_precision_to_64_bit,
547	//	FCVTAU	vfcvtauscalar_Double_precision_to_64_bit,
548	//	FMOV	vfmovgeneral_Double_precision_to_64_bit,
549	//	FMOV	vfmovgeneral_64_bit_to_double_precision,

1	in_use	Opcode	NAME
550	//	FCVTPS	vfcvtpsscalar_Double_precision_to_64_bit,
551	//	FCVTPU	vfcvtpuscalar_Double_precision_to_64_bit,
552	//	FCVTMS	vfcvtmsscalar_Double_precision_to_64_bit,
553	//	FCVTMU	vfcvtmuscalar_Double_precision_to_64_bit,
554	//	FCVTZS	vfcvtzsscalar_integer_Double_precision_to_64_bit,
555	//	FCVTZU	vfcvtzuscalar_integer_Double_precision_to_64_bit,
556	//	FMOV	vfmovgeneral_Top_half_of_128_bit_to_64_bit,
557	//	FMOV	vfmovgeneral_64_bit_to_top_half_of_128_bit,
558	//	**Floating-point data-processin**	
559	//	FMADD	vfmaddSingle_precision,
560	//	FMSUB	vfmsubSingle_precision,
561	//	FNMADD	vfnmaddSingle_precision,
562	//	FNMSUB	vfnmsubSingle_precision,
563	//	FMADD	vfmaddDouble_precision,
564	//	FMSUB	vfmsubDouble_precision,
565	//	FNMADD	vfnmaddDouble_precision,
566	//	FNMSUB	vfnmsubDouble_precision,
567	//	**AdvSIMD scalar three same**	
568	//	SQADD	vsqaddScalar,
569	//	SQSUB	vsqsubScalar,
570	//	CMGT	vcmgtregister_Scalar,
571	//	CMGE	vcmgeregister_Scalar,
572	//	SSHL	vsshlScalar,
573	//	SQSHL	vsqshlregister_Scalar,
574	//	SRSHL	vsrshlScalar,
575	//	SQRSHL	vsqrshlScalar,
576	//	ADD	vaddvector_Scalar,
577	//	CMTST	vcmtstScalar,
578	//	SQDMULH	vsqdmulhvector_Scalar,
579	//	FMULX	vfmulxScalar,
580	//	FCMEQ	vfcmeqregister_Scalar,
581	//	FRECPS	vfrecpsScalar,
582	//	FRSQRTS	vfrsqrtsScalar,
583	//	UQADD	vuqaddScalar,

1	in_use	Opcode	NAME
584	//	UQSUB	vuqsubScalar,
585	//	CMHI	vcmhiregister_Scalar,
586	//	CMHS	vcmhsregister_Scalar,
587	//	USHL	vushlScalar,
588	//	UQSHL	vuqshlregister_Scalar,
589	//	URSHL	vurshlScalar,
590	//	UQRSHL	vuqrshlScalar,
591	//	SUB	vsubvector_Scalar,
592	//	CMEQ	vcmeqregister_Scalar,
593	//	SQRDMULH	vsqrdmulhvector_Scalar,
594	//	FCMGE	vfcmgeregister_Scalar,
595	//	FACGE	vfacgeScalar,
596	//	FABD	vfabdScalar,
597	//	FCMGT	vfcmgtregister_Scalar,
598	//	FACGT	vfacgtScalar,
599	//	**AdvSIMD scalar three differer**	
600	//	SQDMLAL	vsqdmlalvector_Scalar,
601	//	SQDMLAL2	vsqdmlal2vector_Scalar,
602	//	SQDMLSL	vsqdmlslvector_Scalar,
603	//	SQDMLSL2	vsqdmlsl2vector_Scalar,
604	//	SQDMULL	vsqdmullvector_Scalar,
605	//	SQDMULL2	vsqdmull2vector_Scalar,
606	//	**AdvSIMD scalar two-reg misc**	
607	//	SUQADD	vsuqaddScalar,
608	//	SQABS	vsqabsScalar,
609	//	CMGT	vcmgtzero_Scalar,
610	//	CMEQ	vcmeqzero_Scalar,
611	//	CMLT	vcmltzero_Scalar,
612	//	ABS	vabsScalar,
613	//	SQXTN	vsqxtnScalar,
614	//	SQXTN2	vsqxtn2Scalar,
615	//	FCVTNS	vfcvtnsvector_Scalar,
616	//	FCVTMS	vfcvtmsvector_Scalar,
617	//	FCVTAS	vfcvtasvector_Scalar,

1	in_use	Opcode	NAME
618	//	SCVTF	vscvtfvector_integer_Scalar,
619	//	FCMGT	vfcmgtzero_Scalar,
620	//	FCMEQ	vfcmeqzero_Scalar,
621	//	FCMLT	vfcmltzero_Scalar,
622	//	FCVTPS	vfcvtpsvector_Scalar,
623	//	FCVTZS	vfcvtzsvector_integer_Scalar,
624	//	FRECPE	vfrecpeScalar,
625	//	FRECPX	frecpx,
626	//	USQADD	vusqaddScalar,
627	//	SQNEG	vsqnegScalar,
628	//	CMGE	vcmgezero_Scalar,
629	//	CMLE	vcmlezero_Scalar,
630	//	NEG	vnegvector_Scalar,
631	//	SQXTUN	vsqxtunScalar,
632	//	SQXTUN2	vsqxtun2Scalar,
633	//	UQXTN	vuqxtnScalar,
634	//	UQXTN2	vuqxtn2Scalar,
635	//	FCVTXN	vfcvtxnScalar,
636	//	FCVTXN2	vfcvtxn2Scalar,
637	//	FCVTNU	vfcvtnuvector_Scalar,
638	//	FCVTMU	vfcvtmuvector_Scalar,
639	//	FCVTAU	vfcvtauvector_Scalar,
640	//	UCVTF	vucvtfvector_integer_Scalar,
641	//	FCMGE	vfcmgezero_Scalar,
642	//	FCMLE	vfcmlezero_Scalar,
643	//	FCVTPU	vfcvtpuvector_Scalar,
644	//	FCVTZU	vfcvtzuvector_integer_Scalar,
645	//	FRSQRTE	vfrsqrteScalar,
646	//	**AdvSIMD scalar pairwise**	
647	//	ADDP	addpscalar,
648	//	FMAXNMP	fmaxnmpscalar,
649	//	FADDP	faddpscalar,
650	//	FMAXP	fmaxpscalar,
651	//	FMINNMP	fminnmpscalar,

1	in_use	Opcode	NAME
652	//	FMINP	fminpscalar,
653	//	**AdvSIMD scalar copy**	
654	//	DUP	vdupelement_Scalar,
655	//	**AdvSIMD scalar x indexed ele**	
656	//	SQDMLAL	vsqdmlalby_element_Scalar,
657	//	SQDMLAL2	vsqdmlal2by_element_Scalar,
658	//	SQDMLSL	vsqdmlslby_element_Scalar,
659	//	SQDMLSL2	vsqdmlsl2by_element_Scalar,
660	//	SQDMULL	vsqdmullby_element_Scalar,
661	//	SQDMULL2	vsqdmull2by_element_Scalar,
662	//	SQDMULH	vsqdmulhby_element_Scalar,
663	//	SQRDMULH	vsqrdmulhby_element_Scalar,
664	//	FMLA	vfmlaby_element_Scalar,
665	//	FMLS	vfmlsby_element_Scalar,
666	//	FMUL	vfmulby_element_Scalar,
667	//	FMULX	vfmulxby_element_Scalar,
668	//	**AdvSIMD scalar shift by imme**	
669	//	SSHR	vsshrScalar,
670	//	SSRA	vssraScalar,
671	//	SRSHR	vsrshrScalar,
672	//	SRSRA	vsrsraScalar,
673	//	SHL	vshlScalar,
674	//	SQSHL	vsqshlimmediate_Scalar,
675	//	SQSHRN	vsqshrnScalar,
676	//	SQSHRN2	vsqshrn2Scalar,
677	//	SQRSHRN	vsqrshrnScalar,
678	//	SQRSHRN2	vsqrshrn2Scalar,
679	//	SCVTF	vscvtfvector_fixed_point_Scalar,
680	//	FCVTZS	vfcvtzsvector_fixed_point_Scalar,
681	//	USHR	vushrScalar,
682	//	USRA	vusraScalar,
683	//	URSHR	vurshrScalar,
684	//	URSRA	vursraScalar,
685	//	SRI	vsriScalar,

1	in_use	Opcode	NAME
686	//	SLI	vsliScalar,
687	//	SQSHLU	vsqshluScalar,
688	//	UQSHL	vuqshlimmediate_Scalar,
689	//	SQSHRUN	vsqshrunScalar,
690	//	SQSHRUN2	vsqshrun2Scalar,
691	//	SQRSHRUN	vsqrshrunScalar,
692	//	SQRSHRUN2	vsqrshrun2Scalar,
693	//	UQSHRN	vuqshrnScalar,
694	//	UQRSHRN	vuqrshrnScalar,
695	//	UQRSHRN2	vuqrshrn2Scalar,
696	//	UCVTF	vucvtfvector_fixed_point_Scalar,
697	//	FCVTZU	vfcvtzuvector_fixed_point_Scalar,
698	//	**Crypto three-reg SHA**	
699	//	SHA1C	sha1c,
700	//	SHA1P	sha1p,
701	//	SHA1M	sha1m,
702	//	SHA1SU0	sha1su0,
703	//	SHA256H	sha256h,
704	//	SHA256H2	sha256h2,
705	//	SHA256SU1	sha256su1,
706	//	**Crypto two-reg SHA**	
707	//	SHA1H	sha1h,
708	//	SHA1SU1	sha1su1,
709	//	SHA256SU0	sha256su0,
710	//	**Crypto AES**	
711	//	AESE	aese,
712	//	AESD	aesd,
713	//	AESMC	aesmc,
714	//	AESIMC	aesimc,
715	//	**AdvSIMD three same**	
716	//	SHADD	shadd,
717	//	SQADD	vsqaddVector,
718	//	SRHADD	srhadd,
719	//	SHSUB	shsub,

1	in_use	Opcode	NAME
720	//	SQSUB	vsqsubVector,
721	//	CMGT	vcmgtregister_Vector,
722	//	CMGE	vcmgeregister_Vector,
723	//	SSHL Vector	sshl vector,
724	//	SQSHL	vsqshlregister_Vector,
725	//	SRSHL	vsrshlVector,
726	//	SQRSHL	vsqrshlVector,
727	//	SMAX	smax,
728	//	SMIN	smin,
729	//	SABD	sabd,
730	//	SABA	saba,
731	//	ADD	vaddvector_Vector,
732	//	CMTST	vcmtstVector,
733	//	MLA	mlavector,
734	//	MUL	mulvector,
735	//	SMAXP	smaxp,
736	//	SMINP	sminp,
737	//	SQDMULH	vsqdmulhvector_Vector,
738	//	ADDP	addpvector,
739	//	FMAXNM	fmaxnmvector,
740	//	FMLA	fmlavector,
741	//	FADD	faddvector,
742	//	FMULX	vfmulxVector,
743	//	FCMEQ	vfcmeqregister_Vector,
744	//	FMAX	fmaxvector,
745	//	FRECPS	vfrecpsVector,
746	//	AND	andvector,
747	//	BIC	bicvector_register,
748	//	FMINNM	fminnmvector,
749	//	FMLS	fmlsvector,
750	//	FSUB	fsubvector,
751	//	FMIN	fminvector,
752	//	FRSQRTS	vfrsqrtsVector,
753	//	ORR	orrvector_register,

1	in_use	Opcode	NAME
754	//	ORN	ornvector,
755	//	UHADD	uhadd,
756	//	UQADD	vuqaddVector,
757	//	URHADD	urhadd,
758	//	UHSUB	uhsub,
759	//	UQSUB	vuqsubVector,
760	//	CMHI	vcmhiregister_Vector,
761	//	CMHS	vcmhsregister_Vector,
762	//	USHL	vushlVector,
763	//	UQSHL	vuqshlregister_Vector,
764	//	URSHL	vurshlVector,
765	//	UQRSHL	vuqrshlVector,
766	//	UMAX	umax,
767	//	UMIN	umin,
768	//	UABD	uabd,
769	//	UABA	uaba,
770	//	SUB	vsubvector_Vector,
771	//	CMEQ	vcmeqregister_Vector,
772	//	MLS	mlsvector,
773	//	PMUL	pmul,
774	//	UMAXP	umaxp,
775	//	UMINP	uminp,
776	//	SQRDMULH	vsqrdmulhvector_Vector,
777	//	FMAXNMP	fmaxnmpvector,
778	//	FADDP	faddpvector,
779	//	FMUL	fmulvector,
780	//	FCMGE	vfcmgeregister_Vector,
781	//	FACGE	vfacgeVector,
782	//	FMAXP	fmaxpvector,
783	//	FDIV	fdivvector,
784	//	EOR	eorvector,
785	//	BSL	bsl,
786	//	FMINNMP	fminnmpvector,
787	//	FABD	vfabdVector,

1	in_use	Opcode	NAME
788	//	FCMGT	vfcmgtregister_Vector,
789	//	FACGT	vfacgtVector,
790	//	FMINP	fminpvector,
791	//	BIT	bit,
792	//	BIF	bif,
793	//	**AdvSIMD three different**	
794	//	SADDL	saddl,
795	//	SADDL2	saddl2,
796	//	SADDW	saddw,
797	//	SADDW2	saddw2,
798	//	SSUBL	ssubl,
799	//	SSUBL2	ssubl2,
800	//	SSUBW	ssubw,
801	//	SSUBW2	ssubw2,
802	//	ADDHN	addhn,
803	//	ADDHN2	addhn2,
804	//	SABAL	sabal,
805	//	SABAL2	sabal2,
806	//	SUBHN	subhn,
807	//	SUBHN2	subhn2,
808	//	SABDL	sabdl,
809	//	SABDL2	sabdl2,
810	//	SMLAL	smlalvector,
811	//	SMLAL2	smlal2vector,
812	//	SQDMLAL	vsqdmlalvector_Vector,
813	//	SQDMLAL2	vsqdmlal2vector_Vector,
814	//	SMLSL	smlslvector,
815	//	SMLSL2	smlsl2vector,
816	//	SQDMLSL	vsqdmlslvector_Vector,
817	//	SQDMLSL2	vsqdmlsl2vector_Vector,
818	//	SMULL	smullvector,
819	//	SMULL2	smull2vector,
820	//	SQDMULL	vsqdmullvector_Vector,
821	//	SQDMULL2	vsqdmull2vector_Vector,

1	in_use	Opcode	NAME
822	//	PMULL	pmull,
823	//	PMULL2	pmull2,
824	//	UADDL	uaddl,
825	//	UADDL2	uaddl2,
826	//	UADDW	uaddw,
827	//	UADDW2	uaddw2,
828	//	USUBL	usubl,
829	//	USUBL2	usubl2,
830	//	USUBW	usubw,
831	//	USUBW2	usubw2,
832	//	RADDHN	raddhn,
833	//	RADDHN2	raddhn2,
834	//	UABAL	uabal,
835	//	UABAL2	uabal2,
836	//	RSUBHN	rsubhn,
837	//	RSUBHN2	rsubhn2,
838	//	UABDL	uabdl,
839	//	UABDL2	uabdl2,
840	//	UMLAL	umlalvector,
841	//	UMLAL2	umlal2vector,
842	//	UMLSL	umlslvector,
843	//	UMLSL2	umlsl2vector,
844	//	UMULL	umullvector,
845	//	UMULL2	umull2vector,
846	//	**AdvSIMD two-reg misc**	
847	//	REV64	rev64,
848	//	REV16	rev16vector,
849	//	SADDLP	saddlp,
850	//	SUQADD	vsuqaddVector,
851	//	CLS	clsvector,
852	//	CNT	cnt,
853	//	SADALP	sadalp,
854	//	SQABS	vsqabsVector,
855	//	CMGT	vcmgtzero_Vector,

1	in_use	Opcode	NAME
856	//	CMEQ	vcmeqzero_Vector,
857	//	CMLT	vcmltzero_Vector,
858	//	ABS	vabsVector,
859	//	XTN	xtn,
860	//	XTN2	xtn2,
861	//	SQXTN	vsqxtnVector,
862	//	SQXTN2	vsqxtn2Vector,
863	//	FCVTN	fcvtn,
864	//	FCVTN2	fcvtn2,
865	//	FCVTL	fcvtl,
866	//	FCVTL2	fcvtl2,
867	//	FRINTN	frintnvector,
868	//	FRINTM	frintmvector,
869	//	FCVTNS	vfcvtnsvector_Vector,
870	//	FCVTMS	vfcvtmsvector_Vector,
871	//	FCVTAS	vfcvtasvector_Vector,
872	//	SCVTF	vscvtfvector_integer_Vector,
873	//	FCMGT	vfcmgtzero_Vector,
874	//	FCMEQ	vfcmeqzero_Vector,
875	//	FCMLT	vfcmltzero_Vector,
876	//	FABS	fabsvector,
877	//	FRINTP	frintpvector,
878	//	FRINTZ	frintzvector,
879	//	FCVTPS	vfcvtpsvector_Vector,
880	//	FCVTZS	vfcvtzsvector_integer_Vector,
881	//	URECPE	urecpe,
882	//	FRECPE	vfrecpeVector,
883	//	REV32	rev32vector,
884	//	UADDLP	uaddlp,
885	//	USQADD	vusqaddVector,
886	//	CLZ	clzvector,
887	//	UADALP	uadalp,
888	//	SQNEG	vsqnegVector,
889	//	CMGE	vcmgezero_Vector,

1	in_use	Opcode	NAME
890	//	CMLE	vcmlezero_Vector,
891	//	NEG	vnegvector_Vector,
892	//	SQXTUN	vsqxtunVector,
893	//	SQXTUN2	vsqxtun2Vector,
894	//	SHLL	shll,
895	//	SHLL2	shll2,
896	//	UQXTN	vuqxtnVector,
897	//	UQXTN2	vuqxtn2Vector,
898	//	FCVTXN	vfcvtxnVector,
899	//	FCVTXN2	vfcvtxn2Vector,
900	//	FRINTA	frintavector,
901	//	FRINTX	frintxvector,
902	//	FCVTNU	vfcvtnuvector_Vector,
903	//	FCVTMU	vfcvtmuvector_Vector,
904	//	FCVTAU	vfcvtauvector_Vector,
905	//	UCVTF	vucvtfvector_integer_Vector,
906	//	NOT	not,
907	//	RBIT	rbitvector,
908	//	FCMGE	vfcmgezero_Vector,
909	//	FCMLE	vfcmlezero_Vector,
910	//	FNEG	fnegvector,
911	//	FRINTI	frintivector,
912	//	FCVTPU	vfcvtpuvector_Vector,
913	//	FCVTZU	vfcvtzuvector_integer_Vector,
914	//	URSQRTE	ursqrte,
915	//	FRSQRTE	vfrsqrteVector,
916	//	FSQRT	fsqrtvector,
917	//	**AdvSIMD across lanes**	
918	//	SADDLV	saddlv,
919	//	SMAXV	smaxv,
920	//	SMINV	sminv,
921	//	ADDV	addv,
922	//	UADDLV	uaddlv,
923	//	UMAXV	umaxv,

1	in_use	Opcode	NAME
924	//	UMINV	uminv,
925	//	FMAXNMV	fmaxnmv,
926	//	FMAXV	fmaxv,
927	//	FMINNMV	fminnmv,
928	//	FMINV	fminv,
929	//	**AdvSIMD copy**	
930	//	DUP	vdupelement_Vector,
931	//	DUP	dupgeneral,
932	//	SMOV	vsmov32_bit,
933	//	UMOV	vumov32_bit,
934	//	INS	insgeneral,
935	//	SMOV	vsmov64_bit,
936	//	UMOV	vumov64_bit,
937	//	INS	inselement,
938	//	**AdvSIMD vector x indexed ele**	
939	//	SMLAL	smlalby_element,
940	//	SMLAL2	smlal2by_element,
941	//	SQDMLAL	vsqdmlalby_element_Vector,
942	//	SQDMLAL2	vsqdmlal2by_element_Vector,
943	//	SMLSL	smlslby_element,
944	//	SMLSL2	smlsl2by_element,
945	//	SQDMLSL	vsqdmlslby_element_Vector,
946	//	SQDMLSL2	vsqdmlsl2by_element_Vector,
947	//	MUL	mulby_element,
948	//	SMULL	smullby_element,
949	//	SMULL2	smull2by_element,
950	//	SQDMULL	vsqdmullby_element_Vector,
951	//	SQDMULL2	vsqdmull2by_element_Vector,
952	//	SQDMULH	vsqdmulhby_element_Vector,
953	//	SQRDMULH	vsqrdmulhby_element_Vector,
954	//	FMLA	vfmlaby_element_Vector,
955	//	FMLS	vfmlsby_element_Vector,
956	//	FMUL	vfmulby_element_Vector,
957	//	MLA	mlaby_element,

1	in_use	Opcode	NAME
958	//	UMLAL	umlalby_element,
959	//	UMLAL2	umlal2by_element,
960	//	MLS	mlsby_element,
961	//	UMLSL	umlslby_element,
962	//	UMLSL2	umlsl2by_element,
963	//	UMULL	umullby_element,
964	//	UMULL2	umull2by_element,
965	//	FMULX	vfmulxby_element_Vector,
966	//	**AdvSIMD modified immediate**	
967	//	MOVI	vmovi32_bit_shifted_immediate,
968	//	ORR	vorrvector_immediate_32_bit,
969	//	MOVI	vmovi16_bit_shifted_immediate,
970	//	ORR	vorrvector_immediate_16_bit,
971	//	MOVI	vmovi32_bit_shifting_ones,
972	//	MOVI	vmovi8_bit,
973	//	FMOV	vfmovvector_immediate_Single_precision,
974	//	MVNI	vmvni32_bit_shifted_immediate,
975	//	BIC	vbicvector_immediate_32_bit,
976	//	MVNI	vmvni16_bit_shifted_immediate,
977	//	BIC	vbicvector_immediate_16_bit,
978	//	MVNI	vmvni32_bit_shifting_ones,
979	//	MOVI	vmovi64_bit_scalar,
980	//	MOVI	vmovi64_bit_vector,
981	//	FMOV	vfmovvector_immediate_Double_precision,
982	//	**AdvSIMD shift by immediate**	
983	//	SSHR	vsshrVector,
984	//	SSRA	vssraVector,
985	//	SRSHR	vsrshrVector,
986	//	SRSRA	vsrsraVector,
987	//	SHL	vshlVector,
988	//	SQSHL	vsqshlimmediate_Vector,
989	//	SHRN	shrn,
990	//	SHRN2	shrn2,
991	//	RSHRN	rshrn,

1	in_use	Opcode	NAME
992	//	RSHRN2	rshrn2,
993	//	SQSHRN	vsqshrnVector,
994	//	SQSHRN2	vsqshrn2Vector,
995	//	SQRSHRN	vsqrshrnVector,
996	//	SQRSHRN2	vsqrshrn2Vector,
997	//	SSHLL	sshll,
998	//	SSHLL2	sshll2,
999	//	SCVTF	vscvtfvector_fixed_point_Vector,
1000	//	FCVTZS	vfcvtzsvector_fixed_point_Vector,
1001	//	USHR	vushrVector,
1002	//	USRA	vusraVector,
1003	//	URSHR	vurshrVector,
1004	//	URSRA	vursraVector,
1005	//	SRI	vsriVector,
1006	//	SLI	vsliVector,
1007	//	SQSHLU	vsqshluVector,
1008	//	UQSHL	vuqshlimmediate_Vector,
1009	//	SQSHRUN	vsqshrunVector,
1010	//	SQSHRUN2	vsqshrun2Vector,
1011	//	SQRSHRUN	vsqrshrunVector,
1012	//	SQRSHRUN2	vsqrshrun2Vector,
1013	//	UQSHRN	vuqshrnVector,
1014	//	UQRSHRN	vuqrshrnVector,
1015	//	UQRSHRN2	vuqrshrn2Vector,
1016	//	USHLL	ushll,
1017	//	USHLL2	ushll2,
1018	//	UCVTF	vucvtfvector_fixed_point_Vector,
1019	//	FCVTZU	vfcvtzuvector_fixed_point_Vector,
1020	//	**AdvSIMD TBL/TBX**	
1021	//	TBL	vtblSingle_register_table,
1022	//	TBX	vtbxSingle_register_table,
1023	//	TBL	vtblTwo_register_table,
1024	//	TBX	vtbxTwo_register_table,
1025	//	TBL	vtblThree_register_table,

```
1    in_use   Opcode                NAME
1026 //       TBX                   vtbxThree_register_table,
1027 //       TBL                   vtblFour_register_table,
1028 //       TBX                   vtbxFour_register_table,
1029 //     AdvSIMD ZIP/UZP/TRN
1030 //       UZP1                  uzp1,
1031 //       TRN1                  trn1,
1032 //       ZIP1                  zip1,
1033 //       UZP2                  uzp2,
1034 //       TRN2                  trn2,
1035 //       ZIP2                  zip2,
1036 //     AdvSIMD EXT
1037 //       EXT                   ext,
1038 //   Loads and stores
1039 //     AdvSIMD load/store multiple s
1040 //       ST4                   vst4multiple_structures_No_offset,
1041 //       ST1                   vst1multiple_structures_Four_registers,
1042 //       ST3                   vst3multiple_structures_No_offset,
1043 //       ST1                   vst1multiple_structures_Three_registers,
1044 //       ST1                   vst1multiple_structures_One_register,
1045 //       ST2                   vst2multiple_structures_No_offset,
1046 //       ST1                   vst1multiple_structures_Two_registers,
1047 //       LD4                   vld4multiple_structures_No_offset,
1048 //       LD1                   vld1multiple_structures_Four_registers,
1049 //       LD3                   vld3multiple_structures_No_offset,
1050 //       LD1                   vld1multiple_structures_Three_registers,
1051 //       LD1                   vld1multiple_structures_One_register,
1052 //       LD2                   vld2multiple_structures_No_offset,
1053 //       LD1                   vld1multiple_structures_Two_registers,
1054 //     AdvSIMD load/store multiple s
1055 //       ST4                   vst4multiple_structures_Register_offset,
1056 //       ST1                   vst1multiple_structures_Four_registers_register_offset,
1057 //       ST3                   vst3multiple_structures_Register_offset,
1058 //       ST1                   vst1multiple_structures_Three_registers_register_offset,
1059 //       ST1                   vst1multiple_structures_One_register_register_offset,
```

1	in_use	Opcode	NAME
1060	//	ST2	vst2multiple_structures_Register_offset,
1061	//	ST1	vst1multiple_structures_Two_registers_register_offset,
1062	//	ST4	vst4multiple_structures_Immediate_offset,
1063	//	ST1	vst1multiple_structures_Four_registers_immediate_offset,
1064	//	ST3	vst3multiple_structures_Immediate_offset,
1065	//	ST1	vst1multiple_structures_Three_registers_immediate_offset
1066	//	ST1	vst1multiple_structures_One_register_immediate_offset,
1067	//	ST2	vst2multiple_structures_Immediate_offset,
1068	//	ST1	vst1multiple_structures_Two_registers_immediate_offset,
1069	//	LD4	vld4multiple_structures_Register_offset,
1070	//	LD1	vld1multiple_structures_Four_registers_register_offset,
1071	//	LD3	vld3multiple_structures_Register_offset,
1072	//	LD1	vld1multiple_structures_Three_registers_register_offset,
1073	//	LD1	vld1multiple_structures_One_register_register_offset,
1074	//	LD2	vld2multiple_structures_Register_offset,
1075	//	LD1	vld1multiple_structures_Two_registers_register_offset,
1076	//	LD4	vld4multiple_structures_Immediate_offset,
1077	//	LD1	vld1multiple_structures_Four_registers_immediate_offset,
1078	//	LD3	vld3multiple_structures_Immediate_offset,
1079	//	LD1	vld1multiple_structures_Three_registers_immediate_offset
1080	//	LD1	vld1multiple_structures_One_register_immediate_offset,
1081	//	LD2	vld2multiple_structures_Immediate_offset,
1082	//	LD1	vld1multiple_structures_Two_registers_immediate_offset,
1083	//	**AdvSIMD load/store single st**	
1084	//	ST1	vst1single_structure_8_bit,
1085	//	ST3	vst3single_structure_8_bit,
1086	//	ST1	vst1single_structure_16_bit,
1087	//	ST3	vst3single_structure_16_bit,
1088	//	ST1	vst1single_structure_32_bit,
1089	//	ST1	vst1single_structure_64_bit,
1090	//	ST3	vst3single_structure_32_bit,
1091	//	ST3	vst3single_structure_64_bit,
1092	//	ST2	vst2single_structure_8_bit,
1093	//	ST4	vst4single_structure_8_bit,

1	in_use	Opcode	NAME
1094	//	ST2	vst2single_structure_16_bit,
1095	//	ST4	vst4single_structure_16_bit,
1096	//	ST2	vst2single_structure_32_bit,
1097	//	ST2	vst2single_structure_64_bit,
1098	//	ST4	vst4single_structure_32_bit,
1099	//	ST4	vst4single_structure_64_bit,
1100	//	LD1	vld1single_structure_8_bit,
1101	//	LD3	vld3single_structure_8_bit,
1102	//	LD1	vld1single_structure_16_bit,
1103	//	LD3	vld3single_structure_16_bit,
1104	//	LD1	vld1single_structure_32_bit,
1105	//	LD1	vld1single_structure_64_bit,
1106	//	LD3	vld3single_structure_32_bit,
1107	//	LD3	vld3single_structure_64_bit,
1108	//	LD1R	vld1rNo_offset,
1109	//	LD3R	vld3rNo_offset,
1110	//	LD2	vld2single_structure_8_bit,
1111	//	LD4	vld4single_structure_8_bit,
1112	//	LD2	vld2single_structure_16_bit,
1113	//	LD4	vld4single_structure_16_bit,
1114	//	LD2	vld2single_structure_32_bit,
1115	//	LD2	vld2single_structure_64_bit,
1116	//	LD4	vld4single_structure_32_bit,
1117	//	LD4	vld4single_structure_64_bit,
1118	//	LD2R	vld2rNo_offset,
1119	//	LD4R	vld4rNo_offset,
1120	//	**AdvSIMD load/store single st**	
1121	//	ST1	vst1single_structure_8_bit_register_offset,
1122	//	ST3	vst3single_structure_8_bit_register_offset,
1123	//	ST1	vst1single_structure_16_bit_register_offset,
1124	//	ST3	vst3single_structure_16_bit_register_offset,
1125	//	ST1	vst1single_structure_32_bit_register_offset,
1126	//	ST1	vst1single_structure_64_bit_register_offset,
1127	//	ST3	vst3single_structure_32_bit_register_offset,

1	in_use	Opcode	NAME
1128	//	ST3	vst3single_structure_64_bit_register_offset,
1129	//	ST1	vst1single_structure_8_bit_immediate_offset,
1130	//	ST3	vst3single_structure_8_bit_immediate_offset,
1131	//	ST1	vst1single_structure_16_bit_immediate_offset,
1132	//	ST3	vst3single_structure_16_bit_immediate_offset,
1133	//	ST1	vst1single_structure_32_bit_immediate_offset,
1134	//	ST1	vst1single_structure_64_bit_immediate_offset,
1135	//	ST3	vst3single_structure_32_bit_immediate_offset,
1136	//	ST3	vst3single_structure_64_bit_immediate_offset,
1137	//	ST2	vst2single_structure_8_bit_register_offset,
1138	//	ST4	vst4single_structure_8_bit_register_offset,
1139	//	ST2	vst2single_structure_16_bit_register_offset,
1140	//	ST4	vst4single_structure_16_bit_register_offset,
1141	//	ST2	vst2single_structure_32_bit_register_offset,
1142	//	ST2	vst2single_structure_64_bit_register_offset,
1143	//	ST4	vst4single_structure_32_bit_register_offset,
1144	//	ST4	vst4single_structure_64_bit_register_offset,
1145	//	ST2	vst2single_structure_8_bit_immediate_offset,
1146	//	ST4	vst4single_structure_8_bit_immediate_offset,
1147	//	ST2	vst2single_structure_16_bit_immediate_offset,
1148	//	ST4	vst4single_structure_16_bit_immediate_offset,
1149	//	ST2	vst2single_structure_32_bit_immediate_offset,
1150	//	ST2	vst2single_structure_64_bit_immediate_offset,
1151	//	ST4	vst4single_structure_32_bit_immediate_offset,
1152	//	ST4	vst4single_structure_64_bit_immediate_offset,
1153	//	LD1	vld1single_structure_8_bit_register_offset,
1154	//	LD3	vld3single_structure_8_bit_register_offset,
1155	//	LD1	vld1single_structure_16_bit_register_offset,
1156	//	LD3	vld3single_structure_16_bit_register_offset,
1157	//	LD1	vld1single_structure_32_bit_register_offset,
1158	//	LD1	vld1single_structure_64_bit_register_offset,
1159	//	LD3	vld3single_structure_32_bit_register_offset,
1160	//	LD3	vld3single_structure_64_bit_register_offset,
1161	//	LD1R	vld1rRegister_offset,

1	in_use	Opcode	NAME
1162	//	LD3R	vld3rRegister_offset,
1163	//	LD1	vld1single_structure_8_bit_immediate_offset,
1164	//	LD3	vld3single_structure_8_bit_immediate_offset,
1165	//	LD1	vld1single_structure_16_bit_immediate_offset,
1166	//	LD3	vld3single_structure_16_bit_immediate_offset,
1167	//	LD1	vld1single_structure_32_bit_immediate_offset,
1168	//	LD1	vld1single_structure_64_bit_immediate_offset,
1169	//	LD3	vld3single_structure_32_bit_immediate_offset,
1170	//	LD3	vld3single_structure_64_bit_immediate_offset,
1171	//	LD1R	vld1rImmediate_offset,
1172	//	LD3R	vld3rImmediate_offset,
1173	//	LD2	vld2single_structure_8_bit_register_offset,
1174	//	LD4	vld4single_structure_8_bit_register_offset,
1175	//	LD2	vld2single_structure_16_bit_register_offset,
1176	//	LD4	vld4single_structure_16_bit_register_offset,
1177	//	LD2	vld2single_structure_32_bit_register_offset,
1178	//	LD2	vld2single_structure_64_bit_register_offset,
1179	//	LD4	vld4single_structure_32_bit_register_offset,
1180	//	LD4	vld4single_structure_64_bit_register_offset,
1181	//	LD2R	vld2rRegister_offset,
1182	//	LD4R	vld4rRegister_offset,
1183	//	LD2	vld2single_structure_8_bit_immediate_offset,
1184	//	LD4	vld4single_structure_8_bit_immediate_offset,
1185	//	LD2	vld2single_structure_16_bit_immediate_offset,
1186	//	LD4	vld4single_structure_16_bit_immediate_offset,
1187	//	LD2	vld2single_structure_32_bit_immediate_offset,
1188	//	LD2	vld2single_structure_64_bit_immediate_offset,
1189	//	LD4	vld4single_structure_32_bit_immediate_offset,
1190	//	LD4	vld4single_structure_64_bit_immediate_offset,
1191	//	LD2R	vld2rImmediate_offset,
1192	//	LD4R	vld4rImmediate_offset,

	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
2		**UNALLOCATED**	/* UNALLOCATED */			
3		BAD	bad,	/* 0x00000000	BAD	invalid operation */
4		**Branch,exception generatic**	/* Branch,exception generation and system Instruction */			
5		**Compare _ Branch (immediat**	/* Compare _ Branch (immediate) */			
6		CBZ	cbzw,	/* 0x34000000	CBZ	*/
7		CBNZ	cbnzw,	/* 0x35000000	CBNZ	*/
8		CBZ	cbzx,	/* 0xB4000000	CBZ	*/
9		CBNZ	cbnzx,	/* 0xB5000000	CBNZ	*/
10		**Test bit & branch (immediate)**	/* Test bit & branch (immediate) */			
11		TBZ	tbz,	/* 0x36000000	TBZ	*/
12		TBNZ	tbnz,	/* 0x37000000	TBNZ	*/
13		**Conditional branch (immediat**	/* Conditional branch (immediate) */			
14		B_cond	b_cond,	/* 0x54000000	B_cond	*/
15		**Exception generation**	/* Exception generation */			
16	//	SVC	//svc,	/* 0xD4000001	SVC	*/
17	//	HVC	//hvc,	/* 0xD4000002	HVC	*/
18	//	SMC	//smc,	/* 0xD4000003	SMC	*/
19		BRK	brkarm64,	/* 0xD4200000	BRK	AArch64 Specific BRK */
20	//	HLT	//hlt,	/* 0xD4400000	HLT	*/
21	//	DCPS1	//dcps1,	/* 0xD4A00001	DCPS1	*/
22	//	DCPS2	//dcps2,	/* 0xD4A00002	DCPS2	*/
23	//	DCPS3	//dcps3,	/* 0xD4A00003	DCPS3	*/
24	//	**System**	/* System */			
25	//	MSR	//msrimm,	/* 0xD500401F	MSR	*/
26	//	HINT	//hint,	/* 0xD503201F	HINT	*/
27	//	CLREX	//clrex,	/* 0xD503305F	CLREX	*/
28	//	DSB	//dsb,	/* 0xD503309F	DSB	*/
29	//	DMB	//dmb,	/* 0xD50330BF	DMB	*/
30	//	ISB	//isb,	/* 0xD50330DF	ISB	*/
31	//	SYS	//sys,	/* 0xD5080000	SYS	*/
32	//	MSR	//msr,	/* 0xD5100000	MSR	*/
33	//	SYSL	//sysl,	/* 0xD5280000	SYSL	*/
34	//	MRS	//mrs,	/* 0xD5300000	MRS	*/
35		**Unconditional branch (registe**	/* Unconditional branch (register) */			
36		BR	br,	/* 0xD61F0000	BR	*/
37		BLR	blr,	/* 0xD63F0000	BLR	*/

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
38		RET	ret,	/* 0xD65F0000RET */		
39	//	ERET	//eret,	/* 0xD69F03E0ERET */		
40	//	DRPS	//drps,	/* 0xD6BF03E0DRPS */		
41	//	**Unconditional branch (immed**	/* Unconditional branch (immediate) */			
42		B	b,	/* 0x14000000B */		
43		BL	bl,	/* 0x94000000BL */		
44		**Loads and stores**	/* Loads and stores */			
45		**Load/store exclusive**	/* Load/store exclusive */			
46		STXRB	stxrb,	/* 0x08000000STXRB */		
47		STLXRB	stlxrb,	/* 0x08008000STLXRB */		
48		LDXRB	ldxrb,	/* 0x08400000LDXRB */		
49		LDAXRB	ldaxrb,	/* 0x08408000LDAXRB */		
50		STLRB	stlrb,	/* 0x08808000STLRB */		
51		LDARB	ldarb,	/* 0x08C08000LDARB */		
52		STXRH	stxrh,	/* 0x48000000STXRH */		
53		STLXRH	stlxrh,	/* 0x48008000STLXRH */		
54		LDXRH	ldxrh,	/* 0x48400000LDXRH */		
55		LDAXRH	ldaxrh,	/* 0x48408000LDAXRH */		
56		STLRH	stlrh,	/* 0x48808000STLRH */		
57		LDARH	ldarh,	/* 0x48C08000LDARH */		
58		STXR	stxrw,	/* 0x88000000STXR */		
59		STLXR	stlxrw,	/* 0x88008000STLXR */		
60		STXP	stxpw,	/* 0x88200000STXP */		
61		STLXP	stlxpw,	/* 0x88208000STLXP */		
62		LDXR	ldxrw,	/* 0x88400000LDXR */		
63		LDAXR	ldaxrw,	/* 0x88408000LDAXR */		
64		LDXP	ldxpw,	/* 0x88600000LDXP */		
65		LDAXP	ldaxpw,	/* 0x88608000LDAXP */		
66		STLR	stlrw,	/* 0x88808000STLR */		
67		LDAR	ldarw,	/* 0x88C08000LDAR */		
68		STXR	stxrx,	/* 0xC8000000STXR */		
69		STLXR	stlxrx,	/* 0xC8008000STLXR */		
70		STXP	stxpx,	/* 0xC8200000STXP */		
71		STLXP	stlxpx,	/* 0xC8208000STLXP */		
72		LDXR	ldxrx,	/* 0xC8400000LDXR */		
73		LDAXR	ldaxrx,	/* 0xC8408000LDAXR */		
74		LDXP	ldxpx,	/* 0xC8600000LDXP */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
75		LDAXP	ldaxpx,	/* 0xC8608000LDAXP */		
76		STLR	stlrx,	/* 0xC8808000STLR */		
77		LDAR	ldarx,	/* 0xC8C08000LDAR */		
78		**Load register (literal)**	/* Load register (literal) */			
79		LDR	ldrw,	/* 0x18000000LDR */		
80		LDR	vldrs,	/* 0x1C000000LDR */		
81		LDR	ldrx,	/* 0x58000000LDR */		
82		LDR	vldrd,	/* 0x5C000000LDR */		
83		LDRSW	ldrsw,	/* 0x98000000LDRSW */		
84		LDR	vldrq,	/* 0x9C000000LDR */		
85		PRFM	prfm,	/* 0xD8000000PRFM */		
86		**Load/store no-allocate pair (o**	/* Load/store no-allocate pair (offset) */			
87		STNP	stnpw,	/* 0x28000000STNP */		
88		LDNP	ldnpw,	/* 0x28400000LDNP */		
89		STNP	vstnps,	/* 0x2C000000STNP */		
90		LDNP	vldnps,	/* 0x2C400000LDNP */		
91		STNP	vstnpd,	/* 0x6C000000STNP */		
92		LDNP	vldnpd,	/* 0x6C400000LDNP */		
93		STNP	stnpx,	/* 0xA8000000STNP */		
94		LDNP	ldnpx,	/* 0xA8400000LDNP */		
95		STNP	vstnpq,	/* 0xAC000000STNP */		
96		LDNP	vldnpq,	/* 0xAC400000LDNP */		
97		**Load/store register pair (post**	/* Load/store register pair (post-indexed) */			
98		STP	stppostw,	/* 0x28800000STP */		
99		LDP	ldppostw,	/* 0x28C00000LDP */		
100		STP	vstpposts,	/* 0x2C800000STP */		
101		LDP	vldpposts,	/* 0x2CC00000LDP */		
102		LDPSW	ldpswpost,	/* 0x68C00000LDPSW */		
103		STP	vstppostd,	/* 0x6C800000STP */		
104		LDP	vldppostd,	/* 0x6CC00000LDP */		
105		STP	stppostx,	/* 0xA8800000STP */		
106		LDP	ldppostx,	/* 0xA8C00000LDP */		
107		STP	vstppostq,	/* 0xAC800000STP */		
108		LDP	vldppostq,	/* 0xACC00000LDP */		
109		**Load/store register pair (offse**	/* Load/store register pair (offset) */			

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
110		STP	stpoffw,	/* 0x29000000STP */		
111		LDP	ldpoffw,	/* 0x29400000LDP */		
112		STP	vstpoffs,	/* 0x2D000000STP */		
113		LDP	vldpoffs,	/* 0x2D400000LDP */		
114		LDPSW	ldpswoff,	/* 0x69400000LDPSW */		
115		STP	vstpoffd,	/* 0x6D000000STP */		
116		LDP	vldpoffd,	/* 0x6D400000LDP */		
117		STP	stpoffx,	/* 0xA9000000STP */		
118		LDP	ldpoffx,	/* 0xA9400000LDP */		
119		STP	vstpoffq,	/* 0xAD000000STP */		
120		LDP	vldpoffq,	/* 0xAD400000LDP */		
121		**Load/store register pair (pre-i**	/* Load/store register pair (pre-indexed) */			
122		STP	stpprew,	/* 0x29800000STP */		
123		LDP	ldpprew,	/* 0x29C00000LDP */		
124		STP	vstppres,	/* 0x2D800000STP */		
125		LDP	vldppres,	/* 0x2DC00000LDP */		
126		LDPSW	ldpswpre,	/* 0x69C00000LDPSW */		
127		STP	vstppred,	/* 0x6D800000STP */		
128		LDP	vldppred,	/* 0x6DC00000LDP */		
129		STP	stpprex,	/* 0xA9800000STP */		
130		LDP	ldpprex,	/* 0xA9C00000LDP */		
131		STP	vstppreq,	/* 0xAD800000STP */		
132		LDP	vldppreq,	/* 0xADC00000LDP */		
133		**Load/store register (unscaled**	/* Load/store register (unscaled immediate) */			
134		STURB	sturb,	/* 0x38000000STURB */		
135		LDURB	ldurb,	/* 0x38400000LDURB */		
136		LDURSB	ldursbx,	/* 0x38800000LDURSB */		
137		LDURSB	ldursbw,	/* 0x38C00000LDURSB */		
138		STUR	vsturb,	/* 0x3C000000STUR */		
139		LDUR	vldurb,	/* 0x3C400000LDUR */		
140		STUR	vsturq,	/* 0x3C800000STUR */		
141		LDUR	vldurq,	/* 0x3CC00000LDUR */		
142		STURH	sturh,	/* 0x78000000STURH */		
143		LDURH	ldurh,	/* 0x78400000LDURH */		
144		LDURSH	ldurshx,	/* 0x78800000LDURSH */		
145		LDURSH	ldurshw,	/* 0x78C00000LDURSH */		
146		STUR	vsturh,	/* 0x7C000000STUR */		
147		LDUR	vldurh,	/* 0x7C400000LDUR */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
148		STUR	sturw,	/* 0xB8000000STUR */		
149		LDUR	ldurw,	/* 0xB8400000LDUR */		
150		LDURSW	ldursw,	/* 0xB8800000LDURSW */		
151		STUR	vsturs,	/* 0xBC000000STUR */		
152		LDUR	vldurs,	/* 0xBC400000LDUR */		
153		STUR	sturx,	/* 0xF8000000STUR */		
154		LDUR	ldurx,	/* 0xF8400000LDUR */		
155		PRFUM	prfum,	/* 0xF8800000PRFUM */		
156		STUR	vsturd,	/* 0xFC000000STUR */		
157		LDUR	vldurd,	/* 0xFC400000LDUR */		
158		**Load/store register (immediat**	/* Load/store register (immediate post-indexed) */			
159		STRB	strbpost,	/* 0x38000400STRB */		
160		LDRB	ldrbpost,	/* 0x38400400LDRB */		
161		LDRSB	ldrsbpostx,	/* 0x38800400LDRSB */		
162		LDRSB	ldrsbpostw,	/* 0x38C00400LDRSB */		
163		STR	vstrpostb,	/* 0x3C000400STR */		
164		LDR	vldrpostb,	/* 0x3C400400LDR */		
165		STR	vstrpostq,	/* 0x3C800400STR */		
166		LDR	vldrpostq,	/* 0x3CC00400LDR */		
167		STRH	strhpost,	/* 0x78000400STRH */		
168		LDRH	ldrhpost,	/* 0x78400400LDRH */		
169		LDRSH	ldrshpostx,	/* 0x78800400LDRSH */		
170		LDRSH	ldrshpostw,	/* 0x78C00400LDRSH */		
171		STR	vstrposth,	/* 0x7C000400STR */		
172		LDR	vldrposth,	/* 0x7C400400LDR */		
173		STR	strpostw,	/* 0xB8000400STR */		
174		LDR	ldrpostw,	/* 0xB8400400LDR */		
175		LDRSW	ldrswpost,	/* 0xB8800400LDRSW */		
176		STR	vstrposts,	/* 0xBC000400STR */		
177		LDR	vldrposts,	/* 0xBC400400LDR */		
178		STR	strpostx,	/* 0xF8000400STR */		
179		LDR	ldrpostx,	/* 0xF8400400LDR */		
180		STR	vstrpostd,	/* 0xFC000400STR */		
181		LDR	vldrpostd,	/* 0xFC400400LDR */		
182		**Load/store register (unprivile**	/* Load/store register (unprivileged) */			
183		STTRB	sttrb,	/* 0x38000800STTRB */		
184		LDTRB	ldtrb,	/* 0x38400800LDTRB */		
185		LDTRSB	ldtrsbx,	/* 0x38800800LDTRSB */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
186		LDTRSB	ldtrsbw,	/* 0x38C00800LDTRSB */		
187		STTRH	sttrh,	/* 0x78000800STTRH */		
188		LDTRH	ldtrh,	/* 0x78400800LDTRH */		
189		LDTRSH	ldtrshx,	/* 0x78800800LDTRSH */		
190		LDTRSH	ldtrshw,	/* 0x78C00800LDTRSH */		
191		STTR	sttrw,	/* 0xB8000800STTR */		
192		LDTR	ldtrw,	/* 0xB8400800LDTR */		
193		LDTRSW	ldtrsw,	/* 0xB8800800LDTRSW */		
194		STTR	sttrx,	/* 0xF8000800STTR */		
195		LDTR	ldtrx,	/* 0xF8400800LDTR */		
196		**Load/store register (immediat**	/* Load/store register (immediate pre-indexed) */			
197		STRB	strbpre,	/* 0x38000C00STRB */		
198		LDRB	ldrbpre,	/* 0x38400C00LDRB */		
199		LDRSB	ldrsbprex,	/* 0x38800C00LDRSB */		
200		LDRSB	ldrsbprew,	/* 0x38C00C00LDRSB */		
201		STR	vstrpreb,	/* 0x3C000C00STR */		
202		LDR	vldrpreb,	/* 0x3C400C00LDR */		
203		STR	vstrpreq,	/* 0x3C800C00STR */		
204		LDR	vldrpreq,	/* 0x3CC00C00LDR */		
205		STRH	strhpre,	/* 0x78000C00STRH */		
206		LDRH	ldrhpre,	/* 0x78400C00LDRH */		
207		LDRSH	ldrshprex,	/* 0x78800C00LDRSH */		
208		LDRSH	ldrshprew,	/* 0x78C00C00LDRSH */		
209		STR	vstrpreh,	/* 0x7C000C00STR */		
210		LDR	vldrpreh,	/* 0x7C400C00LDR */		
211		STR	strprew,	/* 0xB8000C00STR */		
212		LDR	ldrprew,	/* 0xB8400C00LDR */		
213		LDRSW	ldrswpre,	/* 0xB8800C00LDRSW */		
214		STR	vstrpres,	/* 0xBC000C00STR */		
215		LDR	vldrpres,	/* 0xBC400C00LDR */		
216		STR	strprex,	/* 0xF8000C00STR */		
217		LDR	ldrprex,	/* 0xF8400C00LDR */		
218		STR	vstrpred,	/* 0xFC000C00STR */		
219		LDR	vldrpred,	/* 0xFC400C00LDR */		
220		**Load/store register (register o**	/* Load/store register (register offset) */			
221		STRB	strboff,	/* 0x38200800STRB */		
222		LDRB	ldrboff,	/* 0x38600800LDRB */		
223		LDRSB	ldrsboffx,	/* 0x38A00800LDRSB */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
224		LDRSB	ldrsboffw,	/* 0x38E00800LDRSB */		
225		STR	vstroffb,	/* 0x3C200800STR */		
226		LDR	vldroffb,	/* 0x3C600800LDR */		
227		STR	vstroffq,	/* 0x3CA00800STR */		
228		LDR	vldroffq,	/* 0x3CE00800LDR */		
229		STRH	strhoff,	/* 0x78200800STRH */		
230		LDRH	ldrhoff,	/* 0x78600800LDRH */		
231		LDRSH	ldrshoffx,	/* 0x78A00800LDRSH */		
232		LDRSH	ldrshoffw,	/* 0x78E00800LDRSH */		
233		STR	vstroffh,	/* 0x7C200800STR */		
234		LDR	vldroffh,	/* 0x7C600800LDR */		
235		STR	stroffw,	/* 0xB8200800STR */		
236		LDR	ldroffw,	/* 0xB8600800LDR */		
237		LDRSW	ldrswoff,	/* 0xB8A00800LDRSW */		
238		STR	vstroffs,	/* 0xBC200800STR */		
239		LDR	vldroffs,	/* 0xBC600800LDR */		
240		STR	stroffx,	/* 0xF8200800STR */		
241		LDR	ldroffx,	/* 0xF8600800LDR */		
243		STR	vstroffd,	/* 0xFC200800STR */		
244		LDR	vldroffd,	/* 0xFC600800LDR */		
242		PRFM	prfmoff,	/* 0xF8A00800PRFM */		
245		**Load/store register (unsigned**	/* Load/store register (unsigned immediate) */			
246		STRB	strbimm,	/* 0x39000000STRB */		
247		LDRB	ldrbimm,	/* 0x39400000LDRB */		
248		LDRSB	ldrsbimmx,	/* 0x39800000LDRSB */		
249		LDRSB	ldrsbimmw,	/* 0x39C00000LDRSB */		
250		STR	vstrimmb,	/* 0x3D000000STR */		
251		LDR	vldrimmb,	/* 0x3D400000LDR */		
252		STR	vstrimmq,	/* 0x3D800000STR */		
253		LDR	vldrimmq,	/* 0x3DC00000LDR */		
254		STRH	strhimm,	/* 0x79000000STRH */		
255		LDRH	ldrhimm,	/* 0x79400000LDRH */		
256		LDRSH	ldrshimmx,	/* 0x79800000LDRSH */		
257		LDRSH	ldrshimmw,	/* 0x79C00000LDRSH */		
258		STR	vstrimmh,	/* 0x7D000000STR */		
259		LDR	vldrimmh,	/* 0x7D400000LDR */		
260		STR	strimmw,	/* 0xB9000000STR */		
261		LDR	ldrimmw,	/* 0xB9400000LDR */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
262		LDRSW	ldrswimm,	/* 0xB9800000LDRSW */		
263		STR	vstrimms,	/* 0xBD000000STR */		
264		LDR	vldrimms,	/* 0xBD400000LDR */		
265		STR	strimmx,	/* 0xF9000000STR */		
266		LDR	ldrimmx,	/* 0xF9400000LDR */		
268		STR	vstrimmd,	/* 0xFD000000STR */		
269		LDR	vldrimmd,	/* 0xFD400000LDR */		
267		PRFM	prfmimm,	/* 0xF9800000PRFM */		
270		**Data processing – Immedia**	/* Data processing – Immediate */			
271		**PC-rel. addressing**	/* PC-rel. addressing */			
272		ADR	adr,	/* 0x10000000ADR */		
273		ADRP	adrp,	/* 0x90000000ADRP */		
274		**Add/subtract (immediate)**	/* Add/subtract (immediate) */			
275		ADD	addimmw,	/* 0x11000000ADD */		
276		ADDS	addsimmw,	/* 0x31000000ADDS */		
277		SUB	subimmw,	/* 0x51000000SUB */		
278		SUBS	subsimmw,	/* 0x71000000SUBS */		
279		ADD	addimmx,	/* 0x91000000ADD */		
280		ADDS	addsimmx,	/* 0xB1000000ADDS */		
281		SUB	subimmx,	/* 0xD1000000SUB */		
282		SUBS	subsimmx,	/* 0xF1000000SUBS */		
283		**Logical (immediate)**	/* Logical (immediate) */			
284		AND	andimmw,	/* 0x12000000AND */		
285		ORR	orrimmw,	/* 0x32000000ORR */		
286		EOR	eorimmw,	/* 0x52000000EOR */		
287		ANDS	andsimmw,	/* 0x72000000ANDS */		
288		AND	andimmx,	/* 0x92000000AND */		
289		ORR	orrimmx,	/* 0xB2000000ORR */		
290		EOR	eorimmx,	/* 0xD2000000EOR */		
291		ANDS	andsimmx,	/* 0xF2000000ANDS */		
292		**Move wide (immediate)**	/* Move wide (immediate) */			
293		MOVN	movnw,	/* 0x12800000MOVN */		
294		MOVZ	movzw,	/* 0x52800000MOVZ */		
295		MOVK	movkw,	/* 0x72800000MOVK */		
296		MOVN	movnx,	/* 0x92800000MOVN */		
297		MOVZ	movzx,	/* 0xD2800000MOVZ */		
298		MOVK	movkx,	/* 0xF2800000MOVK */		
299		**Bitfield**	/* Bitfield */			

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
300		SBFM	sbfmw,	/* 0x13000000SBFM */		
301		BFM	bfmw,	/* 0x33000000BFM */		
302		UBFM	ubfmw,	/* 0x53000000UBFM */		
303		SBFM	sbfmx,	/* 0x93400000SBFM */		
304		BFM	bfmx,	/* 0xB3400000BFM */		
305		UBFM	ubfmx,	/* 0xD3400000UBFM */		
306		**Extract**	/* Extract */			
307		EXTR	extrw,	/* 0x13800000EXTR */		
308		EXTR	extrx,	/* 0x93C08000EXTR */		
309		**Data Processing – register**	/* Data Processing – register */			
310		**Logical (shifted register)**	/* Logical (shifted register) */			
311		AND	andw,	/* 0x0A000000AND */		
312		BIC	bicw,	/* 0x0A200000BIC */		
313		ORR	orrw,	/* 0x2A000000ORR */		
314		ORN	ornw,	/* 0x2A200000ORN */		
315		EOR	eorw,	/* 0x4A000000EOR */		
316		EON	eonw,	/* 0x4A200000EON */		
317		ANDS	andsw,	/* 0x6A000000ANDS */		
318		BICS	bicsw,	/* 0x6A200000BICS */		
319		AND	andx,	/* 0x8A000000AND */		
320		BIC	bicx,	/* 0x8A200000BIC */		
321		ORR	orrx,	/* 0xAA000000ORR */		
322		ORN	ornx,	/* 0xAA200000ORN */		
323		EOR	eorx,	/* 0xCA000000EOR */		
324		EON	eonx,	/* 0xCA200000EON */		
325		ANDS	andsx,	/* 0xEA000000ANDS */		
326		BICS	bicsx,	/* 0xEA200000BICS */		
327		**Add/subtract (shifted register**	/* Add/subtract (shifted register) */			
328		ADD	addw,	/* 0x0B000000ADD */		
329		ADDS	addsw,	/* 0x2B000000ADDS */		
330		SUB	subw,	/* 0x4B000000SUB */		
331		SUBS	subsw,	/* 0x6B000000SUBS */		
332		ADD	addx,	/* 0x8B000000ADD */		
333		ADDS	addsx,	/* 0xAB000000ADDS */		
334		SUB	subx,	/* 0xCB000000SUB */		
335		SUBS	subsx,	/* 0xEB000000SUBS */		
336		**Add/subtract (extended regis**	/* Add/subtract (extended register) */			
337		ADD	addextw,	/* 0x0B200000ADD */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
338		ADDS	addsextw,	/* 0x2B200000ADDS */		
339		SUB	subextw,	/* 0x4B200000SUB */		
340		SUBS	subsextw,	/* 0x6B200000SUBS */		
341		ADD	addextx,	/* 0x8B200000ADD */		
342		ADDS	addsextx,	/* 0xAB200000ADDS */		
343		SUB	subextx,	/* 0xCB200000SUB */		
344		SUBS	subsextx,	/* 0xEB200000SUBS */		
345		**Add/subtract (with carry)**	/* Add/subtract (with carry) */			
346		ADC	adcw,	/* 0x1A000000ADC */		
347		ADCS	adcsw,	/* 0x3A000000ADCS */		
348		SBC	sbcw,	/* 0x5A000000SBC */		
349		SBCS	sbcsw,	/* 0x7A000000SBCS */		
350		ADC	adcx,	/* 0x9A000000ADC */		
351		ADCS	adcsx,	/* 0xBA000000ADCS */		
352		SBC	sbcx,	/* 0xDA000000SBC */		
353		SBCS	sbcsx,	/* 0xFA000000SBCS */		
354		**Conditional compare (registe**	/* Conditional compare (register) */			
355		CCMN	ccmnw,	/* 0x3A400000CCMN */		
356		CCMN	ccmnx,	/* 0xBA400000CCMN */		
357		CCMP	ccmpw,	/* 0x7A400000CCMP */		
358		CCMP	ccmpx,	/* 0xFA400000CCMP */		
359		**Conditional compare (immedi**	/* Conditional compare (immediate) */			
360		CCMN	ccmnimmw,	/* 0x3A400800CCMN */		
361		CCMN	ccmnimmx,	/* 0xBA400800CCMN */		
362		CCMP	ccmpimmw,	/* 0x7A400800CCMP */		
363		CCMP	ccmpimmx,	/* 0xFA400800CCMP */		
364		**Conditional select**	/* Conditional select */			
365		CSEL	cselw,	/* 0x1A800000CSEL */		
366		CSINC	csincw,	/* 0x1A800400CSINC */		
367		CSINV	csinvw,	/* 0x5A800000CSINV */		
368		CSNEG	csnegw,	/* 0x5A800400CSNEG */		
369		CSEL	cselx,	/* 0x9A800000CSEL */		
370		CSINC	csincx,	/* 0x9A800400CSINC */		
371		CSINV	csinvx,	/* 0xDA800000CSINV */		
372		CSNEG	csnegx,	/* 0xDA800400CSNEG */		
373		**Data-processing (3 source)**	/* Data-processing (3 source) */			
374		MADD	maddw,	/* 0x1B000000MADD */		
375		MADD	maddx,	/* 0x9B000000MADD */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
376		SMADDL	smaddl,	/* 0x9B200000SMADDL */		
377		UMADDL	umaddl,	/* 0x9BA00000UMADDL */		
378		MSUB	msubw,	/* 0x1B008000MSUB */		
379		MSUB	msubx,	/* 0x9B008000MSUB */		
380		SMSUBL	smsubl,	/* 0x9B208000SMSUBL */		
381		UMSUBL	umsubl,	/* 0x9BA08000UMSUBL */		
382		SMULH	smulh,	/* 0x9B400000SMULH */		
383		UMULH	umulh,	/* 0x9BC00000UMULH */		
384		**Data-processing (2 source)**	/* Data-processing (2 source) */			
385		CRC32X	crc32x,	/* 0x9AC04C00CRC32X */		
386		CRC32CX	crc32cx,	/* 0x9AC05C00CRC32CX */		
387		CRC32B	crc32b,	/* 0x1AC04000CRC32B */		
388		CRC32CB	crc32cb,	/* 0x1AC05000CRC32CB */		
389		CRC32H	crc32h,	/* 0x1AC04400CRC32H */		
390		CRC32CH	crc32ch,	/* 0x1AC05400CRC32CH */		
391		CRC32W	crc32w,	/* 0x1AC04800CRC32W */		
392		CRC32CW	crc32cw,	/* 0x1AC05800CRC32CW */		
393		UDIV	udivw,	/* 0x1AC00800UDIV */		
394		UDIV	udivx,	/* 0x9AC00800UDIV */		
395		SDIV	sdivw,	/* 0x1AC00C00SDIV */		
396		SDIV	sdivx,	/* 0x9AC00C00SDIV */		
397		LSLV	lslvw,	/* 0x1AC02000LSLV */		
398		LSLV	lslvx,	/* 0x9AC02000LSLV */		
399		LSRV	lsrvw,	/* 0x1AC02400LSRV */		
400		LSRV	lsrvx,	/* 0x9AC02400LSRV */		
401		ASRV	asrvw,	/* 0x1AC02800ASRV */		
402		ASRV	asrvx,	/* 0x9AC02800ASRV */		
403		RORV	rorvw,	/* 0x1AC02C00RORV */		
404		RORV	rorvx,	/* 0x9AC02C00RORV */		
405		**Data-processing (1 source)**	/* Data-processing (1 source) */			
406		RBIT	rbitw,	/* 0x5AC00000RBIT */		
407		RBIT	rbitx,	/* 0xDAC00000RBIT */		
408		CLZ	clzw,	/* 0x5AC01000CLZ */		
409		CLZ	clzx,	/* 0xDAC01000CLZ */		
410		CLS	clsw,	/* 0x5AC01400CLS */		
411		CLS	clsx,	/* 0xDAC01400CLS */		
412		REV	revw,	/* 0x5AC00800REV */		
413		REV	revx,	/* 0xDAC00C00REV */		

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
414		REV16	rev16w,	/* 0xDAC00400REV16	*/	
415		REV16	rev16x,	/* 0x5AC00400REV16	*/	
416		REV32	rev32,	/* 0xDAC00800REV32	*/	
417	//	**Data Processing – SIMD an**	/* Data Processing – SIMD and floating point */			
418	//	**Floating-point<->fixed-point c**	/* Floating-point<->fixed-point conversions */			
419	//	SCVTF	//vscvtfscalar_fixed_point_32_bit_to_single_precision,	/* 0x1E020000SCVTF	*/	
420	//	UCVTF	//vucvtfscalar_fixed_point_32_bit_to_single_precision,	/* 0x1E030000UCVTF	*/	
421	//	FCVTZS	//vfcvtzsscalar_fixed_point_Single_precision_to_32_bit,	/* 0x1ED80000FCVTZS	*/	
422	//	FCVTZU	//vfcvtzuscalar_fixed_point_Single_precision_to_32_bit,	/* 0x1ED90000FCVTZU	*/	
423	//	SCVTF	//vscvtfscalar_fixed_point_32_bit_to_double_precision,	/* 0x1E020000SCVTF	*/	
424	//	UCVTF	//vucvtfscalar_fixed_point_32_bit_to_double_precision,	/* 0x1E030000UCVTF	*/	
425	//	FCVTZS	//vfcvtzsscalar_fixed_point_Double_precision_to_32_bit,	/* 0x1ED80000FCVTZS	*/	
426	//	FCVTZU	//vfcvtzuscalar_fixed_point_Double_precision_to_32_bit,	/* 0x1ED90000FCVTZU	*/	
427	//	SCVTF	//vscvtfscalar_fixed_point_64_bit_to_single_precision,	/* 0x9E020000SCVTF	*/	
428	//	UCVTF	//vucvtfscalar_fixed_point_64_bit_to_single_precision,	/* 0x9E030000UCVTF	*/	
429	//	FCVTZS	//vfcvtzsscalar_fixed_point_Single_precision_to_64_bit,	/* 0x9ED80000FCVTZS	*/	
430	//	FCVTZU	//vfcvtzuscalar_fixed_point_Single_precision_to_64_bit,	/* 0x9ED90000FCVTZU	*/	
431	//	SCVTF	//vscvtfscalar_fixed_point_64_bit_to_double_precision,	/* 0x9E020000SCVTF	*/	
432	//	UCVTF	//vucvtfscalar_fixed_point_64_bit_to_double_precision,	/* 0x9E030000UCVTF	*/	
433	//	FCVTZS	//vfcvtzsscalar_fixed_point_Double_precision_to_64_bit,	/* 0x9ED80000FCVTZS	*/	
434	//	FCVTZU	//vfcvtzuscalar_fixed_point_Double_precision_to_64_bit,	/* 0x9ED90000FCVTZU	*/	
435	//	**Floating-point conditional cor**	/* Floating-point conditional compare */			
436	//	FCCMP	//vfccmpSingle_precision,	/* 0x1E200400FCCMP	*/	
437	//	FCCMPE	//vfccmpeSingle_precision,	/* 0x1E200410FCCMPE	*/	
438	//	FCCMP	//vfccmpDouble_precision,	/* 0x1E600400FCCMP	*/	
439	//	FCCMPE	//vfccmpeDouble_precision,	/* 0x1E600410FCCMPE	*/	
440	//	**Floating-point data-processin**	/* Floating-point data-processing (2 source) */			
441	//	FMUL	//vfmulscalar_Single_precision,	/* 0x1E200800FMUL	*/	
442	//	FDIV	//vfdivscalar_Single_precision,	/* 0x1E201800FDIV	*/	
443	//	FADD	//vfaddscalar_Single_precision,	/* 0x1E202800FADD	*/	
444	//	FSUB	//vfsubscalar_Single_precision,	/* 0x1E203800FSUB	*/	
445	//	FMAX	//vfmaxscalar_Single_precision,	/* 0x1E204800FMAX	*/	
446	//	FMIN	//vfminscalar_Single_precision,	/* 0x1E205800FMIN	*/	
447	//	FMAXNM	//vfmaxnmscalar_Single_precision,	/* 0x1E206800FMAXNM	*/	

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
448	//	FMINNM	//vfminnmscalar_Single_precision,	/* 0x1E207800FMINNM */
449	//	FNMUL	//vfnmulSingle_precision,	/* 0x1E208800FNMUL */
450	//	FMUL	//vfmulscalar_Double_precision,	/* 0x1E600800FMUL */
451	//	FDIV	//vfdivscalar_Double_precision,	/* 0x1E601800FDIV */
452	//	FADD	//vfaddscalar_Double_precision,	/* 0x1E602800FADD */
453	//	FSUB	//vfsubscalar_Double_precision,	/* 0x1E603800FSUB */
454	//	FMAX	//vfmaxscalar_Double_precision,	/* 0x1E604800FMAX */
455	//	FMIN	//vfminscalar_Double_precision,	/* 0x1E605800FMIN */
456	//	FMAXNM	//vfmaxnmscalar_Double_precision,	/* 0x1E606800FMAXNM */
457	//	FMINNM	//vfminnmscalar_Double_precision,	/* 0x1E607800FMINNM */
458	//	FNMUL	//vfnmulDouble_precision,	/* 0x1E608800FNMUL */
459	//	**Floating-point conditional sel**	/* Floating-point conditional select */	
460	//	FCSEL	//vfcselSingle_precision,	/* 0x1E200C00FCSEL */
461	//	FCSEL	//vfcselDouble_precision,	/* 0x1E600C00FCSEL */
462	//	**Floating-point immediate**	/* Floating-point immediate */	
463	//	FMOV	//vfmovscalar_immediate_Single_precision,	/* 0x1E201000FMOV */
464	//	FMOV	//vfmovscalar_immediate_Double_precision,	/* 0x1E601000FMOV */
465	//	**Floating-point compare**	/* Floating-point compare */	
466	//	FCMP	//vfcmpSingle_precision,	/* 0x1E202000FCMP */
467	//	FCMP	//vfcmpSingle_precision_zero,	/* 0x1E202008FCMP */
468	//	FCMPE	//vfcmpeSingle_precision,	/* 0x1E202010FCMPE */
469	//	FCMPE	//vfcmpeSingle_precision_zero,	/* 0x1E202018FCMPE */
470	//	FCMP	//vfcmpDouble_precision,	/* 0x1E602000FCMP */
471	//	FCMP	//vfcmpDouble_precision_zero,	/* 0x1E602008FCMP */
472	//	FCMPE	//vfcmpeDouble_precision,	/* 0x1E602010FCMPE */
473	//	FCMPE	//vfcmpeDouble_precision_zero,	/* 0x1E602018FCMPE */
474	//	**Floating-point data-processin**	/* Floating-point data-processing (1 source) */	
475	//	FMOV	//vfmovregister_Single_precision,	/* 0x1E204000FMOV */
476	//	FABS	//vfabsscalar_Single_precision,	/* 0x1E20C000FABS */
477	//	FNEG	//vfnegscalar_Single_precision,	/* 0x1E214000FNEG */
478	//	FSQRT	//vfsqrtscalar_Single_precision,	/* 0x1E21C000FSQRT */
479	//	FCVT	//vfcvtSingle_precision_to_double_precision,	/* 0x1E22C000FCVT */
480	//	FCVT	//vfcvtSingle_precision_to_half_precision,	/* 0x1E23C000FCVT */
481	//	FRINTN	//vfrintnscalar_Single_precision,	/* 0x1E244000FRINTN */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
482	//	FRINTP	//vfrintpscalar_Single_precision,	/* 0x1E24C000FRINTP */
483	//	FRINTM	//vfrintmscalar_Single_precision,	/* 0x1E254000FRINTM */
484	//	FRINTZ	//vfrintzscalar_Single_precision,	/* 0x1E25C000FRINTZ */
485	//	FRINTA	//vfrintascalar_Single_precision,	/* 0x1E264000FRINTA */
486	//	FRINTX	//vfrintxscalar_Single_precision,	/* 0x1E274000FRINTX */
487	//	FRINTI	//vfrintiscalar_Single_precision,	/* 0x1E27C000FRINTI */
488	//	FMOV	//vfmovregister_Double_precision,	/* 0x1E604000FMOV */
489	//	FABS	//vfabsscalar_Double_precision,	/* 0x1E60C000FABS */
490	//	FNEG	//vfnegscalar_Double_precision,	/* 0x1E614000FNEG */
491	//	FSQRT	//vfsqrtscalar_Double_precision,	/* 0x1E61C000FSQRT */
492	//	FCVT	//vfcvtDouble_precision_to_single_precision,	/* 0x1E624000FCVT */
493	//	FCVT	//vfcvtDouble_precision_to_half_precision,	/* 0x1E63C000FCVT */
494	//	FRINTN	//vfrintnscalar_Double_precision,	/* 0x1E644000FRINTN */
495	//	FRINTP	//vfrintpscalar_Double_precision,	/* 0x1E64C000FRINTP */
496	//	FRINTM	//vfrintmscalar_Double_precision,	/* 0x1E654000FRINTM */
497	//	FRINTZ	//vfrintzscalar_Double_precision,	/* 0x1E65C000FRINTZ */
498	//	FRINTA	//vfrintascalar_Double_precision,	/* 0x1E664000FRINTA */
499	//	FRINTX	//vfrintxscalar_Double_precision,	/* 0x1E674000FRINTX */
500	//	FRINTI	//vfrintiscalar_Double_precision,	/* 0x1E67C000FRINTI */
501	//	FCVT	//vfcvtHalf_precision_to_single_precision,	/* 0x1EE24000FCVT */
502	//	FCVT	//vfcvtHalf_precision_to_double_precision,	/* 0x1EE2C000FCVT */
503	//	**Floating-point<->integer conv**	/* Floating-point<->integer conversions */	
504	//	FCVTNS	//vfcvtnsscalar_Single_precision_to_32_bit,	/* 0x1E200000FCVTNS */
505	//	FCVTNU	//vfcvtnuscalar_Single_precision_to_32_bit,	/* 0x1E210000FCVTNU */
506	//	SCVTF	//vscvtfscalar_integer_32_bit_to_single_precision,	/* 0x1E220000SCVTF */
507	//	UCVTF	//vucvtfscalar_integer_32_bit_to_single_precision,	/* 0x1E230000UCVTF */
508	//	FCVTAS	//vfcvtasscalar_Single_precision_to_32_bit,	/* 0x1E240000FCVTAS */
509	//	FCVTAU	//vfcvtauscalar_Single_precision_to_32_bit,	/* 0x1E250000FCVTAU */
510	//	FMOV	//vfmovgeneral_Single_precision_to_32_bit,	/* 0x1E260000FMOV */
511	//	FMOV	//vfmovgeneral_32_bit_to_single_precision,	/* 0x1E270000FMOV */
512	//	FCVTPS	//vfcvtpsscalar_Single_precision_to_32_bit,	/* 0x1E280000FCVTPS */
513	//	FCVTPU	//vfcvtpuscalar_Single_precision_to_32_bit,	/* 0x1E290000FCVTPU */
514	//	FCVTMS	//vfcvtmsscalar_Single_precision_to_32_bit,	/* 0x1E300000FCVTMS */
515	//	FCVTMU	//vfcvtmuscalar_Single_precision_to_32_bit,	/* 0x1E310000FCVTMU */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
516	//	FCVTZS	//vfcvtzsscalar_integer_Single_precision_to_32_bit,	/* 0x1E380000FCVTZS */
517	//	FCVTZU	//vfcvtzuscalar_integer_Single_precision_to_32_bit,	/* 0x1E390000FCVTZU */
518	//	FCVTNS	//vfcvtnsscalar_Double_precision_to_32_bit,	/* 0x1E600000FCVTNS */
519	//	FCVTNU	//vfcvtnuscalar_Double_precision_to_32_bit,	/* 0x1E610000FCVTNU */
520	//	SCVTF	//vscvtfscalar_integer_32_bit_to_double_precision,	/* 0x1E620000SCVTF */
521	//	UCVTF	//vucvtfscalar_integer_32_bit_to_double_precision,	/* 0x1E630000UCVTF */
522	//	FCVTAS	//vfcvtasscalar_Double_precision_to_32_bit,	/* 0x1E640000FCVTAS */
523	//	FCVTAU	//vfcvtauscalar_Double_precision_to_32_bit,	/* 0x1E650000FCVTAU */
524	//	FCVTPS	//vfcvtpsscalar_Double_precision_to_32_bit,	/* 0x1E680000FCVTPS */
525	//	FCVTPU	//vfcvtpuscalar_Double_precision_to_32_bit,	/* 0x1E690000FCVTPU */
526	//	FCVTMS	//vfcvtmsscalar_Double_precision_to_32_bit,	/* 0x1E700000FCVTMS */
527	//	FCVTMU	//vfcvtmuscalar_Double_precision_to_32_bit,	/* 0x1E710000FCVTMU */
528	//	FCVTZS	//vfcvtzsscalar_integer_Double_precision_to_32_bit,	/* 0x1E780000FCVTZS */
529	//	FCVTZU	//vfcvtzuscalar_integer_Double_precision_to_32_bit,	/* 0x1E790000FCVTZU */
530	//	FCVTNS	//vfcvtnsscalar_Single_precision_to_64_bit,	/* 0x9E200000FCVTNS */
531	//	FCVTNU	//vfcvtnuscalar_Single_precision_to_64_bit,	/* 0x9E210000FCVTNU */
532	//	SCVTF	//vscvtfscalar_integer_64_bit_to_single_precision,	/* 0x9E220000SCVTF */
533	//	UCVTF	//vucvtfscalar_integer_64_bit_to_single_precision,	/* 0x9E230000UCVTF */
534	//	FCVTAS	//vfcvtasscalar_Single_precision_to_64_bit,	/* 0x9E240000FCVTAS */
535	//	FCVTAU	//vfcvtauscalar_Single_precision_to_64_bit,	/* 0x9E250000FCVTAU */
536	//	FCVTPS	//vfcvtpsscalar_Single_precision_to_64_bit,	/* 0x9E280000FCVTPS */
537	//	FCVTPU	//vfcvtpuscalar_Single_precision_to_64_bit,	/* 0x9E290000FCVTPU */
538	//	FCVTMS	//vfcvtmsscalar_Single_precision_to_64_bit,	/* 0x9E300000FCVTMS */
539	//	FCVTMU	//vfcvtmuscalar_Single_precision_to_64_bit,	/* 0x9E310000FCVTMU */
540	//	FCVTZS	//vfcvtzsscalar_integer_Single_precision_to_64_bit,	/* 0x9E380000FCVTZS */
541	//	FCVTZU	//vfcvtzuscalar_integer_Single_precision_to_64_bit,	/* 0x9E390000FCVTZU */
542	//	FCVTNS	//vfcvtnsscalar_Double_precision_to_64_bit,	/* 0x9E600000FCVTNS */
543	//	FCVTNU	//vfcvtnuscalar_Double_precision_to_64_bit,	/* 0x9E610000FCVTNU */
544	//	SCVTF	//vscvtfscalar_integer_64_bit_to_double_precision,	/* 0x9E620000SCVTF */
545	//	UCVTF	//vucvtfscalar_integer_64_bit_to_double_precision,	/* 0x9E630000UCVTF */
546	//	FCVTAS	//vfcvtasscalar_Double_precision_to_64_bit,	/* 0x9E640000FCVTAS */
547	//	FCVTAU	//vfcvtauscalar_Double_precision_to_64_bit,	/* 0x9E650000FCVTAU */
548	//	FMOV	//vfmovgeneral_Double_precision_to_64_bit,	/* 0x9E660000FMOV */
549	//	FMOV	//vfmovgeneral_64_bit_to_double_precision,	/* 0x9E670000FMOV */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
550	//	FCVTPS	//vfcvtpsscalar_Double_precision_to_64_bit,	/* 0x9E680000FCVTPS */
551	//	FCVTPU	//vfcvtpuscalar_Double_precision_to_64_bit,	/* 0x9E690000FCVTPU */
552	//	FCVTMS	//vfcvtmsscalar_Double_precision_to_64_bit,	/* 0x9E700000FCVTMS */
553	//	FCVTMU	//vfcvtmuscalar_Double_precision_to_64_bit,	/* 0x9E710000FCVTMU */
554	//	FCVTZS	//vfcvtzsscalar_integer_Double_precision_to_64_bit,	/* 0x9E780000FCVTZS */
555	//	FCVTZU	//vfcvtzuscalar_integer_Double_precision_to_64_bit,	/* 0x9E790000FCVTZU */
556	//	FMOV	//vfmovgeneral_Top_half_of_128_bit_to_64_bit,	/* 0x9EAE0000FMOV */
557	//	FMOV	//vfmovgeneral_64_bit_to_top_half_of_128_bit,	/* 0x9EAF0000FMOV */
558	//	**Floating-point data-processin**	/* Floating-point data-processing (3 source) */	
559	//	FMADD	//vfmaddSingle_precision,	/* 0x1F000000FMADD */
560	//	FMSUB	//vfmsubSingle_precision,	/* 0x1F008000FMSUB */
561	//	FNMADD	//vfnmaddSingle_precision,	/* 0x1F200000FNMADD */
562	//	FNMSUB	//vfnmsubSingle_precision,	/* 0x1F208000FNMSUB */
563	//	FMADD	//vfmaddDouble_precision,	/* 0x1F400000FMADD */
564	//	FMSUB	//vfmsubDouble_precision,	/* 0x1F408000FMSUB */
565	//	FNMADD	//vfnmaddDouble_precision,	/* 0x1F600000FNMADD */
566	//	FNMSUB	//vfnmsubDouble_precision,	/* 0x1F608000FNMSUB */
567	//	**AdvSIMD scalar three same**	/* AdvSIMD scalar three same */	
568	//	SQADD	//vsqaddScalar,	/* 0x5E200C00SQADD */
569	//	SQSUB	//vsqsubScalar,	/* 0x5E202C00SQSUB */
570	//	CMGT	//vcmgtregister_Scalar,	/* 0x5E203400CMGT */
571	//	CMGE	//vcmgeregister_Scalar,	/* 0x5E203C00CMGE */
572	//	SSHL	//vsshlScalar,	/* 0x5E204400SSHL */
573	//	SQSHL	//vsqshlregister_Scalar,	/* 0x5E204C00SQSHL */
574	//	SRSHL	//vsrshlScalar,	/* 0x5E205400SRSHL */
575	//	SQRSHL	//vsqrshlScalar,	/* 0x5E205C00SQRSHL */
576	//	ADD	//vaddvector_Scalar,	/* 0x5E208400ADD */
577	//	CMTST	//vcmtstScalar,	/* 0x5E208C00CMTST */
578	//	SQDMULH	//vsqdmulhvector_Scalar,	/* 0x5E20B400SQDMULH */
579	//	FMULX	//vfmulxScalar,	/* 0x5E20DC00FMULX */
580	//	FCMEQ	//vfcmeqregister_Scalar,	/* 0x5E20E400FCMEQ */
581	//	FRECPS	//vfrecpsScalar,	/* 0x5E20FC00FRECPS */
582	//	FRSQRTS	//vfrsqrtsScalar,	/* 0x5EA0FC00FRSQRTS */
583	//	UQADD	//vuqaddScalar,	/* 0x7E200C00UQADD */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
584	//	UQSUB	//vuqsubScalar,	/* 0x7E202C00UQSUB */
585	//	CMHI	//vcmhiregister_Scalar,	/* 0x7E203400CMHI */
586	//	CMHS	//vcmhsregister_Scalar,	/* 0x7E203C00CMHS */
587	//	USHL	//vushlScalar,	/* 0x7E204400USHL */
588	//	UQSHL	//vuqshlregister_Scalar,	/* 0x7E204C00UQSHL */
589	//	URSHL	//vurshlScalar,	/* 0x7E205400URSHL */
590	//	UQRSHL	//vuqrshlScalar,	/* 0x7E205C00UQRSHL */
591	//	SUB	//vsubvector_Scalar,	/* 0x7E208400SUB */
592	//	CMEQ	//vcmeqregister_Scalar,	/* 0x7E208C00CMEQ */
593	//	SQRDMULH	//vsqrdmulhvector_Scalar,	/* 0x7E20B400SQRDMULH */
594	//	FCMGE	//vfcmgeregister_Scalar,	/* 0x7E20E400FCMGE */
595	//	FACGE	//vfacgeScalar,	/* 0x7E20EC00FACGE */
596	//	FABD	//vfabdScalar,	/* 0x7EA0D400FABD */
597	//	FCMGT	//vfcmgtregister_Scalar,	/* 0x7EA0E400FCMGT */
598	//	FACGT	//vfacgtScalar,	/* 0x7EA0EC00FACGT */
599	//	**AdvSIMD scalar three differer**	/* AdvSIMD scalar three different */	
600	//	SQDMLAL	//vsqdmlalvector_Scalar,	/* 0x5E209000SQDMLAL writes to low half of the dest. register &
601	//	SQDMLAL2	//vsqdmlal2vector_Scalar,	/* 0x5E209000SQDMLAL2 writes to high half of the dest. registe
602	//	SQDMLSL	//vsqdmlslvector_Scalar,	/* 0x5E20B000SQDMLSL writes to low half of the dest. register &
603	//	SQDMLSL2	//vsqdmlsl2vector_Scalar,	/* 0x5E20B000SQDMLSL2 writes to high half of the dest. registe
604	//	SQDMULL	//vsqdmullvector_Scalar,	/* 0x5E20D000SQDMULL writes to low half of the dest. register &
605	//	SQDMULL2	//vsqdmull2vector_Scalar,	/* 0x5E20D000SQDMULL2 writes to high half of the dest. registe
606	//	**AdvSIMD scalar two-reg misc**	/* AdvSIMD scalar two-reg misc */	
607	//	SUQADD	//vsuqaddScalar,	/* 0x5E203800SUQADD */
608	//	SQABS	//vsqabsScalar,	/* 0x5E207800SQABS */
609	//	CMGT	//vcmgtzero_Scalar,	/* 0x5E208800CMGT */
610	//	CMEQ	//vcmeqzero_Scalar,	/* 0x5E209800CMEQ */
611	//	CMLT	//vcmltzero_Scalar,	/* 0x5E20A800CMLT */
612	//	ABS	//vabsScalar,	/* 0x5E20B800ABS */
613	//	SQXTN	//vsqxtnScalar,	/* 0x5E214800SQXTN writes to low half of the dest. register & clear
614	//	SQXTN2	//vsqxtn2Scalar,	/* 0x5E214800SQXTN2 writes to high half of the dest. register & do
615	//	FCVTNS	//vfcvtnsvector_Scalar,	/* 0x5E21A800FCVTNS */
616	//	FCVTMS	//vfcvtmsvector_Scalar,	/* 0x5E21B800FCVTMS */
617	//	FCVTAS	//vfcvtasvector_Scalar,	/* 0x5E21C800FCVTAS */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
618	//	SCVTF	//vscvtfvector_integer_Scalar,	/* 0x5E21D800SCVTF */
619	//	FCMGT	//vfcmgtzero_Scalar,	/* 0x5EA0C800FCMGT */
620	//	FCMEQ	//vfcmeqzero_Scalar,	/* 0x5EA0D800FCMEQ */
621	//	FCMLT	//vfcmltzero_Scalar,	/* 0x5EA0E800FCMLT */
622	//	FCVTPS	//vfcvtpsvector_Scalar,	/* 0x5EA1A800FCVTPS */
623	//	FCVTZS	//vfcvtzsvector_integer_Scalar,	/* 0x5EA1B800FCVTZS */
624	//	FRECPE	//vfrecpeScalar,	/* 0x5EA1D800FRECPE */
625	//	FRECPX	//frecpx,	/* 0x5EA1F800FRECPX */
626	//	USQADD	//vusqaddScalar,	/* 0x7E203800USQADD */
627	//	SQNEG	//vsqnegScalar,	/* 0x7E207800SQNEG */
628	//	CMGE	//vcmgezero_Scalar,	/* 0x7E208800CMGE */
629	//	CMLE	//vcmlezero_Scalar,	/* 0x7E209800CMLE */
630	//	NEG	//vnegvector_Scalar,	/* 0x7E20B800NEG */
631	//	SQXTUN	//vsqxtunScalar,	/* 0x7E212800SQXTUN writes to low half of the dest. register & clea
632	//	SQXTUN2	//vsqxtun2Scalar,	/* 0x7E212800SQXTUN2 writes to high half of the dest. register & d
633	//	UQXTN	//vuqxtnScalar,	/* 0x7E214800UQXTN writes to low half of the dest. register & clear
634	//	UQXTN2	//vuqxtn2Scalar,	/* 0x7E214800UQXTN2 writes to high half of the dest. register & do
635	//	FCVTXN	//vfcvtxnScalar,	/* 0x7E216800FCVTXN writes to low half of the dest. register & clear
636	//	FCVTXN2	//vfcvtxn2Scalar,	/* 0x7E216800FCVTXN2 writes to high half of the dest. register & do
637	//	FCVTNU	//vfcvtnuvector_Scalar,	/* 0x7E21A800FCVTNU */
638	//	FCVTMU	//vfcvtmuvector_Scalar,	/* 0x7E21B800FCVTMU */
639	//	FCVTAU	//vfcvtauvector_Scalar,	/* 0x7E21C800FCVTAU */
640	//	UCVTF	//vucvtfvector_integer_Scalar,	/* 0x7E21D800UCVTF */
641	//	FCMGE	//vfcmgezero_Scalar,	/* 0x7EA0C800FCMGE */
642	//	FCMLE	//vfcmlezero_Scalar,	/* 0x7EA0D800FCMLE */
643	//	FCVTPU	//vfcvtpuvector_Scalar,	/* 0x7EA1A800FCVTPU */
644	//	FCVTZU	//vfcvtzuvector_integer_Scalar,	/* 0x7EA1B800FCVTZU */
645	//	FRSQRTE	//vfrsqrteScalar,	/* 0x7EA1D800FRSQRTE */
646	//	**AdvSIMD scalar pairwise**	/* AdvSIMD scalar pairwise */	
647	//	ADDP	//addpscalar,	/* 0x5E31B800ADDP */
648	//	FMAXNMP	//fmaxnmpscalar,	/* 0x7E30C800FMAXNMP */
649	//	FADDP	//faddpscalar,	/* 0x7E30D800FADDP */
650	//	FMAXP	//fmaxpscalar,	/* 0x7E30F800FMAXP */
651	//	FMINNMP	//fminnmpscalar,	/* 0x7EB0C800FMINNMP */

```
1   in_use   Opcode                    //Opcode                                  BINARY    OPCODE    comments
652 //       FMINP                     //fminpscalar,                          /* 0x7EB0F800FMINP      */
653 //     AdvSIMD scalar copy         /* AdvSIMD scalar copy */
654 //       DUP                       //vdupelement_Scalar,                     /* 0x5E000400DUP        */
655 //     AdvSIMD scalar x indexed ele/* AdvSIMD scalar x indexed element */
656 //       SQDMLAL                   //vsqdmlalby_element_Scalar,              /* 0x5F003000SQDMLAL    */
657 //       SQDMLAL2                  //vsqdmlal2by_element_Scalar,             /* 0x5F003000SQDMLAL2   */
658 //       SQDMLSL                   //vsqdmlslby_element_Scalar,              /* 0x5F007000SQDMLSL    */
659 //       SQDMLSL2                  //vsqdmlsl2by_element_Scalar,             /* 0x5F007000SQDMLSL2   */
660 //       SQDMULL                   //vsqdmullby_element_Scalar,              /* 0x5F00B000SQDMULL    */
661 //       SQDMULL2                  //vsqdmull2by_element_Scalar,             /* 0x5F00B000SQDMULL2   */
662 //       SQDMULH                   //vsqdmulhby_element_Scalar,              /* 0x5F00C000SQDMULH    */
663 //       SQRDMULH                  //vsqrdmulhby_element_Scalar,             /* 0x5F00D000SQRDMULH   */
664 //       FMLA                      //vfmlaby_element_Scalar,                 /* 0x5F801000FMLA       */
665 //       FMLS                      //vfmlsby_element_Scalar,                 /* 0x5F805000FMLS       */
666 //       FMUL                      //vfmulby_element_Scalar,                 /* 0x5F809000FMUL       */
667 //       FMULX                     //vfmulxby_element_Scalar,                /* 0x7F809000FMULX      */
668 //     AdvSIMD scalar shift by imme/* AdvSIMD scalar shift by immediate */
669 //       SSHR                      //vsshrScalar,                          /* 0x5F000400SSHR      immh != 0000 */
670 //       SSRA                      //vssraScalar,                          /* 0x5F001400SSRA      immh != 0000 */
671 //       SRSHR                     //vsrshrScalar,                         /* 0x5F002400SRSHR     immh != 0000 */
672 //       SRSRA                     //vsrsraScalar,                         /* 0x5F003400SRSRA     immh != 0000 */
673 //       SHL                       //vshlScalar,                           /* 0x5F005400SHL       immh != 0000 */
674 //       SQSHL                     //vsqshlimmediate_Scalar,                 /* 0x5F007400SQSHL     immh != 0000 */
675 //       SQSHRN                    //vsqshrnScalar,                        /* 0x5F009400SQSHRN    immh != 0000 */
676 //       SQSHRN2                   //vsqshrn2Scalar,                       /* 0x5F009400SQSHRN2   immh != 0000 */
677 //       SQRSHRN                   //vsqrshrnScalar,                       /* 0x5F009C00SQRSHRN   immh != 0000 */
678 //       SQRSHRN2                  //vsqrshrn2Scalar,                      /* 0x5F009C00SQRSHRN2  immh != 0000 */
679 //       SCVTF                     //vscvtfvector_fixed_point_Scalar,        /* 0x5F00E400SCVTF     immh != 0000 */
680 //       FCVTZS                    //vfcvtzsvector_fixed_point_Scalar,       /* 0x5F00FC00FCVTZS    immh != 0000 */
681 //       USHR                      //vushrScalar,                          /* 0x7F000400USHR      immh != 0000 */
682 //       USRA                      //vusraScalar,                          /* 0x7F001400USRA      immh != 0000 */
683 //       URSHR                     //vurshrScalar,                         /* 0x7F002400URSHR     immh != 0000 */
684 //       URSRA                     //vursraScalar,                         /* 0x7F003400URSRA     immh != 0000 */
685 //       SRI                       //vsriScalar,                           /* 0x7F004400SRI       immh != 0000 */
```

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
686	//	SLI	//vsliScalar,	/* 0x7F005400SLI immh != 0000 */
687	//	SQSHLU	//vsqshluScalar,	/* 0x7F006400SQSHLU immh != 0000 */
688	//	UQSHL	//vuqshlimmediate_Scalar,	/* 0x7F007400UQSHL immh != 0000 */
689	//	SQSHRUN	//vsqshrunScalar,	/* 0x7F008400SQSHRUN immh != 0000 */
690	//	SQSHRUN2	//vsqshrun2Scalar,	/* 0x7F008400SQSHRUN2 immh != 0000 */
691	//	SQRSHRUN	//vsqrshrunScalar,	/* 0x7F008C00SQRSHRUN immh != 0000 */
692	//	SQRSHRUN2	//vsqrshrun2Scalar,	/* 0x7F008C00SQRSHRUN2 immh != 0000 */
693	//	UQSHRN	//vuqshrnScalar,	/* 0x7F009400UQSHRN immh != 0000 */
694	//	UQRSHRN	//vuqrshrnScalar,	/* 0x7F009C00UQRSHRN immh != 0000 */
695	//	UQRSHRN2	//vuqrshrn2Scalar,	/* 0x7F009C00UQRSHRN2 immh != 0000 */
696	//	UCVTF	//vucvtfvector_fixed_point_Scalar,	/* 0x7F00E400UCVTF immh != 0000 */
697	//	FCVTZU	//vfcvtzuvector_fixed_point_Scalar,	/* 0x7F00FC00FCVTZU immh != 0000 */
698	//	**Crypto three-reg SHA**	/* Crypto three-reg SHA */	
699	//	SHA1C	//sha1c,	/* 0x5E000000SHA1C */
700	//	SHA1P	//sha1p,	/* 0x5E001000SHA1P */
701	//	SHA1M	//sha1m,	/* 0x5E002000SHA1M */
702	//	SHA1SU0	//sha1su0,	/* 0x5E003000SHA1SU0 */
703	//	SHA256H	//sha256h,	/* 0x5E004000SHA256H */
704	//	SHA256H2	//sha256h2,	/* 0x5E005000SHA256H2 */
705	//	SHA256SU1	//sha256su1,	/* 0x5E006000SHA256SU1 */
706	//	**Crypto two-reg SHA**	/* Crypto two-reg SHA */	
707	//	SHA1H	//sha1h,	/* 0x5E280800SHA1H */
708	//	SHA1SU1	//sha1su1,	/* 0x5E281800SHA1SU1 */
709	//	SHA256SU0	//sha256su0,	/* 0x5E282800SHA256SU0 */
710	//	**Crypto AES**	/* Crypto AES */	
711	//	AESE	//aese,	/* 0x4E284800AESE */
712	//	AESD	//aesd,	/* 0x4E285800AESD */
713	//	AESMC	//aesmc,	/* 0x4E286800AESMC */
714	//	AESIMC	//aesimc,	/* 0x4E287800AESIMC */
715	//	**AdvSIMD three same**	/* AdvSIMD three same */	
716	//	SHADD	//shadd,	/* 0x0E200400SHADD */
717	//	SQADD	//vsqaddVector,	/* 0x0E200C00SQADD */
718	//	SRHADD	//srhadd,	/* 0x0E201400SRHADD */
719	//	SHSUB	//shsub,	/* 0x0E202400SHSUB */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
720	//	SQSUB	//vsqsubVector,	/* 0x0E202C00SQSUB */
721	//	CMGT	//vcmgtregister_Vector,	/* 0x0E203400CMGT */
722	//	CMGE	//vcmgeregister_Vector,	/* 0x0E203C00CMGE */
723	//	SSHL Vector	//sshl vector,	/* 0x0E204400SSHL Vecto */
724	//	SQSHL	//vsqshlregister_Vector,	/* 0x0E204C00SQSHL */
725	//	SRSHL	//vsrshlVector,	/* 0x0E205400SRSHL */
726	//	SQRSHL	//vsqrshlVector,	/* 0x0E205C00SQRSHL */
727	//	SMAX	//smax,	/* 0x0E206400SMAX */
728	//	SMIN	//smin,	/* 0x0E206C00SMIN */
729	//	SABD	//sabd,	/* 0x0E207400SABD */
730	//	SABA	//saba,	/* 0x0E207C00SABA */
731	//	ADD	//vaddvector_Vector,	/* 0x0E208400ADD */
732	//	CMTST	//vcmtstVector,	/* 0x0E208C00CMTST */
733	//	MLA	//mlavector,	/* 0x0E209400MLA */
734	//	MUL	//mulvector,	/* 0x0E209C00MUL */
735	//	SMAXP	//smaxp,	/* 0x0E20A400SMAXP */
736	//	SMINP	//sminp,	/* 0x0E20AC00SMINP */
737	//	SQDMULH	//vsqdmulhvector_Vector,	/* 0x0E20B400SQDMULH */
738	//	ADDP	//addpvector,	/* 0x0E20BC00ADDP */
739	//	FMAXNM	//fmaxnmvector,	/* 0x0E20C400FMAXNM */
740	//	FMLA	//fmlavector,	/* 0x0E20CC00FMLA */
741	//	FADD	//faddvector,	/* 0x0E20D400FADD */
742	//	FMULX	//vfmulxVector,	/* 0x0E20DC00FMULX */
743	//	FCMEQ	//vfcmeqregister_Vector,	/* 0x0E20E400FCMEQ */
744	//	FMAX	//fmaxvector,	/* 0x0E20F400FMAX */
745	//	FRECPS	//vfrecpsVector,	/* 0x0E20FC00FRECPS */
746	//	AND	//andvector,	/* 0x0E201C00AND */
747	//	BIC	//bicvector_register,	/* 0x0E601C00BIC */
748	//	FMINNM	//fminnmvector,	/* 0x0EA0C400FMINNM */
749	//	FMLS	//fmlsvector,	/* 0x0EA0CC00FMLS */
750	//	FSUB	//fsubvector,	/* 0x0EA0D400FSUB */
751	//	FMIN	//fminvector,	/* 0x0EA0F400FMIN */
752	//	FRSQRTS	//vfrsqrtsVector,	/* 0x0EA0FC00FRSQRTS */
753	//	ORR	//orrvector_register,	/* 0x0EA01C00ORR */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
754	//	ORN	//ornvector,	/* 0x0EE01C00ORN */
755	//	UHADD	//uhadd,	/* 0x2E200400UHADD */
756	//	UQADD	//vuqaddVector,	/* 0x2E200C00UQADD */
757	//	URHADD	//urhadd,	/* 0x2E201400URHADD */
758	//	UHSUB	//uhsub,	/* 0x2E202400UHSUB */
759	//	UQSUB	//vuqsubVector,	/* 0x2E202C00UQSUB */
760	//	CMHI	//vcmhiregister_Vector,	/* 0x2E203400CMHI */
761	//	CMHS	//vcmhsregister_Vector,	/* 0x2E203C00CMHS */
762	//	USHL	//vushlVector,	/* 0x2E204400USHL */
763	//	UQSHL	//vuqshlregister_Vector,	/* 0x2E204C00UQSHL */
764	//	URSHL	//vurshlVector,	/* 0x2E205400URSHL */
765	//	UQRSHL	//vuqrshlVector,	/* 0x2E205C00UQRSHL */
766	//	UMAX	//umax,	/* 0x2E206400UMAX */
767	//	UMIN	//umin,	/* 0x2E206C00UMIN */
768	//	UABD	//uabd,	/* 0x2E207400UABD */
769	//	UABA	//uaba,	/* 0x2E207C00UABA */
770	//	SUB	//vsubvector_Vector,	/* 0x2E208400SUB */
771	//	CMEQ	//vcmeqregister_Vector,	/* 0x2E208C00CMEQ */
772	//	MLS	//mlsvector,	/* 0x2E209400MLS */
773	//	PMUL	//pmul,	/* 0x2E209C00PMUL */
774	//	UMAXP	//umaxp,	/* 0x2E20A400UMAXP */
775	//	UMINP	//uminp,	/* 0x2E20AC00UMINP */
776	//	SQRDMULH	//vsqrdmulhvector_Vector,	/* 0x2E20B400SQRDMULH */
777	//	FMAXNMP	//fmaxnmpvector,	/* 0x2E20C400FMAXNMP */
778	//	FADDP	//faddpvector,	/* 0x2E20D400FADDP */
779	//	FMUL	//fmulvector,	/* 0x2E20DC00FMUL */
780	//	FCMGE	//vfcmgeregister_Vector,	/* 0x2E20E400FCMGE */
781	//	FACGE	//vfacgeVector,	/* 0x2E20EC00FACGE */
782	//	FMAXP	//fmaxpvector,	/* 0x2E20F400FMAXP */
783	//	FDIV	//fdivvector,	/* 0x2E20FC00FDIV */
784	//	EOR	//eorvector,	/* 0x2E201C00EOR */
785	//	BSL	//bsl,	/* 0x2E601C00BSL */
786	//	FMINNMP	//fminnmpvector,	/* 0x2EA0C400FMINNMP */
787	//	FABD	//vfabdVector,	/* 0x2EA0D400FABD */

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
788	//	FCMGT	//vfcmgtregister_Vector,	/* 0x2EA0E400FCMGT */
789	//	FACGT	//vfacgtVector,	/* 0x2EA0EC00FACGT */
790	//	FMINP	//fminpvector,	/* 0x2EA0F400FMINP */
791	//	BIT	//bit,	/* 0x2EA01C00BIT */
792	//	BIF	//bif,	/* 0x2EE01C00BIF */
793	//	**AdvSIMD three different**	/* AdvSIMD three different */	
794	//	SADDL	//saddl,	/* 0x0E200000SADDL writes to low half of the dest. register & clears th
795	//	SADDL2	//saddl2,	/* 0x4E200000SADDL2 writes to high half of the dest. register & don't t
796	//	SADDW	//saddw,	/* 0x0E201000SADDW writes to low half of the dest. register & clears
797	//	SADDW2	//saddw2,	/* 0x4E201000SADDW2 writes to high half of the dest. register & don'
798	//	SSUBL	//ssubl,	/* 0x0E202000SSUBL writes to low half of the dest. register & clears th
799	//	SSUBL2	//ssubl2,	/* 0x4E202000SSUBL2 writes to high half of the dest. register & don't t
800	//	SSUBW	//ssubw,	/* 0x0E203000SSUBW writes to low half of the dest. register & clears
801	//	SSUBW2	//ssubw2,	/* 0x4E203000SSUBW2 writes to high half of the dest. register & don't
802	//	ADDHN	//addhn,	/* 0x0E204000ADDHN writes to low half of the dest. register & clears t
803	//	ADDHN2	//addhn2,	/* 0x4E204000ADDHN2 writes to high half of the dest. register & don't
804	//	SABAL	//sabal,	/* 0x0E205000SABAL writes to low half of the dest. register & clears th
805	//	SABAL2	//sabal2,	/* 0x4E205000SABAL2 writes to high half of the dest. register & don't t
806	//	SUBHN	//subhn,	/* 0x0E206000SUBHN writes to low half of the dest. register & clears t
807	//	SUBHN2	//subhn2,	/* 0x4E206000SUBHN2 writes to high half of the dest. register & don't
808	//	SABDL	//sabdl,	/* 0x0E207000SABDL writes to low half of the dest. register & clears th
809	//	SABDL2	//sabdl2,	/* 0x4E207000SABDL2 writes to high half of the dest. register & don't t
810	//	SMLAL	//smlalvector,	/* 0x0E208000SMLAL writes to low half of the dest. register & clears
811	//	SMLAL2	//smlal2vector,	/* 0x4E208000SMLAL2 writes to high half of the dest. register & don'
812	//	SQDMLAL	//vsqdmlalvector_Vector,	/* 0x0E209000SQDMLAL writes to low half of the dest. register &
813	//	SQDMLAL2	//vsqdmlal2vector_Vector,	/* 0x4E209000SQDMLAL2 writes to high half of the dest. registe
814	//	SMLSL	//smlslvector,	/* 0x0E20A000SMLSL writes to low half of the dest. register & clears
815	//	SMLSL2	//smlsl2vector,	/* 0x4E20A000SMLSL2 writes to high half of the dest. register & don'
816	//	SQDMLSL	//vsqdmlslvector_Vector,	/* 0x0E20B000SQDMLSL writes to low half of the dest. register &
817	//	SQDMLSL2	//vsqdmlsl2vector_Vector,	/* 0x4E20B000SQDMLSL2 writes to high half of the dest. registe
818	//	SMULL	//smullvector,	/* 0x0E20C000SMULL writes to low half of the dest. register & clears
819	//	SMULL2	//smull2vector,	/* 0x4E20C000SMULL2 writes to high half of the dest. register & don
820	//	SQDMULL	//vsqdmullvector_Vector,	/* 0x0E20D000SQDMULL writes to low half of the dest. register &
821	//	SQDMULL2	//vsqdmull2vector_Vector,	/* 0x4E20D000SQDMULL2 writes to high half of the dest. registe

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
822	//	PMULL	//pmull,	/* 0x0E20E000PMULL writes to low half of the dest. register & clears th
823	//	PMULL2	//pmull2,	/* 0x4E20E000PMULL2 writes to high half of the dest. register & don't t
824	//	UADDL	//uaddl,	/* 0x2E200000UADDL writes to low half of the dest. register & clears th
825	//	UADDL2	//uaddl2,	/* 0x6E200000UADDL2 writes to high half of the dest. register & don't t
826	//	UADDW	//uaddw,	/* 0x2E201000UADDW writes to low half of the dest. register & clears
827	//	UADDW2	//uaddw2,	/* 0x6E201000UADDW2 writes to high half of the dest. register & don'
828	//	USUBL	//usubl,	/* 0x2E202000USUBL writes to low half of the dest. register & clears th
829	//	USUBL2	//usubl2,	/* 0x6E202000USUBL2 writes to high half of the dest. register & don't t
830	//	USUBW	//usubw,	/* 0x2E203000USUBW writes to low half of the dest. register & clears
831	//	USUBW2	//usubw2,	/* 0x6E203000USUBW2 writes to high half of the dest. register & don'
832	//	RADDHN	//raddhn,	/* 0x2E204000RADDHN writes to low half of the dest. register & clears
833	//	RADDHN2	//raddhn2,	/* 0x6E204000RADDHN2 writes to high half of the dest. register & don
834	//	UABAL	//uabal,	/* 0x2E205000UABAL writes to low half of the dest. register & clears th
835	//	UABAL2	//uabal2,	/* 0x6E205000UABAL2 writes to high half of the dest. register & don't t
836	//	RSUBHN	//rsubhn,	/* 0x2E206000RSUBHN writes to low half of the dest. register & clears
837	//	RSUBHN2	//rsubhn2,	/* 0x6E206000RSUBHN2 writes to high half of the dest. register & don'
838	//	UABDL	//uabdl,	/* 0x2E207000UABDL writes to low half of the dest. register & clears th
839	//	UABDL2	//uabdl2,	/* 0x6E207000UABDL2 writes to high half of the dest. register & don't t
840	//	UMLAL	//umlalvector,	/* 0x2E208000UMLAL writes to low half of the dest. register & clears
841	//	UMLAL2	//umlal2vector,	/* 0x6E208000UMLAL2 writes to high half of the dest. register & don
842	//	UMLSL	//umlslvector,	/* 0x2E20A000UMLSL writes to low half of the dest. register & clears
843	//	UMLSL2	//umlsl2vector,	/* 0x6E20A000UMLSL2 writes to high half of the dest. register & don
844	//	UMULL	//umullvector,	/* 0x2E20C000UMULL writes to low half of the dest. register & clears
845	//	UMULL2	//umull2vector,	/* 0x6E20C000UMULL2 writes to high half of the dest. register & dor
846	//	**AdvSIMD two-reg misc**	/* AdvSIMD two-reg misc */	
847	//	REV64	//rev64,	/* 0x0E200800REV64 */
848	//	REV16	//rev16vector,	/* 0x0E201800REV16 */
849	//	SADDLP	//saddlp,	/* 0x0E202800SADDLP */
850	//	SUQADD	//vsuqaddVector,	/* 0x0E203800SUQADD */
851	//	CLS	//clsvector,	/* 0x0E204800CLS */
852	//	CNT	//cnt,	/* 0x0E205800CNT */
853	//	SADALP	//sadalp,	/* 0x0E206800SADALP */
854	//	SQABS	//vsqabsVector,	/* 0x0E207800SQABS */
855	//	CMGT	//vcmgtzero_Vector,	/* 0x0E208800CMGT */

1	in_use	Opcode	//Opcode	BINARY OPCODE	comments
856	//	CMEQ	//vcmeqzero_Vector,	/* 0x0E209800CMEQ */	
857	//	CMLT	//vcmltzero_Vector,	/* 0x0E20A800CMLT */	
858	//	ABS	//vabsVector,	/* 0x0E20B800ABS */	
859	//	XTN	//xtn,	/* 0x0E212800XTN */	
860	//	XTN2	//xtn2,	/* 0x0E212800XTN2 */	
861	//	SQXTN	//vsqxtnVector,	/* 0x0E214800SQXTN */	
862	//	SQXTN2	//vsqxtn2Vector,	/* 0x0E214800SQXTN2 */	
863	//	FCVTN	//fcvtn,	/* 0x0E216800FCVTN */	
864	//	FCVTN2	//fcvtn2,	/* 0x0E216800FCVTN2 */	
865	//	FCVTL	//fcvtl,	/* 0x0E217800FCVTL */	
866	//	FCVTL2	//fcvtl2,	/* 0x0E217800FCVTL2 */	
867	//	FRINTN	//frintnvector,	/* 0x0E218800FRINTN */	
868	//	FRINTM	//frintmvector,	/* 0x0E219800FRINTM */	
869	//	FCVTNS	//vfcvtnsvector_Vector,	/* 0x0E21A800FCVTNS */	
870	//	FCVTMS	//vfcvtmsvector_Vector,	/* 0x0E21B800FCVTMS */	
871	//	FCVTAS	//vfcvtasvector_Vector,	/* 0x0E21C800FCVTAS */	
872	//	SCVTF	//vscvtfvector_integer_Vector,	/* 0x0E21D800SCVTF */	
873	//	FCMGT	//vfcmgtzero_Vector,	/* 0x0EA0C800FCMGT */	
874	//	FCMEQ	//vfcmeqzero_Vector,	/* 0x0EA0D800FCMEQ */	
875	//	FCMLT	//vfcmltzero_Vector,	/* 0x0EA0E800FCMLT */	
876	//	FABS	//fabsvector,	/* 0x0EA0F800FABS */	
877	//	FRINTP	//frintpvector,	/* 0x0EA18800FRINTP */	
878	//	FRINTZ	//frintzvector,	/* 0x0EA19800FRINTZ */	
879	//	FCVTPS	//vfcvtpsvector_Vector,	/* 0x0EA1A800FCVTPS */	
880	//	FCVTZS	//vfcvtzsvector_integer_Vector,	/* 0x0EA1B800FCVTZS */	
881	//	URECPE	//urecpe,	/* 0x0EA1C800URECPE */	
882	//	FRECPE	//vfrecpeVector,	/* 0x0EA1D800FRECPE */	
883	//	REV32	//rev32vector,	/* 0x2E200800REV32 */	
884	//	UADDLP	//uaddlp,	/* 0x2E202800UADDLP */	
885	//	USQADD	//vusqaddVector,	/* 0x2E203800USQADD */	
886	//	CLZ	//clzvector,	/* 0x2E204800CLZ */	
887	//	UADALP	//uadalp,	/* 0x2E206800UADALP */	
888	//	SQNEG	//vsqnegVector,	/* 0x2E207800SQNEG */	
889	//	CMGE	//vcmgezero_Vector,	/* 0x2E208800CMGE */	

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
890	//	CMLE	//vcmlezero_Vector,	/* 0x2E209800CMLE */		
891	//	NEG	//vnegvector_Vector,	/* 0x2E20B800NEG */		
892	//	SQXTUN	//vsqxtunVector,	/* 0x2E212800SQXTUN */		
893	//	SQXTUN2	//vsqxtun2Vector,	/* 0x2E212800SQXTUN2 */		
894	//	SHLL	//shll,	/* 0x2E213800SHLL */		
895	//	SHLL2	//shll2,	/* 0x2E213800SHLL2 */		
896	//	UQXTN	//vuqxtnVector,	/* 0x2E214800UQXTN */		
897	//	UQXTN2	//vuqxtn2Vector,	/* 0x2E214800UQXTN2 */		
898	//	FCVTXN	//vfcvtxnVector,	/* 0x2E216800FCVTXN */		
899	//	FCVTXN2	//vfcvtxn2Vector,	/* 0x2E216800FCVTXN2 */		
900	//	FRINTA	//frintavector,	/* 0x2E218800FRINTA */		
901	//	FRINTX	//frintxvector,	/* 0x2E219800FRINTX */		
902	//	FCVTNU	//vfcvtnuvector_Vector,	/* 0x2E21A800FCVTNU */		
903	//	FCVTMU	//vfcvtmuvector_Vector,	/* 0x2E21B800FCVTMU */		
904	//	FCVTAU	//vfcvtauvector_Vector,	/* 0x2E21C800FCVTAU */		
905	//	UCVTF	//vucvtfvector_integer_Vector,	/* 0x2E21D800UCVTF */		
906	//	NOT	//not,	/* 0x2E205800NOT */		
907	//	RBIT	//rbitvector,	/* 0x2E605800RBIT */		
908	//	FCMGE	//vfcmgezero_Vector,	/* 0x2EA0C800FCMGE */		
909	//	FCMLE	//vfcmlezero_Vector,	/* 0x2EA0D800FCMLE */		
910	//	FNEG	//fnegvector,	/* 0x2EA0F800FNEG */		
911	//	FRINTI	//frintivector,	/* 0x2EA19800FRINTI */		
912	//	FCVTPU	//vfcvtpuvector_Vector,	/* 0x2EA1A800FCVTPU */		
913	//	FCVTZU	//vfcvtzuvector_integer_Vector,	/* 0x2EA1B800FCVTZU */		
914	//	URSQRTE	//ursqrte,	/* 0x2EA1C800URSQRTE */		
915	//	FRSQRTE	//vfrsqrteVector,	/* 0x2EA1D800FRSQRTE */		
916	//	FSQRT	//fsqrtvector,	/* 0x2EA1F800FSQRT */		
917	//	**AdvSIMD across lanes**	/* AdvSIMD across lanes */			
918	//	SADDLV	//saddlv,	/* 0x0E303800SADDLV */		
919	//	SMAXV	//smaxv,	/* 0x0E30A800SMAXV */		
920	//	SMINV	//sminv,	/* 0x0E31A800SMINV */		
921	//	ADDV	//addv,	/* 0x0E31B800ADDV */		
922	//	UADDLV	//uaddlv,	/* 0x2E303800UADDLV */		
923	//	UMAXV	//umaxv,	/* 0x2E30A800UMAXV */		

1	in_use	Opcode	//Opcode	BINARY OPCODE	comments
924	//	UMINV	//uminv,	/* 0x2E31A800UMINV */	
925	//	FMAXNMV	//fmaxnmv,	/* 0x2E30C800FMAXNMV */	
926	//	FMAXV	//fmaxv,	/* 0x2E30F800FMAXV */	
927	//	FMINNMV	//fminnmv,	/* 0x2EB0C800FMINNMV */	
928	//	FMINV	//fminv,	/* 0x2EB0F800FMINV */	
929	//	**AdvSIMD copy**	/* AdvSIMD copy */		
930	//	DUP	//vdupelement_Vector,	/* 0x0E000400DUP */	
931	//	DUP	//dupgeneral,	/* 0x0E000C00DUP */	
932	//	SMOV	//vsmov32_bit,	/* 0x0E002C00SMOV */	
933	//	UMOV	//vumov32_bit,	/* 0x0E003C00UMOV */	
934	//	INS	//insgeneral,	/* 0x4E001C00INS */	
935	//	SMOV	//vsmov64_bit,	/* 0x4E002C00SMOV */	
936	//	UMOV	//vumov64_bit,	/* 0x4E003C00UMOV */	
937	//	INS	//inselement,	/* 0x6E000400INS */	
938	//	**AdvSIMD vector x indexed ele**	/* AdvSIMD vector x indexed element */		
939	//	SMLAL	//smlalby_element,	/* 0x0F002000SMLAL */	
940	//	SMLAL2	//smlal2by_element,	/* 0x0F002000SMLAL2 */	
941	//	SQDMLAL	//vsqdmlalby_element_Vector,	/* 0x0F003000SQDMLAL */	
942	//	SQDMLAL2	//vsqdmlal2by_element_Vector,	/* 0x0F003000SQDMLAL2 */	
943	//	SMLSL	//smlslby_element,	/* 0x0F006000SMLSL */	
944	//	SMLSL2	//smlsl2by_element,	/* 0x0F006000SMLSL2 */	
945	//	SQDMLSL	//vsqdmlslby_element_Vector,	/* 0x0F007000SQDMLSL */	
946	//	SQDMLSL2	//vsqdmlsl2by_element_Vector,	/* 0x0F007000SQDMLSL2 */	
947	//	MUL	//mulby_element,	/* 0x0F008000MUL */	
948	//	SMULL	//smullby_element,	/* 0x0F00A000SMULL */	
949	//	SMULL2	//smull2by_element,	/* 0x0F00A000SMULL2 */	
950	//	SQDMULL	//vsqdmullby_element_Vector,	/* 0x0F00B000SQDMULL */	
951	//	SQDMULL2	//vsqdmull2by_element_Vector,	/* 0x0F00B000SQDMULL2 */	
952	//	SQDMULH	//vsqdmulhby_element_Vector,	/* 0x0F00C000SQDMULH */	
953	//	SQRDMULH	//vsqrdmulhby_element_Vector,	/* 0x0F00D000SQRDMULH */	
954	//	FMLA	//vfmlaby_element_Vector,	/* 0x0F801000FMLA */	
955	//	FMLS	//vfmlsby_element_Vector,	/* 0x0F805000FMLS */	
956	//	FMUL	//vfmulby_element_Vector,	/* 0x0F809000FMUL */	
957	//	MLA	//mlaby_element,	/* 0x2F000000MLA */	

1	in_use	Opcode	//Opcode	BINARY OPCODE	comments
958	//	UMLAL	//umlalby_element,	/* 0x2F002000UMLAL */	
959	//	UMLAL2	//umlal2by_element,	/* 0x2F002000UMLAL2 */	
960	//	MLS	//mlsby_element,	/* 0x2F004000MLS */	
961	//	UMLSL	//umlslby_element,	/* 0x2F006000UMLSL */	
962	//	UMLSL2	//umlsl2by_element,	/* 0x2F006000UMLSL2 */	
963	//	UMULL	//umullby_element,	/* 0x2F00A000UMULL */	
964	//	UMULL2	//umull2by_element,	/* 0x2F00A000UMULL2 */	
965	//	FMULX	//vfmulxby_element_Vector,	/* 0x2F809000FMULX */	
966	//	**AdvSIMD modified immediate**	/* AdvSIMD modified immediate */		
967	//	MOVI	//vmovi32_bit_shifted_immediate,	/* 0x0F000400MOVI */	
968	//	ORR	//vorrvector_immediate_32_bit,	/* 0x0F001400ORR */	
969	//	MOVI	//vmovi16_bit_shifted_immediate,	/* 0x0F008400MOVI */	
970	//	ORR	//vorrvector_immediate_16_bit,	/* 0x0F009400ORR */	
971	//	MOVI	//vmovi32_bit_shifting_ones,	/* 0x0F00C400MOVI */	
972	//	MOVI	//vmovi8_bit,	/* 0x0F00E400MOVI */	
973	//	FMOV	//vfmovvector_immediate_Single_precision,	/* 0x0F00F400FMOV */	
974	//	MVNI	//vmvni32_bit_shifted_immediate,	/* 0x2F000400MVNI */	
975	//	BIC	//vbicvector_immediate_32_bit,	/* 0x2F001400BIC */	
976	//	MVNI	//vmvni16_bit_shifted_immediate,	/* 0x2F008400MVNI */	
977	//	BIC	//vbicvector_immediate_16_bit,	/* 0x2F009400BIC */	
978	//	MVNI	//vmvni32_bit_shifting_ones,	/* 0x2F00C400MVNI */	
979	//	MOVI	//vmovi64_bit_scalar,	/* 0x2F00E400MOVI */	
980	//	MOVI	//vmovi64_bit_vector,	/* 0x6F00E400MOVI */	
981	//	FMOV	//vfmovvector_immediate_Double_precision,	/* 0x6F00F400FMOV */	
982	//	**AdvSIMD shift by immediate**	/* AdvSIMD shift by immediate */		
983	//	SSHR	//vsshrVector,	/* 0x0F000400SSHR */	
984	//	SSRA	//vssraVector,	/* 0x0F001400SSRA */	
985	//	SRSHR	//vsrshrVector,	/* 0x0F002400SRSHR */	
986	//	SRSRA	//vsrsraVector,	/* 0x0F003400SRSRA */	
987	//	SHL	//vshlVector,	/* 0x0F005400SHL */	
988	//	SQSHL	//vsqshlimmediate_Vector,	/* 0x0F007400SQSHL */	
989	//	SHRN	//shrn,	/* 0x0F008400SHRN */	
990	//	SHRN2	//shrn2,	/* 0x0F008400SHRN2 */	
991	//	RSHRN	//rshrn,	/* 0x0F008C00RSHRN */	

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
992	//	RSHRN2	//rshrn2,	/* 0x0F008C00RSHRN2 */
993	//	SQSHRN	//vsqshrnVector,	/* 0x0F009400SQSHRN */
994	//	SQSHRN2	//vsqshrn2Vector,	/* 0x0F009400SQSHRN2 */
995	//	SQRSHRN	//vsqrshrnVector,	/* 0x0F009C00SQRSHRN */
996	//	SQRSHRN2	//vsqrshrn2Vector,	/* 0x0F009C00SQRSHRN2 */
997	//	SSHLL	//sshll,	/* 0x0F00A400SSHLL */
998	//	SSHLL2	//sshll2,	/* 0x0F00A400SSHLL2 */
999	//	SCVTF	//vscvtfvector_fixed_point_Vector,	/* 0x0F00E400SCVTF */
1000	//	FCVTZS	//vfcvtzsvector_fixed_point_Vector,	/* 0x0F00FC00FCVTZS */
1001	//	USHR	//vushrVector,	/* 0x2F000400USHR */
1002	//	USRA	//vusraVector,	/* 0x2F001400USRA */
1003	//	URSHR	//vurshrVector,	/* 0x2F002400URSHR */
1004	//	URSRA	//vursraVector,	/* 0x2F003400URSRA */
1005	//	SRI	//vsriVector,	/* 0x2F004400SRI */
1006	//	SLI	//vsliVector,	/* 0x2F005400SLI */
1007	//	SQSHLU	//vsqshluVector,	/* 0x2F006400SQSHLU */
1008	//	UQSHL	//vuqshlimmediate_Vector,	/* 0x2F007400UQSHL */
1009	//	SQSHRUN	//vsqshrunVector,	/* 0x2F008400SQSHRUN */
1010	//	SQSHRUN2	//vsqshrun2Vector,	/* 0x2F008400SQSHRUN2 */
1011	//	SQRSHRUN	//vsqrshrunVector,	/* 0x2F008C00SQRSHRUN */
1012	//	SQRSHRUN2	//vsqrshrun2Vector,	/* 0x2F008C00SQRSHRUN2 */
1013	//	UQSHRN	//vuqshrnVector,	/* 0x2F009400UQSHRN */
1014	//	UQRSHRN	//vuqrshrnVector,	/* 0x2F009C00UQRSHRN */
1015	//	UQRSHRN2	//vuqrshrn2Vector,	/* 0x2F009C00UQRSHRN2 */
1016	//	USHLL	//ushll,	/* 0x2F00A400USHLL */
1017	//	USHLL2	//ushll2,	/* 0x2F00A400USHLL2 */
1018	//	UCVTF	//vucvtfvector_fixed_point_Vector,	/* 0x2F00E400UCVTF */
1019	//	FCVTZU	//vfcvtzuvector_fixed_point_Vector,	/* 0x2F00FC00FCVTZU */
1020	//	**AdvSIMD TBL/TBX**	/* AdvSIMD TBL/TBX */	
1021	//	TBL	//vtblSingle_register_table,	/* 0x0E000000TBL */
1022	//	TBX	//vtbxSingle_register_table,	/* 0x0E001000TBX */
1023	//	TBL	//vtblTwo_register_table,	/* 0x0E002000TBL */
1024	//	TBX	//vtbxTwo_register_table,	/* 0x0E003000TBX */
1025	//	TBL	//vtblThree_register_table,	/* 0x0E004000TBL */

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
1026	//	TBX	//vtbxThree_register_table,	/* 0x0E005000TBX */		
1027	//	TBL	//vtblFour_register_table,	/* 0x0E006000TBL */		
1028	//	TBX	//vtbxFour_register_table,	/* 0x0E007000TBX */		
1029	//	**AdvSIMD ZIP/UZP/TRN**	/* AdvSIMD ZIP/UZP/TRN */			
1030	//	UZP1	//uzp1,	/* 0x0E001800UZP1 */		
1031	//	TRN1	//trn1,	/* 0x0E002800TRN1 */		
1032	//	ZIP1	//zip1,	/* 0x0E003800ZIP1 */		
1033	//	UZP2	//uzp2,	/* 0x0E005800UZP2 */		
1034	//	TRN2	//trn2,	/* 0x0E006800TRN2 */		
1035	//	ZIP2	//zip2,	/* 0x0E007800ZIP2 */		
1036	//	**AdvSIMD EXT**	/* AdvSIMD EXT */			
1037	//	EXT	//ext,	/* 0x2E000000EXT */		
1038	//	**Loads and stores**	/* Loads and stores */			
1039	//	**AdvSIMD load/store multiple s**	/* AdvSIMD load/store multiple structures */			
1040	//	ST4	//vst4multiple_structures_No_offset,	/* 0x0C000000ST4 */		
1041	//	ST1	//vst1multiple_structures_Four_registers,	/* 0x0C002000ST1 */		
1042	//	ST3	//vst3multiple_structures_No_offset,	/* 0x0C004000ST3 */		
1043	//	ST1	//vst1multiple_structures_Three_registers,	/* 0x0C006000ST1 */		
1044	//	ST1	//vst1multiple_structures_One_register,	/* 0x0C007000ST1 */		
1045	//	ST2	//vst2multiple_structures_No_offset,	/* 0x0C008000ST2 */		
1046	//	ST1	//vst1multiple_structures_Two_registers,	/* 0x0C00A000ST1 */		
1047	//	LD4	//vld4multiple_structures_No_offset,	/* 0x0C400000LD4 */		
1048	//	LD1	//vld1multiple_structures_Four_registers,	/* 0x0C402000LD1 */		
1049	//	LD3	//vld3multiple_structures_No_offset,	/* 0x0C404000LD3 */		
1050	//	LD1	//vld1multiple_structures_Three_registers,	/* 0x0C406000LD1 */		
1051	//	LD1	//vld1multiple_structures_One_register,	/* 0x0C407000LD1 */		
1052	//	LD2	//vld2multiple_structures_No_offset,	/* 0x0C408000LD2 */		
1053	//	LD1	//vld1multiple_structures_Two_registers,	/* 0x0C40A000LD1 */		
1054	//	**AdvSIMD load/store multiple s**	/* AdvSIMD load/store multiple structures (post-indexed) */			
1055	//	ST4	//vst4multiple_structures_Register_offset,	/* 0x0C800000ST4 Rm != 11111 */		
1056	//	ST1	//vst1multiple_structures_Four_registers_register_offset,	/* 0x0C802000ST1 Rm != 11111 */		
1057	//	ST3	//vst3multiple_structures_Register_offset,	/* 0x0C804000ST3 Rm != 11111 */		
1058	//	ST1	//vst1multiple_structures_Three_registers_register_offset,/* 0x0C806000ST1 Rm != 11111 */			
1059	//	ST1	//vst1multiple_structures_One_register_register_offset,	/* 0x0C807000ST1 Rm != 11111 */		

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
1060	//	ST2	//vst2multiple_structures_Register_offset,	/* 0x0C808000ST2 Rm != 11111 */
1061	//	ST1	//vst1multiple_structures_Two_registers_register_offset,	/* 0x0C80A000ST1 Rm != 11111 */
1062	//	ST4	//vst4multiple_structures_Immediate_offset,	/* 0x0C9F0000ST4 */
1063	//	ST1	//vst1multiple_structures_Four_registers_immediate_offset,/*	0x0C9F2000ST1 */
1064	//	ST3	//vst3multiple_structures_Immediate_offset,	/* 0x0C9F4000ST3 */
1065	//	ST1	//vst1multiple_structures_Three_registers_immediate_offset/*	0x0C9F6000ST1 */
1066	//	ST1	//vst1multiple_structures_One_register_immediate_offset,	/* 0x0C9F7000ST1 */
1067	//	ST2	//vst2multiple_structures_Immediate_offset,	/* 0x0C9F8000ST2 */
1068	//	ST1	//vst1multiple_structures_Two_registers_immediate_offset,	/* 0x0C9FA000ST1 */
1069	//	LD4	//vld4multiple_structures_Register_offset,	/* 0x0CC00000LD4 Rm != 11111 */
1070	//	LD1	//vld1multiple_structures_Four_registers_register_offset,	/* 0x0CC02000LD1 Rm != 11111 */
1071	//	LD3	//vld3multiple_structures_Register_offset,	/* 0x0CC04000LD3 Rm != 11111 */
1072	//	LD1	//vld1multiple_structures_Three_registers_register_offset,/*	0x0CC06000LD1 Rm != 11111 */
1073	//	LD1	//vld1multiple_structures_One_register_register_offset,	/* 0x0CC07000LD1 Rm != 11111 */
1074	//	LD2	//vld2multiple_structures_Register_offset,	/* 0x0CC08000LD2 Rm != 11111 */
1075	//	LD1	//vld1multiple_structures_Two_registers_register_offset,	/* 0x0CC0A000LD1 Rm != 11111 */
1076	//	LD4	//vld4multiple_structures_Immediate_offset,	/* 0x0CDF0000LD4 */
1077	//	LD1	//vld1multiple_structures_Four_registers_immediate_offset,/*	0x0CDF2000LD1 */
1078	//	LD3	//vld3multiple_structures_Immediate_offset,	/* 0x0CDF4000LD3 */
1079	//	LD1	//vld1multiple_structures_Three_registers_immediate_offset/*	0x0CDF6000LD1 */
1080	//	LD1	//vld1multiple_structures_One_register_immediate_offset,	/* 0x0CDF7000LD1 */
1081	//	LD2	//vld2multiple_structures_Immediate_offset,	/* 0x0CDF8000LD2 */
1082	//	LD1	//vld1multiple_structures_Two_registers_immediate_offset,	/* 0x0CDFA000LD1 */
1083	//	**AdvSIMD load/store single st**	/* AdvSIMD load/store single structure */	
1084	//	ST1	//vst1single_structure_8_bit,	/* 0x0D000000ST1 */
1085	//	ST3	//vst3single_structure_8_bit,	/* 0x0D002000ST3 */
1086	//	ST1	//vst1single_structure_16_bit,	/* 0x0D004000ST1 */
1087	//	ST3	//vst3single_structure_16_bit,	/* 0x0D006000ST3 */
1088	//	ST1	//vst1single_structure_32_bit,	/* 0x0D008000ST1 */
1089	//	ST1	//vst1single_structure_64_bit,	/* 0x0D008400ST1 */
1090	//	ST3	//vst3single_structure_32_bit,	/* 0x0D00A000ST3 */
1091	//	ST3	//vst3single_structure_64_bit,	/* 0x0D00A400ST3 */
1092	//	ST2	//vst2single_structure_8_bit,	/* 0x0D200000ST2 */
1093	//	ST4	//vst4single_structure_8_bit,	/* 0x0D202000ST4 */

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
1094	//	ST2	//vst2single_structure_16_bit,	/* 0x0D204000ST2	*/	
1095	//	ST4	//vst4single_structure_16_bit,	/* 0x0D206000ST4	*/	
1096	//	ST2	//vst2single_structure_32_bit,	/* 0x0D208000ST2	*/	
1097	//	ST2	//vst2single_structure_64_bit,	/* 0x0D208400ST2	*/	
1098	//	ST4	//vst4single_structure_32_bit,	/* 0x0D20A000ST4	*/	
1099	//	ST4	//vst4single_structure_64_bit,	/* 0x0D20A400ST4	*/	
1100	//	LD1	//vld1single_structure_8_bit,	/* 0x0D400000LD1	*/	
1101	//	LD3	//vld3single_structure_8_bit,	/* 0x0D402000LD3	*/	
1102	//	LD1	//vld1single_structure_16_bit,	/* 0x0D404000LD1	*/	
1103	//	LD3	//vld3single_structure_16_bit,	/* 0x0D406000LD3	*/	
1104	//	LD1	//vld1single_structure_32_bit,	/* 0x0D408000LD1	*/	
1105	//	LD1	//vld1single_structure_64_bit,	/* 0x0D408400LD1	*/	
1106	//	LD3	//vld3single_structure_32_bit,	/* 0x0D40A000LD3	*/	
1107	//	LD3	//vld3single_structure_64_bit,	/* 0x0D40A400LD3	*/	
1108	//	LD1R	//vld1rNo_offset,	/* 0x0D40C000LD1R	*/	
1109	//	LD3R	//vld3rNo_offset,	/* 0x0D40E000LD3R	*/	
1110	//	LD2	//vld2single_structure_8_bit,	/* 0x0D600000LD2	*/	
1111	//	LD4	//vld4single_structure_8_bit,	/* 0x0D602000LD4	*/	
1112	//	LD2	//vld2single_structure_16_bit,	/* 0x0D604000LD2	*/	
1113	//	LD4	//vld4single_structure_16_bit,	/* 0x0D606000LD4	*/	
1114	//	LD2	//vld2single_structure_32_bit,	/* 0x0D608000LD2	*/	
1115	//	LD2	//vld2single_structure_64_bit,	/* 0x0D608400LD2	*/	
1116	//	LD4	//vld4single_structure_32_bit,	/* 0x0D60A000LD4	*/	
1117	//	LD4	//vld4single_structure_64_bit,	/* 0x0D60A400LD4	*/	
1118	//	LD2R	//vld2rNo_offset,	/* 0x0D60C000LD2R	*/	
1119	//	LD4R	//vld4rNo_offset,	/* 0x0D60E000LD4R	*/	
1120	//	**AdvSIMD load/store single st**	/* AdvSIMD load/store single structure (post-indexed) */			
1121	//	ST1	//vst1single_structure_8_bit_register_offset,	/* 0x0D800000ST1	Rm != 11111 */	
1122	//	ST3	//vst3single_structure_8_bit_register_offset,	/* 0x0D802000ST3	Rm != 11111 */	
1123	//	ST1	//vst1single_structure_16_bit_register_offset,	/* 0x0D804000ST1	Rm != 11111 */	
1124	//	ST3	//vst3single_structure_16_bit_register_offset,	/* 0x0D806000ST3	Rm != 11111 */	
1125	//	ST1	//vst1single_structure_32_bit_register_offset,	/* 0x0D808000ST1	Rm != 11111 */	
1126	//	ST1	//vst1single_structure_64_bit_register_offset,	/* 0x0D808400ST1	Rm != 11111 */	
1127	//	ST3	//vst3single_structure_32_bit_register_offset,	/* 0x0D80A000ST3	Rm != 11111 */	

1	in_use	Opcode	//Opcode	BINARY OPCODE comments
1128	//	ST3	//vst3single_structure_64_bit_register_offset,	/* 0x0D80A400ST3 Rm != 11111 */
1129	//	ST1	//vst1single_structure_8_bit_immediate_offset,	/* 0x0D9F0000ST1 */
1130	//	ST3	//vst3single_structure_8_bit_immediate_offset,	/* 0x0D9F2000ST3 */
1131	//	ST1	//vst1single_structure_16_bit_immediate_offset,	/* 0x0D9F4000ST1 */
1132	//	ST3	//vst3single_structure_16_bit_immediate_offset,	/* 0x0D9F6000ST3 */
1133	//	ST1	//vst1single_structure_32_bit_immediate_offset,	/* 0x0D9F8000ST1 */
1134	//	ST1	//vst1single_structure_64_bit_immediate_offset,	/* 0x0D9F8400ST1 */
1135	//	ST3	//vst3single_structure_32_bit_immediate_offset,	/* 0x0D9FA000ST3 */
1136	//	ST3	//vst3single_structure_64_bit_immediate_offset,	/* 0x0D9FA400ST3 */
1137	//	ST2	//vst2single_structure_8_bit_register_offset,	/* 0x0DA00000ST2 Rm != 11111 */
1138	//	ST4	//vst4single_structure_8_bit_register_offset,	/* 0x0DA02000ST4 Rm != 11111 */
1139	//	ST2	//vst2single_structure_16_bit_register_offset,	/* 0x0DA04000ST2 Rm != 11111 */
1140	//	ST4	//vst4single_structure_16_bit_register_offset,	/* 0x0DA06000ST4 Rm != 11111 */
1141	//	ST2	//vst2single_structure_32_bit_register_offset,	/* 0x0DA08000ST2 Rm != 11111 */
1142	//	ST2	//vst2single_structure_64_bit_register_offset,	/* 0x0DA08400ST2 Rm != 11111 */
1143	//	ST4	//vst4single_structure_32_bit_register_offset,	/* 0x0DA0A000ST4 Rm != 11111 */
1144	//	ST4	//vst4single_structure_64_bit_register_offset,	/* 0x0DA0A400ST4 Rm != 11111 */
1145	//	ST2	//vst2single_structure_8_bit_immediate_offset,	/* 0x0DBF0000ST2 */
1146	//	ST4	//vst4single_structure_8_bit_immediate_offset,	/* 0x0DBF2000ST4 */
1147	//	ST2	//vst2single_structure_16_bit_immediate_offset,	/* 0x0DBF4000ST2 */
1148	//	ST4	//vst4single_structure_16_bit_immediate_offset,	/* 0x0DBF6000ST4 */
1149	//	ST2	//vst2single_structure_32_bit_immediate_offset,	/* 0x0DBF8000ST2 */
1150	//	ST2	//vst2single_structure_64_bit_immediate_offset,	/* 0x0DBF8400ST2 */
1151	//	ST4	//vst4single_structure_32_bit_immediate_offset,	/* 0x0DBFA000ST4 */
1152	//	ST4	//vst4single_structure_64_bit_immediate_offset,	/* 0x0DBFA400ST4 */
1153	//	LD1	//vld1single_structure_8_bit_register_offset,	/* 0x0DC00000LD1 Rm != 11111 */
1154	//	LD3	//vld3single_structure_8_bit_register_offset,	/* 0x0DC02000LD3 Rm != 11111 */
1155	//	LD1	//vld1single_structure_16_bit_register_offset,	/* 0x0DC04000LD1 Rm != 11111 */
1156	//	LD3	//vld3single_structure_16_bit_register_offset,	/* 0x0DC06000LD3 Rm != 11111 */
1157	//	LD1	//vld1single_structure_32_bit_register_offset,	/* 0x0DC08000LD1 Rm != 11111 */
1158	//	LD1	//vld1single_structure_64_bit_register_offset,	/* 0x0DC08400LD1 Rm != 11111 */
1159	//	LD3	//vld3single_structure_32_bit_register_offset,	/* 0x0DC0A000LD3 Rm != 11111 */
1160	//	LD3	//vld3single_structure_64_bit_register_offset,	/* 0x0DC0A400LD3 Rm != 11111 */
1161	//	LD1R	//vld1rRegister_offset,	/* 0x0DC0C000LD1R Rm != 11111 */

1	in_use	Opcode	//Opcode	BINARY	OPCODE	comments
1162	//	LD3R	//vld3rRegister_offset,	/* 0x0DC0E000	LD3R	Rm != 11111 */
1163	//	LD1	//vld1single_structure_8_bit_immediate_offset,	/* 0x0DDF0000	LD1	*/
1164	//	LD3	//vld3single_structure_8_bit_immediate_offset,	/* 0x0DDF2000	LD3	*/
1165	//	LD1	//vld1single_structure_16_bit_immediate_offset,	/* 0x0DDF4000	LD1	*/
1166	//	LD3	//vld3single_structure_16_bit_immediate_offset,	/* 0x0DDF6000	LD3	*/
1167	//	LD1	//vld1single_structure_32_bit_immediate_offset,	/* 0x0DDF8000	LD1	*/
1168	//	LD1	//vld1single_structure_64_bit_immediate_offset,	/* 0x0DDF8400	LD1	*/
1169	//	LD3	//vld3single_structure_32_bit_immediate_offset,	/* 0x0DDFA000	LD3	*/
1170	//	LD3	//vld3single_structure_64_bit_immediate_offset,	/* 0x0DDFA400	LD3	*/
1171	//	LD1R	//vld1rImmediate_offset,	/* 0x0DDFC000	LD1R	*/
1172	//	LD3R	//vld3rImmediate_offset,	/* 0x0DDFE000	LD3R	*/
1173	//	LD2	//vld2single_structure_8_bit_register_offset,	/* 0x0DE00000	LD2	Rm != 11111 */
1174	//	LD4	//vld4single_structure_8_bit_register_offset,	/* 0x0DE02000	LD4	Rm != 11111 */
1175	//	LD2	//vld2single_structure_16_bit_register_offset,	/* 0x0DE04000	LD2	Rm != 11111 */
1176	//	LD4	//vld4single_structure_16_bit_register_offset,	/* 0x0DE06000	LD4	Rm != 11111 */
1177	//	LD2	//vld2single_structure_32_bit_register_offset,	/* 0x0DE08000	LD2	Rm != 11111 */
1178	//	LD2	//vld2single_structure_64_bit_register_offset,	/* 0x0DE08400	LD2	Rm != 11111 */
1179	//	LD4	//vld4single_structure_32_bit_register_offset,	/* 0x0DE0A000	LD4	Rm != 11111 */
1180	//	LD4	//vld4single_structure_64_bit_register_offset,	/* 0x0DE0A400	LD4	Rm != 11111 */
1181	//	LD2R	//vld2rRegister_offset,	/* 0x0DE0C000	LD2R	Rm != 11111 */
1182	//	LD4R	//vld4rRegister_offset,	/* 0x0DE0E000	LD4R	Rm != 11111 */
1183	//	LD2	//vld2single_structure_8_bit_immediate_offset,	/* 0x0DFF0000	LD2	*/
1184	//	LD4	//vld4single_structure_8_bit_immediate_offset,	/* 0x0DFF2000	LD4	*/
1185	//	LD2	//vld2single_structure_16_bit_immediate_offset,	/* 0x0DFF4000	LD2	*/
1186	//	LD4	//vld4single_structure_16_bit_immediate_offset,	/* 0x0DFF6000	LD4	*/
1187	//	LD2	//vld2single_structure_32_bit_immediate_offset,	/* 0x0DFF8000	LD2	*/
1188	//	LD2	//vld2single_structure_64_bit_immediate_offset,	/* 0x0DFF8400	LD2	*/
1189	//	LD4	//vld4single_structure_32_bit_immediate_offset,	/* 0x0DFFA000	LD4	*/
1190	//	LD4	//vld4single_structure_64_bit_immediate_offset,	/* 0x0DFFA400	LD4	*/
1191	//	LD2R	//vld2rImmediate_offset,	/* 0x0DFFC000	LD2R	*/
1192	//	LD4R	//vld4rImmediate_offset,	/* 0x0DFFE000	LD4R	*/

1	**in_use**	**Opcode**	**//BINARY Opcode Opcodecomments**
2		**UNALLOCATED**	/* UNALLOCATED */
3		BAD	0x00000000,/* BAD badinvalid operation */
4		**Branch,exception generatic**	/* Branch,exception generation and system Instruction */
5		**Compare _ Branch (immediat**	/* Compare _ Branch (immediate) */
6		CBZ	0x34000000,/* CBZ cbzw */
7		CBNZ	0x35000000,/* CBNZ cbnzw */
8		CBZ	0xB4000000,/* CBZ cbzx */
9		CBNZ	0xB5000000,/* CBNZ cbnzx */
10		**Test bit & branch (immediate)**	/* Test bit & branch (immediate) */
11		TBZ	0x36000000,/* TBZ tbz */
12		TBNZ	0x37000000,/* TBNZ tbnz */
13		**Conditional branch (immediat**	/* Conditional branch (immediate) */
14		B_cond	0x54000000,/* B_cond b_cond */
15		**Exception generation**	/* Exception generation */
16	//	SVC	//0xD4000001,/* SVC svc */
17	//	HVC	//0xD4000002,/* HVC hvc */
18	//	SMC	//0xD4000003,/* SMC smc */
19		BRK	0xD4200000,/* BRK brkarm64AArch64 Specific BRK */
20	//	HLT	//0xD4400000,/* HLT hlt */
21	//	DCPS1	//0xD4A00001,/* DCPS1 dcps1 */
22	//	DCPS2	//0xD4A00002,/* DCPS2 dcps2 */
23	//	DCPS3	//0xD4A00003,/* DCPS3 dcps3 */
24	//	**System**	/* System */
25	//	MSR	//0xD500401F,/* MSR msrimm */
26	//	HINT	//0xD503201F,/* HINT hint */
27	//	CLREX	//0xD503305F,/* CLREX clrex */
28	//	DSB	//0xD503309F,/* DSB dsb */
29	//	DMB	//0xD50330BF,/* DMB dmb */
30	//	ISB	//0xD50330DF,/* ISB isb */
31	//	SYS	//0xD5080000,/* SYS sys */
32	//	MSR	//0xD5100000,/* MSR msr */
33	//	SYSL	//0xD5280000,/* SYSL sysl */
34	//	MRS	//0xD5300000,/* MRS mrs */
35		**Unconditional branch (registe**	/* Unconditional branch (register) */
36		BR	0xD61F0000,/* BR br */
37		BLR	0xD63F0000,/* BLR blr */

	in_use	Opcode	//BINARY Opcode Opcodecomments
38		RET	0xD65F0000,/* RET ret */
39	//	ERET	//0xD69F03E0,/* ERET eret */
40	//	DRPS	//0xD6BF03E0,/* DRPS drps */
41	//	**Unconditional branch (immed**	/* Unconditional branch (immediate) */
42		B	0x14000000,/* B b */
43		BL	0x94000000,/* BL bl */
44		**Loads and stores**	/* Loads and stores */
45		**Load/store exclusive**	/* Load/store exclusive */
46		STXRB	0x08000000,/* STXRB stxrb */
47		STLXRB	0x08008000,/* STLXRB stlxrb */
48		LDXRB	0x08400000,/* LDXRB ldxrb */
49		LDAXRB	0x08408000,/* LDAXRB ldaxrb */
50		STLRB	0x08808000,/* STLRB stlrb */
51		LDARB	0x08C08000,/* LDARB ldarb */
52		STXRH	0x48000000,/* STXRH stxrh */
53		STLXRH	0x48008000,/* STLXRH stlxrh */
54		LDXRH	0x48400000,/* LDXRH ldxrh */
55		LDAXRH	0x48408000,/* LDAXRH ldaxrh */
56		STLRH	0x48808000,/* STLRH stlrh */
57		LDARH	0x48C08000,/* LDARH ldarh */
58		STXR	0x88000000,/* STXR stxrw */
59		STLXR	0x88008000,/* STLXR stlxrw */
60		STXP	0x88200000,/* STXP stxpw */
61		STLXP	0x88208000,/* STLXP stlxpw */
62		LDXR	0x88400000,/* LDXR ldxrw */
63		LDAXR	0x88408000,/* LDAXR ldaxrw */
64		LDXP	0x88600000,/* LDXP ldxpw */
65		LDAXP	0x88608000,/* LDAXP ldaxpw */
66		STLR	0x88808000,/* STLR stlrw */
67		LDAR	0x88C08000,/* LDAR ldarw */
68		STXR	0xC8000000,/* STXR stxrx */
69		STLXR	0xC8008000,/* STLXR stlxrx */
70		STXP	0xC8200000,/* STXP stxpx */
71		STLXP	0xC8208000,/* STLXP stlxpx */
72		LDXR	0xC8400000,/* LDXR ldxrx */
73		LDAXR	0xC8408000,/* LDAXR ldaxrx */
74		LDXP	0xC8600000,/* LDXP ldxpx */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
75		LDAXP	0xC8608000,/* LDAXP ldaxpx */
76		STLR	0xC8808000,/* STLR stlrx */
77		LDAR	0xC8C08000,/* LDAR ldarx */
78		**Load register (literal)**	/* Load register (literal) */
79		LDR	0x18000000,/* LDR ldrw */
80		LDR	0x1C000000,/* LDR vldrs */
81		LDR	0x58000000,/* LDR ldrx */
82		LDR	0x5C000000,/* LDR vldrd */
83		LDRSW	0x98000000,/* LDRSW ldrsw */
84		LDR	0x9C000000,/* LDR vldrq */
85		PRFM	0xD8000000,/* PRFM prfm */
86		**Load/store no-allocate pair (o**	/* Load/store no-allocate pair (offset) */
87		STNP	0x28000000,/* STNP stnpw */
88		LDNP	0x28400000,/* LDNP ldnpw */
89		STNP	0x2C000000,/* STNP vstnps */
90		LDNP	0x2C400000,/* LDNP vldnps */
91		STNP	0x6C000000,/* STNP vstnpd */
92		LDNP	0x6C400000,/* LDNP vldnpd */
93		STNP	0xA8000000,/* STNP stnpx */
94		LDNP	0xA8400000,/* LDNP ldnpx */
95		STNP	0xAC000000,/* STNP vstnpq */
96		LDNP	0xAC400000,/* LDNP vldnpq */
97		**Load/store register pair (post**	/* Load/store register pair (post-indexed) */
98		STP	0x28800000,/* STP stppostw */
99		LDP	0x28C00000,/* LDP ldppostw */
100		STP	0x2C800000,/* STP vstpposts */
101		LDP	0x2CC00000,/* LDP vldpposts */
102		LDPSW	0x68C00000,/* LDPSW ldpswpost */
103		STP	0x6C800000,/* STP vstppostd */
104		LDP	0x6CC00000,/* LDP vldppostd */
105		STP	0xA8800000,/* STP stppostx */
106		LDP	0xA8C00000,/* LDP ldppostx */
107		STP	0xAC800000,/* STP vstppostq */
108		LDP	0xACC00000,/* LDP vldppostq */
109		**Load/store register pair (offse**	/* Load/store register pair (offset) */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
110		STP	0x29000000,/* STP stpoffw */
111		LDP	0x29400000,/* LDP ldpoffw */
112		STP	0x2D000000,/* STP vstpoffs */
113		LDP	0x2D400000,/* LDP vldpoffs */
114		LDPSW	0x69400000,/* LDPSW ldpswoff */
115		STP	0x6D000000,/* STP vstpoffd */
116		LDP	0x6D400000,/* LDP vldpoffd */
117		STP	0xA9000000,/* STP stpoffx */
118		LDP	0xA9400000,/* LDP ldpoffx */
119		STP	0xAD000000,/* STP vstpoffq */
120		LDP	0xAD400000,/* LDP vldpoffq */
121		**Load/store register pair (pre-i**	/* Load/store register pair (pre-indexed) */
122		STP	0x29800000,/* STP stpprew */
123		LDP	0x29C00000,/* LDP ldpprew */
124		STP	0x2D800000,/* STP vstppres */
125		LDP	0x2DC00000,/* LDP vldppres */
126		LDPSW	0x69C00000,/* LDPSW ldpswpre */
127		STP	0x6D800000,/* STP vstppred */
128		LDP	0x6DC00000,/* LDP vldppred */
129		STP	0xA9800000,/* STP stpprex */
130		LDP	0xA9C00000,/* LDP ldpprex */
131		STP	0xAD800000,/* STP vstppreq */
132		LDP	0xADC00000,/* LDP vldppreq */
133		**Load/store register (unscaled**	/* Load/store register (unscaled immediate) */
134		STURB	0x38000000,/* STURB sturb */
135		LDURB	0x38400000,/* LDURB ldurb */
136		LDURSB	0x38800000,/* LDURSB ldursbx */
137		LDURSB	0x38C00000,/* LDURSB ldursbw */
138		STUR	0x3C000000,/* STUR vsturb */
139		LDUR	0x3C400000,/* LDUR vldurb */
140		STUR	0x3C800000,/* STUR vsturq */
141		LDUR	0x3CC00000,/* LDUR vldurq */
142		STURH	0x78000000,/* STURH sturh */
143		LDURH	0x78400000,/* LDURH ldurh */
144		LDURSH	0x78800000,/* LDURSH ldurshx */
145		LDURSH	0x78C00000,/* LDURSH ldurshw */
146		STUR	0x7C000000,/* STUR vsturh */
147		LDUR	0x7C400000,/* LDUR vldurh */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
148		STUR	0xB8000000,/* STUR	sturw */	
149		LDUR	0xB8400000,/* LDUR	ldurw */	
150		LDURSW	0xB8800000,/* LDURSW	ldursw */	
151		STUR	0xBC000000,/* STUR	vsturs */	
152		LDUR	0xBC400000,/* LDUR	vldurs */	
153		STUR	0xF8000000,/* STUR	sturx */	
154		LDUR	0xF8400000,/* LDUR	ldurx */	
155		PRFUM	0xF8800000,/* PRFUM	prfum */	
156		STUR	0xFC000000,/* STUR	vsturd */	
157		LDUR	0xFC400000,/* LDUR	vldurd */	
158		**Load/store register (immediat**	/* Load/store register (immediate post-indexed) */		
159		STRB	0x38000400,/* STRB	strbpost */	
160		LDRB	0x38400400,/* LDRB	ldrbpost */	
161		LDRSB	0x38800400,/* LDRSB	ldrsbpostx */	
162		LDRSB	0x38C00400,/* LDRSB	ldrsbpostw */	
163		STR	0x3C000400,/* STR	vstrpostb */	
164		LDR	0x3C400400,/* LDR	vldrpostb */	
165		STR	0x3C800400,/* STR	vstrpostq */	
166		LDR	0x3CC00400,/* LDR	vldrpostq */	
167		STRH	0x78000400,/* STRH	strhpost */	
168		LDRH	0x78400400,/* LDRH	ldrhpost */	
169		LDRSH	0x78800400,/* LDRSH	ldrshpostx */	
170		LDRSH	0x78C00400,/* LDRSH	ldrshpostw */	
171		STR	0x7C000400,/* STR	vstrposth */	
172		LDR	0x7C400400,/* LDR	vldrposth */	
173		STR	0xB8000400,/* STR	strpostw */	
174		LDR	0xB8400400,/* LDR	ldrpostw */	
175		LDRSW	0xB8800400,/* LDRSW	ldrswpost */	
176		STR	0xBC000400,/* STR	vstrposts */	
177		LDR	0xBC400400,/* LDR	vldrposts */	
178		STR	0xF8000400,/* STR	strpostx */	
179		LDR	0xF8400400,/* LDR	ldrpostx */	
180		STR	0xFC000400,/* STR	vstrpostd */	
181		LDR	0xFC400400,/* LDR	vldrpostd */	
182		**Load/store register (unprivile**	/* Load/store register (unprivileged) */		
183		STTRB	0x38000800,/* STTRB	sttrb */	
184		LDTRB	0x38400800,/* LDTRB	ldtrb */	
185		LDTRSB	0x38800800,/* LDTRSB	ldtrsbx */	

1	in_use	Opcode	//BINARY Opcode Opcodecomments
186		LDTRSB	0x38C00800,/* LDTRSB ldtrsbw */
187		STTRH	0x78000800,/* STTRH sttrh */
188		LDTRH	0x78400800,/* LDTRH ldtrh */
189		LDTRSH	0x78800800,/* LDTRSH ldtrshx */
190		LDTRSH	0x78C00800,/* LDTRSH ldtrshw */
191		STTR	0xB8000800,/* STTR sttrw */
192		LDTR	0xB8400800,/* LDTR ldtrw */
193		LDTRSW	0xB8800800,/* LDTRSW ldtrsw */
194		STTR	0xF8000800,/* STTR sttrx */
195		LDTR	0xF8400800,/* LDTR ldtrx */
196		**Load/store register (immediat**	/* Load/store register (immediate pre-indexed) */
197		STRB	0x38000C00,/* STRB strbpre */
198		LDRB	0x38400C00,/* LDRB ldrbpre */
199		LDRSB	0x38800C00,/* LDRSB ldrsbprex */
200		LDRSB	0x38C00C00,/* LDRSB ldrsbprew */
201		STR	0x3C000C00,/* STR vstrpreb */
202		LDR	0x3C400C00,/* LDR vldrpreb */
203		STR	0x3C800C00,/* STR vstrpreq */
204		LDR	0x3CC00C00,/* LDR vldrpreq */
205		STRH	0x78000C00,/* STRH strhpre */
206		LDRH	0x78400C00,/* LDRH ldrhpre */
207		LDRSH	0x78800C00,/* LDRSH ldrshprex */
208		LDRSH	0x78C00C00,/* LDRSH ldrshprew */
209		STR	0x7C000C00,/* STR vstrpreh */
210		LDR	0x7C400C00,/* LDR vldrpreh */
211		STR	0xB8000C00,/* STR strprew */
212		LDR	0xB8400C00,/* LDR ldrprew */
213		LDRSW	0xB8800C00,/* LDRSW ldrswpre */
214		STR	0xBC000C00,/* STR vstrpres */
215		LDR	0xBC400C00,/* LDR vldrpres */
216		STR	0xF8000C00,/* STR strprex */
217		LDR	0xF8400C00,/* LDR ldrprex */
218		STR	0xFC000C00,/* STR vstrpred */
219		LDR	0xFC400C00,/* LDR vldrpred */
220		**Load/store register (register c**	/* Load/store register (register offset) */
221		STRB	0x38200800,/* STRB strboff */
222		LDRB	0x38600800,/* LDRB ldrboff */
223		LDRSB	0x38A00800,/* LDRSB ldrsboffx */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
224		LDRSB	0x38E00800,/* LDRSB	ldrsboffw */	
225		STR	0x3C200800,/* STR	vstroffb */	
226		LDR	0x3C600800,/* LDR	vldroffb */	
227		STR	0x3CA00800,/* STR	vstroffq */	
228		LDR	0x3CE00800,/* LDR	vldroffq */	
229		STRH	0x78200800,/* STRH	strhoff */	
230		LDRH	0x78600800,/* LDRH	ldrhoff */	
231		LDRSH	0x78A00800,/* LDRSH	ldrshoffx */	
232		LDRSH	0x78E00800,/* LDRSH	ldrshoffw */	
233		STR	0x7C200800,/* STR	vstroffh */	
234		LDR	0x7C600800,/* LDR	vldroffh */	
235		STR	0xB8200800,/* STR	stroffw */	
236		LDR	0xB8600800,/* LDR	ldroffw */	
237		LDRSW	0xB8A00800,/* LDRSW	ldrswoff */	
238		STR	0xBC200800,/* STR	vstroffs */	
239		LDR	0xBC600800,/* LDR	vldroffs */	
240		STR	0xF8200800,/* STR	stroffx */	
241		LDR	0xF8600800,/* LDR	ldroffx */	
243		STR	0xFC200800,/* STR	vstroffd */	
244		LDR	0xFC600800,/* LDR	vldroffd */	
242		PRFM	0xF8A00800,/* PRFM	prfmoff */	
245		**Load/store register (unsigned**	/* Load/store register (unsigned immediate) */		
246		STRB	0x39000000,/* STRB	strbimm */	
247		LDRB	0x39400000,/* LDRB	ldrbimm */	
248		LDRSB	0x39800000,/* LDRSB	ldrsbimmx */	
249		LDRSB	0x39C00000,/* LDRSB	ldrsbimmw */	
250		STR	0x3D000000,/* STR	vstrimmb */	
251		LDR	0x3D400000,/* LDR	vldrimmb */	
252		STR	0x3D800000,/* STR	vstrimmq */	
253		LDR	0x3DC00000,/* LDR	vldrimmq */	
254		STRH	0x79000000,/* STRH	strhimm */	
255		LDRH	0x79400000,/* LDRH	ldrhimm */	
256		LDRSH	0x79800000,/* LDRSH	ldrshimmx */	
257		LDRSH	0x79C00000,/* LDRSH	ldrshimmw */	
258		STR	0x7D000000,/* STR	vstrimmh */	
259		LDR	0x7D400000,/* LDR	vldrimmh */	
260		STR	0xB9000000,/* STR	strimmw */	
261		LDR	0xB9400000,/* LDR	ldrimmw */	

1	in_use	Opcode	//BINARY Opcode Opcodecomments
262		LDRSW	0xB9800000,/* LDRSW ldrswimm */
263		STR	0xBD000000,/* STR vstrimms */
264		LDR	0xBD400000,/* LDR vldrimms */
265		STR	0xF9000000,/* STR strimmx */
266		LDR	0xF9400000,/* LDR ldrimmx */
268		STR	0xFD000000,/* STR vstrimmd */
269		LDR	0xFD400000,/* LDR vldrimmd */
267		PRFM	0xF9800000,/* PRFM prfmimm */
270		**Data processing – Immedia**	/* Data processing – Immediate */
271		**PC-rel. addressing**	/* PC-rel. addressing */
272		ADR	0x10000000,/* ADR adr */
273		ADRP	0x90000000,/* ADRP adrp */
274		**Add/subtract (immediate)**	/* Add/subtract (immediate) */
275		ADD	0x11000000,/* ADD addimmw */
276		ADDS	0x31000000,/* ADDS addsimmw */
277		SUB	0x51000000,/* SUB subimmw */
278		SUBS	0x71000000,/* SUBS subsimmw */
279		ADD	0x91000000,/* ADD addimmx */
280		ADDS	0xB1000000,/* ADDS addsimmx */
281		SUB	0xD1000000,/* SUB subimmx */
282		SUBS	0xF1000000,/* SUBS subsimmx */
283		**Logical (immediate)**	/* Logical (immediate) */
284		AND	0x12000000,/* AND andimmw */
285		ORR	0x32000000,/* ORR orrimmw */
286		EOR	0x52000000,/* EOR eorimmw */
287		ANDS	0x72000000,/* ANDS andsimmw */
288		AND	0x92000000,/* AND andimmx */
289		ORR	0xB2000000,/* ORR orrimmx */
290		EOR	0xD2000000,/* EOR eorimmx */
291		ANDS	0xF2000000,/* ANDS andsimmx */
292		**Move wide (immediate)**	/* Move wide (immediate) */
293		MOVN	0x12800000,/* MOVN movnw */
294		MOVZ	0x52800000,/* MOVZ movzw */
295		MOVK	0x72800000,/* MOVK movkw */
296		MOVN	0x92800000,/* MOVN movnx */
297		MOVZ	0xD2800000,/* MOVZ movzx */
298		MOVK	0xF2800000,/* MOVK movkx */
299		**Bitfield**	/* Bitfield */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
300		SBFM	0x13000000,/* SBFM sbfmw */
301		BFM	0x33000000,/* BFM bfmw */
302		UBFM	0x53000000,/* UBFM ubfmw */
303		SBFM	0x93400000,/* SBFM sbfmx */
304		BFM	0xB3400000,/* BFM bfmx */
305		UBFM	0xD3400000,/* UBFM ubfmx */
306		**Extract**	/* Extract */
307		EXTR	0x13800000,/* EXTR extrw */
308		EXTR	0x93C08000,/* EXTR extrx */
309		**Data Processing – register**	/* Data Processing – register */
310		**Logical (shifted register)**	/* Logical (shifted register) */
311		AND	0x0A000000,/* AND andw */
312		BIC	0x0A200000,/* BIC bicw */
313		ORR	0x2A000000,/* ORR orrw */
314		ORN	0x2A200000,/* ORN ornw */
315		EOR	0x4A000000,/* EOR eorw */
316		EON	0x4A200000,/* EON eonw */
317		ANDS	0x6A000000,/* ANDS andsw */
318		BICS	0x6A200000,/* BICS bicsw */
319		AND	0x8A000000,/* AND andx */
320		BIC	0x8A200000,/* BIC bicx */
321		ORR	0xAA000000,/* ORR orrx */
322		ORN	0xAA200000,/* ORN ornx */
323		EOR	0xCA000000,/* EOR eorx */
324		EON	0xCA200000,/* EON eonx */
325		ANDS	0xEA000000,/* ANDS andsx */
326		BICS	0xEA200000,/* BICS bicsx */
327		**Add/subtract (shifted register**	/* Add/subtract (shifted register) */
328		ADD	0x0B000000,/* ADD addw */
329		ADDS	0x2B000000,/* ADDS addsw */
330		SUB	0x4B000000,/* SUB subw */
331		SUBS	0x6B000000,/* SUBS subsw */
332		ADD	0x8B000000,/* ADD addx */
333		ADDS	0xAB000000,/* ADDS addsx */
334		SUB	0xCB000000,/* SUB subx */
335		SUBS	0xEB000000,/* SUBS subsx */
336		**Add/subtract (extended regist**	/* Add/subtract (extended register) */
337		ADD	0x0B200000,/* ADD addextw */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
338		ADDS	0x2B200000,/* ADDS addsextw */
339		SUB	0x4B200000,/* SUB subextw */
340		SUBS	0x6B200000,/* SUBS subsextw */
341		ADD	0x8B200000,/* ADD addextx */
342		ADDS	0xAB200000,/* ADDS addsextx */
343		SUB	0xCB200000,/* SUB subextx */
344		SUBS	0xEB200000,/* SUBS subsextx */
345		**Add/subtract (with carry)**	/* Add/subtract (with carry) */
346		ADC	0x1A000000,/* ADC adcw */
347		ADCS	0x3A000000,/* ADCS adcsw */
348		SBC	0x5A000000,/* SBC sbcw */
349		SBCS	0x7A000000,/* SBCS sbcsw */
350		ADC	0x9A000000,/* ADC adcx */
351		ADCS	0xBA000000,/* ADCS adcsx */
352		SBC	0xDA000000,/* SBC sbcx */
353		SBCS	0xFA000000,/* SBCS sbcsx */
354		**Conditional compare (registe**	/* Conditional compare (register) */
355		CCMN	0x3A400000,/* CCMN ccmnw */
356		CCMN	0xBA400000,/* CCMN ccmnx */
357		CCMP	0x7A400000,/* CCMP ccmpw */
358		CCMP	0xFA400000,/* CCMP ccmpx */
359		**Conditional compare (immedi**	/* Conditional compare (immediate) */
360		CCMN	0x3A400800,/* CCMN ccmnimmw */
361		CCMN	0xBA400800,/* CCMN ccmnimmx */
362		CCMP	0x7A400800,/* CCMP ccmpimmw */
363		CCMP	0xFA400800,/* CCMP ccmpimmx */
364		**Conditional select**	/* Conditional select */
365		CSEL	0x1A800000,/* CSEL cselw */
366		CSINC	0x1A800400,/* CSINC csincw */
367		CSINV	0x5A800000,/* CSINV csinvw */
368		CSNEG	0x5A800400,/* CSNEG csnegw */
369		CSEL	0x9A800000,/* CSEL cselx */
370		CSINC	0x9A800400,/* CSINC csincx */
371		CSINV	0xDA800000,/* CSINV csinvx */
372		CSNEG	0xDA800400,/* CSNEG csnegx */
373		**Data-processing (3 source)**	/* Data-processing (3 source) */
374		MADD	0x1B000000,/* MADD maddw */
375		MADD	0x9B000000,/* MADD maddx */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
376		SMADDL	0x9B200000,/* SMADDL smaddl */
377		UMADDL	0x9BA00000,/* UMADDL umaddl */
378		MSUB	0x1B008000,/* MSUB msubw */
379		MSUB	0x9B008000,/* MSUB msubx */
380		SMSUBL	0x9B208000,/* SMSUBL smsubl */
381		UMSUBL	0x9BA08000,/* UMSUBL umsubl */
382		SMULH	0x9B400000,/* SMULH smulh */
383		UMULH	0x9BC00000,/* UMULH umulh */
384		**Data-processing (2 source)**	/* Data-processing (2 source) */
385		CRC32X	0x9AC04C00,/* CRC32X crc32x */
386		CRC32CX	0x9AC05C00,/* CRC32CX crc32cx */
387		CRC32B	0x1AC04000,/* CRC32B crc32b */
388		CRC32CB	0x1AC05000,/* CRC32CB crc32cb */
389		CRC32H	0x1AC04400,/* CRC32H crc32h */
390		CRC32CH	0x1AC05400,/* CRC32CH crc32ch */
391		CRC32W	0x1AC04800,/* CRC32W crc32w */
392		CRC32CW	0x1AC05800,/* CRC32CW crc32cw */
393		UDIV	0x1AC00800,/* UDIV udivw */
394		UDIV	0x9AC00800,/* UDIV udivx */
395		SDIV	0x1AC00C00,/* SDIV sdivw */
396		SDIV	0x9AC00C00,/* SDIV sdivx */
397		LSLV	0x1AC02000,/* LSLV lslvw */
398		LSLV	0x9AC02000,/* LSLV lslvx */
399		LSRV	0x1AC02400,/* LSRV lsrvw */
400		LSRV	0x9AC02400,/* LSRV lsrvx */
401		ASRV	0x1AC02800,/* ASRV asrvw */
402		ASRV	0x9AC02800,/* ASRV asrvx */
403		RORV	0x1AC02C00,/* RORV rorvw */
404		RORV	0x9AC02C00,/* RORV rorvx */
405		**Data-processing (1 source)**	/* Data-processing (1 source) */
406		RBIT	0x5AC00000,/* RBIT rbitw */
407		RBIT	0xDAC00000,/* RBIT rbitx */
408		CLZ	0x5AC01000,/* CLZ clzw */
409		CLZ	0xDAC01000,/* CLZ clzx */
410		CLS	0x5AC01400,/* CLS clsw */
411		CLS	0xDAC01400,/* CLS clsx */
412		REV	0x5AC00800,/* REV revw */
413		REV	0xDAC00C00,/* REV revx */

1	**in_use**	**Opcode**	**//BINARY**	**Opcode**	**Opcodecomments**
414		REV16	0xDAC00400,/* REV16	rev16w */	
415		REV16	0x5AC00400,/* REV16	rev16x */	
416		REV32	0xDAC00800,/* REV32	rev32 */	
417	//	**Data Processing – SIMD an**	/* Data Processing – SIMD and floating point */		
418	//	**Floating-point<->fixed-point c**	/* Floating-point<->fixed-point conversions */		
419	//	SCVTF	//0x1E020000,/* SCVTF	vscvtfscalar_fixed_point_32_bit_to_single_precision */	
420	//	UCVTF	//0x1E030000,/* UCVTF	vucvtfscalar_fixed_point_32_bit_to_single_precision */	
421	//	FCVTZS	//0x1ED80000,/* FCVTZS	vfcvtzsscalar_fixed_point_Single_precision_to_32_bi	
422	//	FCVTZU	//0x1ED90000,/* FCVTZU	vfcvtzuscalar_fixed_point_Single_precision_to_32_bi	
423	//	SCVTF	//0x1E020000,/* SCVTF	vscvtfscalar_fixed_point_32_bit_to_double_precision *	
424	//	UCVTF	//0x1E030000,/* UCVTF	vucvtfscalar_fixed_point_32_bit_to_double_precision *	
425	//	FCVTZS	//0x1ED80000,/* FCVTZS	vfcvtzsscalar_fixed_point_Double_precision_to_32_b	
426	//	FCVTZU	//0x1ED90000,/* FCVTZU	vfcvtzuscalar_fixed_point_Double_precision_to_32_b	
427	//	SCVTF	//0x9E020000,/* SCVTF	vscvtfscalar_fixed_point_64_bit_to_single_precision */	
428	//	UCVTF	//0x9E030000,/* UCVTF	vucvtfscalar_fixed_point_64_bit_to_single_precision */	
429	//	FCVTZS	//0x9ED80000,/* FCVTZS	vfcvtzsscalar_fixed_point_Single_precision_to_64_bi	
430	//	FCVTZU	//0x9ED90000,/* FCVTZU	vfcvtzuscalar_fixed_point_Single_precision_to_64_bi	
431	//	SCVTF	//0x9E020000,/* SCVTF	vscvtfscalar_fixed_point_64_bit_to_double_precision *	
432	//	UCVTF	//0x9E030000,/* UCVTF	vucvtfscalar_fixed_point_64_bit_to_double_precision *	
433	//	FCVTZS	//0x9ED80000,/* FCVTZS	vfcvtzsscalar_fixed_point_Double_precision_to_64_b	
434	//	FCVTZU	//0x9ED90000,/* FCVTZU	vfcvtzuscalar_fixed_point_Double_precision_to_64_b	
435	//	**Floating-point conditional cor**	/* Floating-point conditional compare */		
436	//	FCCMP	//0x1E200400,/* FCCMP	vfccmpSingle_precision */	
437	//	FCCMPE	//0x1E200410,/* FCCMPE	vfccmpeSingle_precision */	
438	//	FCCMP	//0x1E600400,/* FCCMP	vfccmpDouble_precision */	
439	//	FCCMPE	//0x1E600410,/* FCCMPE	vfccmpeDouble_precision */	
440	//	**Floating-point data-processin**	/* Floating-point data-processing (2 source) */		
441	//	FMUL	//0x1E200800,/* FMUL	vfmulscalar_Single_precision */	
442	//	FDIV	//0x1E201800,/* FDIV	vfdivscalar_Single_precision */	
443	//	FADD	//0x1E202800,/* FADD	vfaddscalar_Single_precision */	
444	//	FSUB	//0x1E203800,/* FSUB	vfsubscalar_Single_precision */	
445	//	FMAX	//0x1E204800,/* FMAX	vfmaxscalar_Single_precision */	
446	//	FMIN	//0x1E205800,/* FMIN	vfminscalar_Single_precision */	
447	//	FMAXNM	//0x1E206800,/* FMAXNM	vfmaxnmscalar_Single_precision */	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
448	//	FMINNM	//0x1E207800,/*	FMINNM	vfminnmscalar_Single_precision */
449	//	FNMUL	//0x1E208800,/*	FNMUL	vfnmulSingle_precision */
450	//	FMUL	//0x1E600800,/*	FMUL	vfmulscalar_Double_precision */
451	//	FDIV	//0x1E601800,/*	FDIV	vfdivscalar_Double_precision */
452	//	FADD	//0x1E602800,/*	FADD	vfaddscalar_Double_precision */
453	//	FSUB	//0x1E603800,/*	FSUB	vfsubscalar_Double_precision */
454	//	FMAX	//0x1E604800,/*	FMAX	vfmaxscalar_Double_precision */
455	//	FMIN	//0x1E605800,/*	FMIN	vfminscalar_Double_precision */
456	//	FMAXNM	//0x1E606800,/*	FMAXNM	vfmaxnmscalar_Double_precision */
457	//	FMINNM	//0x1E607800,/*	FMINNM	vfminnmscalar_Double_precision */
458	//	FNMUL	//0x1E608800,/*	FNMUL	vfnmulDouble_precision */
459	//	**Floating-point conditional sel**	/* Floating-point conditional select */		
460	//	FCSEL	//0x1E200C00,/*	FCSEL	vfcselSingle_precision */
461	//	FCSEL	//0x1E600C00,/*	FCSEL	vfcselDouble_precision */
462	//	**Floating-point immediate**	/* Floating-point immediate */		
463	//	FMOV	//0x1E201000,/*	FMOV	vfmovscalar_immediate_Single_precision */
464	//	FMOV	//0x1E601000,/*	FMOV	vfmovscalar_immediate_Double_precision */
465	//	**Floating-point compare**	/* Floating-point compare */		
466	//	FCMP	//0x1E202000,/*	FCMP	vfcmpSingle_precision */
467	//	FCMP	//0x1E202008,/*	FCMP	vfcmpSingle_precision_zero */
468	//	FCMPE	//0x1E202010,/*	FCMPE	vfcmpeSingle_precision */
469	//	FCMPE	//0x1E202018,/*	FCMPE	vfcmpeSingle_precision_zero */
470	//	FCMP	//0x1E602000,/*	FCMP	vfcmpDouble_precision */
471	//	FCMP	//0x1E602008,/*	FCMP	vfcmpDouble_precision_zero */
472	//	FCMPE	//0x1E602010,/*	FCMPE	vfcmpeDouble_precision */
473	//	FCMPE	//0x1E602018,/*	FCMPE	vfcmpeDouble_precision_zero */
474	//	**Floating-point data-processin**	/* Floating-point data-processing (1 source) */		
475	//	FMOV	//0x1E204000,/*	FMOV	vfmovregister_Single_precision */
476	//	FABS	//0x1E20C000,/*	FABS	vfabsscalar_Single_precision */
477	//	FNEG	//0x1E214000,/*	FNEG	vfnegscalar_Single_precision */
478	//	FSQRT	//0x1E21C000,/*	FSQRT	vfsqrtscalar_Single_precision */
479	//	FCVT	//0x1E22C000,/*	FCVT	vfcvtSingle_precision_to_double_precision */
480	//	FCVT	//0x1E23C000,/*	FCVT	vfcvtSingle_precision_to_half_precision */
481	//	FRINTN	//0x1E244000,/*	FRINTN	vfrintnscalar_Single_precision */

1	in_use	Opcode	//BINARY Opcode	Opcodecomments
482	//	FRINTP	//0x1E24C000,/* FRINTP	vfrintpscalar_Single_precision */
483	//	FRINTM	//0x1E254000,/* FRINTM	vfrintmscalar_Single_precision */
484	//	FRINTZ	//0x1E25C000,/* FRINTZ	vfrintzscalar_Single_precision */
485	//	FRINTA	//0x1E264000,/* FRINTA	vfrintascalar_Single_precision */
486	//	FRINTX	//0x1E274000,/* FRINTX	vfrintxscalar_Single_precision */
487	//	FRINTI	//0x1E27C000,/* FRINTI	vfrintiscalar_Single_precision */
488	//	FMOV	//0x1E604000,/* FMOV	vfmovregister_Double_precision */
489	//	FABS	//0x1E60C000,/* FABS	vfabsscalar_Double_precision */
490	//	FNEG	//0x1E614000,/* FNEG	vfnegscalar_Double_precision */
491	//	FSQRT	//0x1E61C000,/* FSQRT	vfsqrtscalar_Double_precision */
492	//	FCVT	//0x1E624000,/* FCVT	vfcvtDouble_precision_to_single_precision */
493	//	FCVT	//0x1E63C000,/* FCVT	vfcvtDouble_precision_to_half_precision */
494	//	FRINTN	//0x1E644000,/* FRINTN	vfrintnscalar_Double_precision */
495	//	FRINTP	//0x1E64C000,/* FRINTP	vfrintpscalar_Double_precision */
496	//	FRINTM	//0x1E654000,/* FRINTM	vfrintmscalar_Double_precision */
497	//	FRINTZ	//0x1E65C000,/* FRINTZ	vfrintzscalar_Double_precision */
498	//	FRINTA	//0x1E664000,/* FRINTA	vfrintascalar_Double_precision */
499	//	FRINTX	//0x1E674000,/* FRINTX	vfrintxscalar_Double_precision */
500	//	FRINTI	//0x1E67C000,/* FRINTI	vfrintiscalar_Double_precision */
501	//	FCVT	//0x1EE24000,/* FCVT	vfcvtHalf_precision_to_single_precision */
502	//	FCVT	//0x1EE2C000,/* FCVT	vfcvtHalf_precision_to_double_precision */
503	//	**Floating-point<->integer conv**	/* Floating-point<->integer conversions */	
504	//	FCVTNS	//0x1E200000,/* FCVTNS	vfcvtnsscalar_Single_precision_to_32_bit */
505	//	FCVTNU	//0x1E210000,/* FCVTNU	vfcvtnuscalar_Single_precision_to_32_bit */
506	//	SCVTF	//0x1E220000,/* SCVTF	vscvtfscalar_integer_32_bit_to_single_precision */
507	//	UCVTF	//0x1E230000,/* UCVTF	vucvtfscalar_integer_32_bit_to_single_precision */
508	//	FCVTAS	//0x1E240000,/* FCVTAS	vfcvtasscalar_Single_precision_to_32_bit */
509	//	FCVTAU	//0x1E250000,/* FCVTAU	vfcvtauscalar_Single_precision_to_32_bit */
510	//	FMOV	//0x1E260000,/* FMOV	vfmovgeneral_Single_precision_to_32_bit */
511	//	FMOV	//0x1E270000,/* FMOV	vfmovgeneral_32_bit_to_single_precision */
512	//	FCVTPS	//0x1E280000,/* FCVTPS	vfcvtpsscalar_Single_precision_to_32_bit */
513	//	FCVTPU	//0x1E290000,/* FCVTPU	vfcvtpuscalar_Single_precision_to_32_bit */
514	//	FCVTMS	//0x1E300000,/* FCVTMS	vfcvtmsscalar_Single_precision_to_32_bit */
515	//	FCVTMU	//0x1E310000,/* FCVTMU	vfcvtmuscalar_Single_precision_to_32_bit */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
516	//	FCVTZS	//0x1E380000,/*	FCVTZS	vfcvtzsscalar_integer_Single_precision_to_32_bit */
517	//	FCVTZU	//0x1E390000,/*	FCVTZU	vfcvtzuscalar_integer_Single_precision_to_32_bit */
518	//	FCVTNS	//0x1E600000,/*	FCVTNS	vfcvtnsscalar_Double_precision_to_32_bit */
519	//	FCVTNU	//0x1E610000,/*	FCVTNU	vfcvtnuscalar_Double_precision_to_32_bit */
520	//	SCVTF	//0x1E620000,/*	SCVTF	vscvtfscalar_integer_32_bit_to_double_precision */
521	//	UCVTF	//0x1E630000,/*	UCVTF	vucvtfscalar_integer_32_bit_to_double_precision */
522	//	FCVTAS	//0x1E640000,/*	FCVTAS	vfcvtasscalar_Double_precision_to_32_bit */
523	//	FCVTAU	//0x1E650000,/*	FCVTAU	vfcvtauscalar_Double_precision_to_32_bit */
524	//	FCVTPS	//0x1E680000,/*	FCVTPS	vfcvtpsscalar_Double_precision_to_32_bit */
525	//	FCVTPU	//0x1E690000,/*	FCVTPU	vfcvtpuscalar_Double_precision_to_32_bit */
526	//	FCVTMS	//0x1E700000,/*	FCVTMS	vfcvtmsscalar_Double_precision_to_32_bit */
527	//	FCVTMU	//0x1E710000,/*	FCVTMU	vfcvtmuscalar_Double_precision_to_32_bit */
528	//	FCVTZS	//0x1E780000,/*	FCVTZS	vfcvtzsscalar_integer_Double_precision_to_32_bit */
529	//	FCVTZU	//0x1E790000,/*	FCVTZU	vfcvtzuscalar_integer_Double_precision_to_32_bit */
530	//	FCVTNS	//0x9E200000,/*	FCVTNS	vfcvtnsscalar_Single_precision_to_64_bit */
531	//	FCVTNU	//0x9E210000,/*	FCVTNU	vfcvtnuscalar_Single_precision_to_64_bit */
532	//	SCVTF	//0x9E220000,/*	SCVTF	vscvtfscalar_integer_64_bit_to_single_precision */
533	//	UCVTF	//0x9E230000,/*	UCVTF	vucvtfscalar_integer_64_bit_to_single_precision */
534	//	FCVTAS	//0x9E240000,/*	FCVTAS	vfcvtasscalar_Single_precision_to_64_bit */
535	//	FCVTAU	//0x9E250000,/*	FCVTAU	vfcvtauscalar_Single_precision_to_64_bit */
536	//	FCVTPS	//0x9E280000,/*	FCVTPS	vfcvtpsscalar_Single_precision_to_64_bit */
537	//	FCVTPU	//0x9E290000,/*	FCVTPU	vfcvtpuscalar_Single_precision_to_64_bit */
538	//	FCVTMS	//0x9E300000,/*	FCVTMS	vfcvtmsscalar_Single_precision_to_64_bit */
539	//	FCVTMU	//0x9E310000,/*	FCVTMU	vfcvtmuscalar_Single_precision_to_64_bit */
540	//	FCVTZS	//0x9E380000,/*	FCVTZS	vfcvtzsscalar_integer_Single_precision_to_64_bit */
541	//	FCVTZU	//0x9E390000,/*	FCVTZU	vfcvtzuscalar_integer_Single_precision_to_64_bit */
542	//	FCVTNS	//0x9E600000,/*	FCVTNS	vfcvtnsscalar_Double_precision_to_64_bit */
543	//	FCVTNU	//0x9E610000,/*	FCVTNU	vfcvtnuscalar_Double_precision_to_64_bit */
544	//	SCVTF	//0x9E620000,/*	SCVTF	vscvtfscalar_integer_64_bit_to_double_precision */
545	//	UCVTF	//0x9E630000,/*	UCVTF	vucvtfscalar_integer_64_bit_to_double_precision */
546	//	FCVTAS	//0x9E640000,/*	FCVTAS	vfcvtasscalar_Double_precision_to_64_bit */
547	//	FCVTAU	//0x9E650000,/*	FCVTAU	vfcvtauscalar_Double_precision_to_64_bit */
548	//	FMOV	//0x9E660000,/*	FMOV	vfmovgeneral_Double_precision_to_64_bit */
549	//	FMOV	//0x9E670000,/*	FMOV	vfmovgeneral_64_bit_to_double_precision */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
550	//	FCVTPS	//0x9E680000,/*	FCVTPS	vfcvtpsscalar_Double_precision_to_64_bit */
551	//	FCVTPU	//0x9E690000,/*	FCVTPU	vfcvtpuscalar_Double_precision_to_64_bit */
552	//	FCVTMS	//0x9E700000,/*	FCVTMS	vfcvtmsscalar_Double_precision_to_64_bit */
553	//	FCVTMU	//0x9E710000,/*	FCVTMU	vfcvtmuscalar_Double_precision_to_64_bit */
554	//	FCVTZS	//0x9E780000,/*	FCVTZS	vfcvtzsscalar_integer_Double_precision_to_64_bit */
555	//	FCVTZU	//0x9E790000,/*	FCVTZU	vfcvtzuscalar_integer_Double_precision_to_64_bit */
556	//	FMOV	//0x9EAE0000,/*	FMOV	vfmovgeneral_Top_half_of_128_bit_to_64_bit */
557	//	FMOV	//0x9EAF0000,/*	FMOV	vfmovgeneral_64_bit_to_top_half_of_128_bit */
558	//	**Floating-point data-processin**	/* Floating-point data-processing (3 source) */		
559	//	FMADD	//0x1F000000,/*	FMADD	vfmaddSingle_precision */
560	//	FMSUB	//0x1F008000,/*	FMSUB	vfmsubSingle_precision */
561	//	FNMADD	//0x1F200000,/*	FNMADD	vfnmaddSingle_precision */
562	//	FNMSUB	//0x1F208000,/*	FNMSUB	vfnmsubSingle_precision */
563	//	FMADD	//0x1F400000,/*	FMADD	vfmaddDouble_precision */
564	//	FMSUB	//0x1F408000,/*	FMSUB	vfmsubDouble_precision */
565	//	FNMADD	//0x1F600000,/*	FNMADD	vfnmaddDouble_precision */
566	//	FNMSUB	//0x1F608000,/*	FNMSUB	vfnmsubDouble_precision */
567	//	**AdvSIMD scalar three same**	/* AdvSIMD scalar three same */		
568	//	SQADD	//0x5E200C00,/*	SQADD	vsqaddScalar */
569	//	SQSUB	//0x5E202C00,/*	SQSUB	vsqsubScalar */
570	//	CMGT	//0x5E203400,/*	CMGT	vcmgtregister_Scalar */
571	//	CMGE	//0x5E203C00,/*	CMGE	vcmgeregister_Scalar */
572	//	SSHL	//0x5E204400,/*	SSHL	vsshlScalar */
573	//	SQSHL	//0x5E204C00,/*	SQSHL	vsqshlregister_Scalar */
574	//	SRSHL	//0x5E205400,/*	SRSHL	vsrshlScalar */
575	//	SQRSHL	//0x5E205C00,/*	SQRSHL	vsqrshlScalar */
576	//	ADD	//0x5E208400,/*	ADD	vaddvector_Scalar */
577	//	CMTST	//0x5E208C00,/*	CMTST	vcmtstScalar */
578	//	SQDMULH	//0x5E20B400,/*	SQDMULH	vsqdmulhvector_Scalar */
579	//	FMULX	//0x5E20DC00,/*	FMULX	vfmulxScalar */
580	//	FCMEQ	//0x5E20E400,/*	FCMEQ	vfcmeqregister_Scalar */
581	//	FRECPS	//0x5E20FC00,/*	FRECPS	vfrecpsScalar */
582	//	FRSQRTS	//0x5EA0FC00,/*	FRSQRTS	vfrsqrtsScalar */
583	//	UQADD	//0x7E200C00,/*	UQADD	vuqaddScalar */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
584	//	UQSUB	//0x7E202C00,/*	UQSUB	vuqsubScalar */
585	//	CMHI	//0x7E203400,/*	CMHI	vcmhiregister_Scalar */
586	//	CMHS	//0x7E203C00,/*	CMHS	vcmhsregister_Scalar */
587	//	USHL	//0x7E204400,/*	USHL	vushlScalar */
588	//	UQSHL	//0x7E204C00,/*	UQSHL	vuqshlregister_Scalar */
589	//	URSHL	//0x7E205400,/*	URSHL	vurshlScalar */
590	//	UQRSHL	//0x7E205C00,/*	UQRSHL	vuqrshlScalar */
591	//	SUB	//0x7E208400,/*	SUB	vsubvector_Scalar */
592	//	CMEQ	//0x7E208C00,/*	CMEQ	vcmeqregister_Scalar */
593	//	SQRDMULH	//0x7E20B400,/*	SQRDMULH	vsqrdmulhvector_Scalar */
594	//	FCMGE	//0x7E20E400,/*	FCMGE	vfcmgeregister_Scalar */
595	//	FACGE	//0x7E20EC00,/*	FACGE	vfacgeScalar */
596	//	FABD	//0x7EA0D400,/*	FABD	vfabdScalar */
597	//	FCMGT	//0x7EA0E400,/*	FCMGT	vfcmgtregister_Scalar */
598	//	FACGT	//0x7EA0EC00,/*	FACGT	vfacgtScalar */
599	//	**AdvSIMD scalar three differer**	/* AdvSIMD scalar three different */		
600	//	SQDMLAL	//0x5E209000,/*	SQDMLAL	vsqdmlalvector_Scalarwrites to low half of the dest.
601	//	SQDMLAL2	//0x5E209000,/*	SQDMLAL2	vsqdmlal2vector_Scalarwrites to high half of the des
602	//	SQDMLSL	//0x5E20B000,/*	SQDMLSL	vsqdmlslvector_Scalarwrites to low half of the dest.
603	//	SQDMLSL2	//0x5E20B000,/*	SQDMLSL2	vsqdmlsl2vector_Scalarwrites to high half of the des
604	//	SQDMULL	//0x5E20D000,/*	SQDMULL	vsqdmullvector_Scalarwrites to low half of the dest.
605	//	SQDMULL2	//0x5E20D000,/*	SQDMULL2	vsqdmull2vector_Scalarwrites to high half of the de
606	//	**AdvSIMD scalar two-reg misc**	/* AdvSIMD scalar two-reg misc */		
607	//	SUQADD	//0x5E203800,/*	SUQADD	vsuqaddScalar */
608	//	SQABS	//0x5E207800,/*	SQABS	vsqabsScalar */
609	//	CMGT	//0x5E208800,/*	CMGT	vcmgtzero_Scalar */
610	//	CMEQ	//0x5E209800,/*	CMEQ	vcmeqzero_Scalar */
611	//	CMLT	//0x5E20A800,/*	CMLT	vcmltzero_Scalar */
612	//	ABS	//0x5E20B800,/*	ABS	vabsScalar */
613	//	SQXTN	//0x5E214800,/*	SQXTN	vsqxtnScalarwrites to low half of the dest. register & cl
614	//	SQXTN2	//0x5E214800,/*	SQXTN2	vsqxtn2Scalarwrites to high half of the dest. register &
615	//	FCVTNS	//0x5E21A800,/*	FCVTNS	vfcvtnsvector_Scalar */
616	//	FCVTMS	//0x5E21B800,/*	FCVTMS	vfcvtmsvector_Scalar */
617	//	FCVTAS	//0x5E21C800,/*	FCVTAS	vfcvtasvector_Scalar */

1	in_use	Opcode	//BINARY Opcode	Opcodecomments
618	//	SCVTF	//0x5E21D800,/* SCVTF	vscvtfvector_integer_Scalar */
619	//	FCMGT	//0x5EA0C800,/* FCMGT	vfcmgtzero_Scalar */
620	//	FCMEQ	//0x5EA0D800,/* FCMEQ	vfcmeqzero_Scalar */
621	//	FCMLT	//0x5EA0E800,/* FCMLT	vfcmltzero_Scalar */
622	//	FCVTPS	//0x5EA1A800,/* FCVTPS	vfcvtpsvector_Scalar */
623	//	FCVTZS	//0x5EA1B800,/* FCVTZS	vfcvtzsvector_integer_Scalar */
624	//	FRECPE	//0x5EA1D800,/* FRECPE	vfrecpeScalar */
625	//	FRECPX	//0x5EA1F800,/* FRECPX	frecpx */
626	//	USQADD	//0x7E203800,/* USQADD	vusqaddScalar */
627	//	SQNEG	//0x7E207800,/* SQNEG	vsqnegScalar */
628	//	CMGE	//0x7E208800,/* CMGE	vcmgezero_Scalar */
629	//	CMLE	//0x7E209800,/* CMLE	vcmlezero_Scalar */
630	//	NEG	//0x7E20B800,/* NEG	vnegvector_Scalar */
631	//	SQXTUN	//0x7E212800,/* SQXTUN	vsqxtunScalarwrites to low half of the dest. register &
632	//	SQXTUN2	//0x7E212800,/* SQXTUN2	vsqxtun2Scalarwrites to high half of the dest. registe
633	//	UQXTN	//0x7E214800,/* UQXTN	vuqxtnScalarwrites to low half of the dest. register & c
634	//	UQXTN2	//0x7E214800,/* UQXTN2	vuqxtn2Scalarwrites to high half of the dest. register &
635	//	FCVTXN	//0x7E216800,/* FCVTXN	vfcvtxnScalarwrites to low half of the dest. register &
636	//	FCVTXN2	//0x7E216800,/* FCVTXN2	vfcvtxn2Scalarwrites to high half of the dest. register
637	//	FCVTNU	//0x7E21A800,/* FCVTNU	vfcvtnuvector_Scalar */
638	//	FCVTMU	//0x7E21B800,/* FCVTMU	vfcvtmuvector_Scalar */
639	//	FCVTAU	//0x7E21C800,/* FCVTAU	vfcvtauvector_Scalar */
640	//	UCVTF	//0x7E21D800,/* UCVTF	vucvtfvector_integer_Scalar */
641	//	FCMGE	//0x7EA0C800,/* FCMGE	vfcmgezero_Scalar */
642	//	FCMLE	//0x7EA0D800,/* FCMLE	vfcmlezero_Scalar */
643	//	FCVTPU	//0x7EA1A800,/* FCVTPU	vfcvtpuvector_Scalar */
644	//	FCVTZU	//0x7EA1B800,/* FCVTZU	vfcvtzuvector_integer_Scalar */
645	//	FRSQRTE	//0x7EA1D800,/* FRSQRTE	vfrsqrteScalar */
646	//	**AdvSIMD scalar pairwise**	/* AdvSIMD scalar pairwise */	
647	//	ADDP	//0x5E31B800,/* ADDP	addpscalar */
648	//	FMAXNMP	//0x7E30C800,/* FMAXNMP	fmaxnmpscalar */
649	//	FADDP	//0x7E30D800,/* FADDP	faddpscalar */
650	//	FMAXP	//0x7E30F800,/* FMAXP	fmaxpscalar */
651	//	FMINNMP	//0x7EB0C800,/* FMINNMP	fminnmpscalar */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
652	//	FMINP	//0x7EB0F800,/* FMINP fminpscalar */
653	//	**AdvSIMD scalar copy**	/* AdvSIMD scalar copy */
654	//	DUP	//0x5E000400,/* DUP vdupelement_Scalar */
655	//	**AdvSIMD scalar x indexed ele**	/* AdvSIMD scalar x indexed element */
656	//	SQDMLAL	//0x5F003000,/* SQDMLAL vsqdmlalby_element_Scalar */
657	//	SQDMLAL2	//0x5F003000,/* SQDMLAL2 vsqdmlal2by_element_Scalar */
658	//	SQDMLSL	//0x5F007000,/* SQDMLSL vsqdmlslby_element_Scalar */
659	//	SQDMLSL2	//0x5F007000,/* SQDMLSL2 vsqdmlsl2by_element_Scalar */
660	//	SQDMULL	//0x5F00B000,/* SQDMULL vsqdmullby_element_Scalar */
661	//	SQDMULL2	//0x5F00B000,/* SQDMULL2 vsqdmull2by_element_Scalar */
662	//	SQDMULH	//0x5F00C000,/* SQDMULH vsqdmulhby_element_Scalar */
663	//	SQRDMULH	//0x5F00D000,/* SQRDMULH vsqrdmulhby_element_Scalar */
664	//	FMLA	//0x5F801000,/* FMLA vfmlaby_element_Scalar */
665	//	FMLS	//0x5F805000,/* FMLS vfmlsby_element_Scalar */
666	//	FMUL	//0x5F809000,/* FMUL vfmulby_element_Scalar */
667	//	FMULX	//0x7F809000,/* FMULX vfmulxby_element_Scalar */
668	//	**AdvSIMD scalar shift by imme**	/* AdvSIMD scalar shift by immediate */
669	//	SSHR	//0x5F000400,/* SSHR vsshrScalarimmh != 0000 */
670	//	SSRA	//0x5F001400,/* SSRA vssraScalarimmh != 0000 */
671	//	SRSHR	//0x5F002400,/* SRSHR vsrshrScalarimmh != 0000 */
672	//	SRSRA	//0x5F003400,/* SRSRA vsrsraScalarimmh != 0000 */
673	//	SHL	//0x5F005400,/* SHL vshlScalarimmh != 0000 */
674	//	SQSHL	//0x5F007400,/* SQSHL vsqshlimmediate_Scalarimmh != 0000 */
675	//	SQSHRN	//0x5F009400,/* SQSHRN vsqshrnScalarimmh != 0000 */
676	//	SQSHRN2	//0x5F009400,/* SQSHRN2 vsqshrn2Scalarimmh != 0000 */
677	//	SQRSHRN	//0x5F009C00,/* SQRSHRN vsqrshrnScalarimmh != 0000 */
678	//	SQRSHRN2	//0x5F009C00,/* SQRSHRN2 vsqrshrn2Scalarimmh != 0000 */
679	//	SCVTF	//0x5F00E400,/* SCVTF vscvtfvector_fixed_point_Scalarimmh != 0000 */
680	//	FCVTZS	//0x5F00FC00,/* FCVTZS vfcvtzsvector_fixed_point_Scalarimmh != 0000 */
681	//	USHR	//0x7F000400,/* USHR vushrScalarimmh != 0000 */
682	//	USRA	//0x7F001400,/* USRA vusraScalarimmh != 0000 */
683	//	URSHR	//0x7F002400,/* URSHR vurshrScalarimmh != 0000 */
684	//	URSRA	//0x7F003400,/* URSRA vursraScalarimmh != 0000 */
685	//	SRI	//0x7F004400,/* SRI vsriScalarimmh != 0000 */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
686	//	SLI	//0x7F005400,/* SLI vsliScalarimmh != 0000 */
687	//	SQSHLU	//0x7F006400,/* SQSHLU vsqshluScalarimmh != 0000 */
688	//	UQSHL	//0x7F007400,/* UQSHL vuqshlimmediate_Scalarimmh != 0000 */
689	//	SQSHRUN	//0x7F008400,/* SQSHRUN vsqshrunScalarimmh != 0000 */
690	//	SQSHRUN2	//0x7F008400,/* SQSHRUN2 vsqshrun2Scalarimmh != 0000 */
691	//	SQRSHRUN	//0x7F008C00,/* SQRSHRUN vsqrshrunScalarimmh != 0000 */
692	//	SQRSHRUN2	//0x7F008C00,/* SQRSHRUN2 vsqrshrun2Scalarimmh != 0000 */
693	//	UQSHRN	//0x7F009400,/* UQSHRN vuqshrnScalarimmh != 0000 */
694	//	UQRSHRN	//0x7F009C00,/* UQRSHRN vuqrshrnScalarimmh != 0000 */
695	//	UQRSHRN2	//0x7F009C00,/* UQRSHRN2 vuqrshrn2Scalarimmh != 0000 */
696	//	UCVTF	//0x7F00E400,/* UCVTF vucvtfvector_fixed_point_Scalarimmh != 0000 */
697	//	FCVTZU	//0x7F00FC00,/* FCVTZU vfcvtzuvector_fixed_point_Scalarimmh != 0000 */
698	//	**Crypto three-reg SHA**	/* Crypto three-reg SHA */
699	//	SHA1C	//0x5E000000,/* SHA1C sha1c */
700	//	SHA1P	//0x5E001000,/* SHA1P sha1p */
701	//	SHA1M	//0x5E002000,/* SHA1M sha1m */
702	//	SHA1SU0	//0x5E003000,/* SHA1SU0 sha1su0 */
703	//	SHA256H	//0x5E004000,/* SHA256H sha256h */
704	//	SHA256H2	//0x5E005000,/* SHA256H2 sha256h2 */
705	//	SHA256SU1	//0x5E006000,/* SHA256SU1 sha256su1 */
706	//	**Crypto two-reg SHA**	/* Crypto two-reg SHA */
707	//	SHA1H	//0x5E280800,/* SHA1H sha1h */
708	//	SHA1SU1	//0x5E281800,/* SHA1SU1 sha1su1 */
709	//	SHA256SU0	//0x5E282800,/* SHA256SU0 sha256su0 */
710	//	**Crypto AES**	/* Crypto AES */
711	//	AESE	//0x4E284800,/* AESE aese */
712	//	AESD	//0x4E285800,/* AESD aesd */
713	//	AESMC	//0x4E286800,/* AESMC aesmc */
714	//	AESIMC	//0x4E287800,/* AESIMC aesimc */
715	//	**AdvSIMD three same**	/* AdvSIMD three same */
716	//	SHADD	//0x0E200400,/* SHADD shadd */
717	//	SQADD	//0x0E200C00,/* SQADD vsqaddVector */
718	//	SRHADD	//0x0E201400,/* SRHADD srhadd */
719	//	SHSUB	//0x0E202400,/* SHSUB shsub */

1	in_use	Opcode	//BINARY Opcode	Opcodecomments
720	//	SQSUB	//0x0E202C00,/* SQSUB	vsqsubVector */
721	//	CMGT	//0x0E203400,/* CMGT	vcmgtregister_Vector */
722	//	CMGE	//0x0E203C00,/* CMGE	vcmgeregister_Vector */
723	//	SSHL Vector	//0x0E204400,/* SSHL Vectosshl vector */	
724	//	SQSHL	//0x0E204C00,/* SQSHL	vsqshlregister_Vector */
725	//	SRSHL	//0x0E205400,/* SRSHL	vsrshlVector */
726	//	SQRSHL	//0x0E205C00,/* SQRSHL	vsqrshlVector */
727	//	SMAX	//0x0E206400,/* SMAX	smax */
728	//	SMIN	//0x0E206C00,/* SMIN	smin */
729	//	SABD	//0x0E207400,/* SABD	sabd */
730	//	SABA	//0x0E207C00,/* SABA	saba */
731	//	ADD	//0x0E208400,/* ADD	vaddvector_Vector */
732	//	CMTST	//0x0E208C00,/* CMTST	vcmtstVector */
733	//	MLA	//0x0E209400,/* MLA	mlavector */
734	//	MUL	//0x0E209C00,/* MUL	mulvector */
735	//	SMAXP	//0x0E20A400,/* SMAXP	smaxp */
736	//	SMINP	//0x0E20AC00,/* SMINP	sminp */
737	//	SQDMULH	//0x0E20B400,/* SQDMULH	vsqdmulhvector_Vector */
738	//	ADDP	//0x0E20BC00,/* ADDP	addpvector */
739	//	FMAXNM	//0x0E20C400,/* FMAXNM	fmaxnmvector */
740	//	FMLA	//0x0E20CC00,/* FMLA	fmlavector */
741	//	FADD	//0x0E20D400,/* FADD	faddvector */
742	//	FMULX	//0x0E20DC00,/* FMULX	vfmulxVector */
743	//	FCMEQ	//0x0E20E400,/* FCMEQ	vfcmeqregister_Vector */
744	//	FMAX	//0x0E20F400,/* FMAX	fmaxvector */
745	//	FRECPS	//0x0E20FC00,/* FRECPS	vfrecpsVector */
746	//	AND	//0x0E201C00,/* AND	andvector */
747	//	BIC	//0x0E601C00,/* BIC	bicvector_register */
748	//	FMINNM	//0x0EA0C400,/* FMINNM	fminnmvector */
749	//	FMLS	//0x0EA0CC00,/* FMLS	fmlsvector */
750	//	FSUB	//0x0EA0D400,/* FSUB	fsubvector */
751	//	FMIN	//0x0EA0F400,/* FMIN	fminvector */
752	//	FRSQRTS	//0x0EA0FC00,/* FRSQRTS	vfrsqrtsVector */
753	//	ORR	//0x0EA01C00,/* ORR	orrvector_register */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
754	//	ORN	//0x0EE01C00,/* ORN ornvector */
755	//	UHADD	//0x2E200400,/* UHADD uhadd */
756	//	UQADD	//0x2E200C00,/* UQADD vuqaddVector */
757	//	URHADD	//0x2E201400,/* URHADD urhadd */
758	//	UHSUB	//0x2E202400,/* UHSUB uhsub */
759	//	UQSUB	//0x2E202C00,/* UQSUB vuqsubVector */
760	//	CMHI	//0x2E203400,/* CMHI vcmhiregister_Vector */
761	//	CMHS	//0x2E203C00,/* CMHS vcmhsregister_Vector */
762	//	USHL	//0x2E204400,/* USHL vushlVector */
763	//	UQSHL	//0x2E204C00,/* UQSHL vuqshlregister_Vector */
764	//	URSHL	//0x2E205400,/* URSHL vurshlVector */
765	//	UQRSHL	//0x2E205C00,/* UQRSHL vuqrshlVector */
766	//	UMAX	//0x2E206400,/* UMAX umax */
767	//	UMIN	//0x2E206C00,/* UMIN umin */
768	//	UABD	//0x2E207400,/* UABD uabd */
769	//	UABA	//0x2E207C00,/* UABA uaba */
770	//	SUB	//0x2E208400,/* SUB vsubvector_Vector */
771	//	CMEQ	//0x2E208C00,/* CMEQ vcmeqregister_Vector */
772	//	MLS	//0x2E209400,/* MLS mlsvector */
773	//	PMUL	//0x2E209C00,/* PMUL pmul */
774	//	UMAXP	//0x2E20A400,/* UMAXP umaxp */
775	//	UMINP	//0x2E20AC00,/* UMINP uminp */
776	//	SQRDMULH	//0x2E20B400,/* SQRDMULH vsqrdmulhvector_Vector */
777	//	FMAXNMP	//0x2E20C400,/* FMAXNMP fmaxnmpvector */
778	//	FADDP	//0x2E20D400,/* FADDP faddpvector */
779	//	FMUL	//0x2E20DC00,/* FMUL fmulvector */
780	//	FCMGE	//0x2E20E400,/* FCMGE vfcmgeregister_Vector */
781	//	FACGE	//0x2E20EC00,/* FACGE vfacgeVector */
782	//	FMAXP	//0x2E20F400,/* FMAXP fmaxpvector */
783	//	FDIV	//0x2E20FC00,/* FDIV fdivvector */
784	//	EOR	//0x2E201C00,/* EOR eorvector */
785	//	BSL	//0x2E601C00,/* BSL bsl */
786	//	FMINNMP	//0x2EA0C400,/* FMINNMP fminnmpvector */
787	//	FABD	//0x2EA0D400,/* FABD vfabdVector */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
788	//	FCMGT	//0x2EA0E400,/* FCMGT vfcmgtregister_Vector */
789	//	FACGT	//0x2EA0EC00,/* FACGT vfacgtVector */
790	//	FMINP	//0x2EA0F400,/* FMINP fminpvector */
791	//	BIT	//0x2EA01C00,/* BIT bit */
792	//	BIF	//0x2EE01C00,/* BIF bif */
793	//	**AdvSIMD three different**	/* AdvSIMD three different */
794	//	SADDL	//0x0E200000,/* SADDL saddlwrites to low half of the dest. register & clears the
795	//	SADDL2	//0x4E200000,/* SADDL2 saddl2writes to high half of the dest. register & don't t
796	//	SADDW	//0x0E201000,/* SADDW saddwwrites to low half of the dest. register & clears t
797	//	SADDW2	//0x4E201000,/* SADDW2 saddw2writes to high half of the dest. register & don't
798	//	SSUBL	//0x0E202000,/* SSUBL ssublwrites to low half of the dest. register & clears the
799	//	SSUBL2	//0x4E202000,/* SSUBL2 ssubl2writes to high half of the dest. register & don't t
800	//	SSUBW	//0x0E203000,/* SSUBW ssubwwrites to low half of the dest. register & clears t
801	//	SSUBW2	//0x4E203000,/* SSUBW2 ssubw2writes to high half of the dest. register & don't
802	//	ADDHN	//0x0E204000,/* ADDHN addhnwrites to low half of the dest. register & clears th
803	//	ADDHN2	//0x4E204000,/* ADDHN2 addhn2writes to high half of the dest. register & don't
804	//	SABAL	//0x0E205000,/* SABAL sabalwrites to low half of the dest. register & clears the
805	//	SABAL2	//0x4E205000,/* SABAL2 sabal2writes to high half of the dest. register & don't t
806	//	SUBHN	//0x0E206000,/* SUBHN subhnwrites to low half of the dest. register & clears th
807	//	SUBHN2	//0x4E206000,/* SUBHN2 subhn2writes to high half of the dest. register & don't
808	//	SABDL	//0x0E207000,/* SABDL sabdlwrites to low half of the dest. register & clears the
809	//	SABDL2	//0x4E207000,/* SABDL2 sabdl2writes to high half of the dest. register & don't t
810	//	SMLAL	//0x0E208000,/* SMLAL smlalvectorwrites to low half of the dest. register & clea
811	//	SMLAL2	//0x4E208000,/* SMLAL2 smlal2vectorwrites to high half of the dest. register &
812	//	SQDMLAL	//0x0E209000,/* SQDMLAL vsqdmlalvector_Vectorwrites to low half of the dest.
813	//	SQDMLAL2	//0x4E209000,/* SQDMLAL2 vsqdmlal2vector_Vectorwrites to high half of the des
814	//	SMLSL	//0x0E20A000,/* SMLSL smlslvectorwrites to low half of the dest. register & clea
815	//	SMLSL2	//0x4E20A000,/* SMLSL2 smlsl2vectorwrites to high half of the dest. register &
816	//	SQDMLSL	//0x0E20B000,/* SQDMLSL vsqdmlslvector_Vectorwrites to low half of the dest.
817	//	SQDMLSL2	//0x4E20B000,/* SQDMLSL2 vsqdmlsl2vector_Vectorwrites to high half of the des
818	//	SMULL	//0x0E20C000,/* SMULL smullvectorwrites to low half of the dest. register & cle
819	//	SMULL2	//0x4E20C000,/* SMULL2 smull2vectorwrites to high half of the dest. register &
820	//	SQDMULL	//0x0E20D000,/* SQDMULL vsqdmullvector_Vectorwrites to low half of the dest.
821	//	SQDMULL2	//0x4E20D000,/* SQDMULL2 vsqdmull2vector_Vectorwrites to high half of the de

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
822	//	PMULL	//0x0E20E000,/*	PMULL	pmullwrites to low half of the dest. register & clears the
823	//	PMULL2	//0x4E20E000,/*	PMULL2	pmull2writes to high half of the dest. register & don't t
824	//	UADDL	//0x2E200000,/*	UADDL	uaddlwrites to low half of the dest. register & clears the
825	//	UADDL2	//0x6E200000,/*	UADDL2	uaddl2writes to high half of the dest. register & don't t
826	//	UADDW	//0x2E201000,/*	UADDW	uaddwwrites to low half of the dest. register & clears t
827	//	UADDW2	//0x6E201000,/*	UADDW2	uaddw2writes to high half of the dest. register & don'
828	//	USUBL	//0x2E202000,/*	USUBL	usublwrites to low half of the dest. register & clears the
829	//	USUBL2	//0x6E202000,/*	USUBL2	usubl2writes to high half of the dest. register & don't t
830	//	USUBW	//0x2E203000,/*	USUBW	usubwwrites to low half of the dest. register & clears t
831	//	USUBW2	//0x6E203000,/*	USUBW2	usubw2writes to high half of the dest. register & don't
832	//	RADDHN	//0x2E204000,/*	RADDHN	raddhnwrites to low half of the dest. register & clears
833	//	RADDHN2	//0x6E204000,/*	RADDHN2	raddhn2writes to high half of the dest. register & dor
834	//	UABAL	//0x2E205000,/*	UABAL	uabalwrites to low half of the dest. register & clears the
835	//	UABAL2	//0x6E205000,/*	UABAL2	uabal2writes to high half of the dest. register & don't t
836	//	RSUBHN	//0x2E206000,/*	RSUBHN	rsubhnwrites to low half of the dest. register & clears
837	//	RSUBHN2	//0x6E206000,/*	RSUBHN2	rsubhn2writes to high half of the dest. register & don
838	//	UABDL	//0x2E207000,/*	UABDL	uabdlwrites to low half of the dest. register & clears the
839	//	UABDL2	//0x6E207000,/*	UABDL2	uabdl2writes to high half of the dest. register & don't t
840	//	UMLAL	//0x2E208000,/*	UMLAL	umlalvectorwrites to low half of the dest. register & cle
841	//	UMLAL2	//0x6E208000,/*	UMLAL2	umlal2vectorwrites to high half of the dest. register &
842	//	UMLSL	//0x2E20A000,/*	UMLSL	umlslvectorwrites to low half of the dest. register & cle
843	//	UMLSL2	//0x6E20A000,/*	UMLSL2	umlsl2vectorwrites to high half of the dest. register &
844	//	UMULL	//0x2E20C000,/*	UMULL	umullvectorwrites to low half of the dest. register & cle
845	//	UMULL2	//0x6E20C000,/*	UMULL2	umull2vectorwrites to high half of the dest. register &
846	//	**AdvSIMD two-reg misc**	/* AdvSIMD two-reg misc */		
847	//	REV64	//0x0E200800,/*	REV64	rev64 */
848	//	REV16	//0x0E201800,/*	REV16	rev16vector */
849	//	SADDLP	//0x0E202800,/*	SADDLP	saddlp */
850	//	SUQADD	//0x0E203800,/*	SUQADD	vsuqaddVector */
851	//	CLS	//0x0E204800,/*	CLS	clsvector */
852	//	CNT	//0x0E205800,/*	CNT	cnt */
853	//	SADALP	//0x0E206800,/*	SADALP	sadalp */
854	//	SQABS	//0x0E207800,/*	SQABS	vsqabsVector */
855	//	CMGT	//0x0E208800,/*	CMGT	vcmgtzero_Vector */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
856	//	CMEQ	//0x0E209800,/* CMEQ vcmeqzero_Vector */
857	//	CMLT	//0x0E20A800,/* CMLT vcmltzero_Vector */
858	//	ABS	//0x0E20B800,/* ABS vabsVector */
859	//	XTN	//0x0E212800,/* XTN xtn */
860	//	XTN2	//0x0E212800,/* XTN2 xtn2 */
861	//	SQXTN	//0x0E214800,/* SQXTN vsqxtnVector */
862	//	SQXTN2	//0x0E214800,/* SQXTN2 vsqxtn2Vector */
863	//	FCVTN	//0x0E216800,/* FCVTN fcvtn */
864	//	FCVTN2	//0x0E216800,/* FCVTN2 fcvtn2 */
865	//	FCVTL	//0x0E217800,/* FCVTL fcvtl */
866	//	FCVTL2	//0x0E217800,/* FCVTL2 fcvtl2 */
867	//	FRINTN	//0x0E218800,/* FRINTN frintnvector */
868	//	FRINTM	//0x0E219800,/* FRINTM frintmvector */
869	//	FCVTNS	//0x0E21A800,/* FCVTNS vfcvtnsvector_Vector */
870	//	FCVTMS	//0x0E21B800,/* FCVTMS vfcvtmsvector_Vector */
871	//	FCVTAS	//0x0E21C800,/* FCVTAS vfcvtasvector_Vector */
872	//	SCVTF	//0x0E21D800,/* SCVTF vscvtfvector_integer_Vector */
873	//	FCMGT	//0x0EA0C800,/* FCMGT vfcmgtzero_Vector */
874	//	FCMEQ	//0x0EA0D800,/* FCMEQ vfcmeqzero_Vector */
875	//	FCMLT	//0x0EA0E800,/* FCMLT vfcmltzero_Vector */
876	//	FABS	//0x0EA0F800,/* FABS fabsvector */
877	//	FRINTP	//0x0EA18800,/* FRINTP frintpvector */
878	//	FRINTZ	//0x0EA19800,/* FRINTZ frintzvector */
879	//	FCVTPS	//0x0EA1A800,/* FCVTPS vfcvtpsvector_Vector */
880	//	FCVTZS	//0x0EA1B800,/* FCVTZS vfcvtzsvector_integer_Vector */
881	//	URECPE	//0x0EA1C800,/* URECPE urecpe */
882	//	FRECPE	//0x0EA1D800,/* FRECPE vfrecpeVector */
883	//	REV32	//0x2E200800,/* REV32 rev32vector */
884	//	UADDLP	//0x2E202800,/* UADDLP uaddlp */
885	//	USQADD	//0x2E203800,/* USQADD vusqaddVector */
886	//	CLZ	//0x2E204800,/* CLZ clzvector */
887	//	UADALP	//0x2E206800,/* UADALP uadalp */
888	//	SQNEG	//0x2E207800,/* SQNEG vsqnegVector */
889	//	CMGE	//0x2E208800,/* CMGE vcmgezero_Vector */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
890	//	CMLE	//0x2E209800,/* CMLE vcmlezero_Vector */
891	//	NEG	//0x2E20B800,/* NEG vnegvector_Vector */
892	//	SQXTUN	//0x2E212800,/* SQXTUN vsqxtunVector */
893	//	SQXTUN2	//0x2E212800,/* SQXTUN2 vsqxtun2Vector */
894	//	SHLL	//0x2E213800,/* SHLL shll */
895	//	SHLL2	//0x2E213800,/* SHLL2 shll2 */
896	//	UQXTN	//0x2E214800,/* UQXTN vuqxtnVector */
897	//	UQXTN2	//0x2E214800,/* UQXTN2 vuqxtn2Vector */
898	//	FCVTXN	//0x2E216800,/* FCVTXN vfcvtxnVector */
899	//	FCVTXN2	//0x2E216800,/* FCVTXN2 vfcvtxn2Vector */
900	//	FRINTA	//0x2E218800,/* FRINTA frintavector */
901	//	FRINTX	//0x2E219800,/* FRINTX frintxvector */
902	//	FCVTNU	//0x2E21A800,/* FCVTNU vfcvtnuvector_Vector */
903	//	FCVTMU	//0x2E21B800,/* FCVTMU vfcvtmuvector_Vector */
904	//	FCVTAU	//0x2E21C800,/* FCVTAU vfcvtauvector_Vector */
905	//	UCVTF	//0x2E21D800,/* UCVTF vucvtfvector_integer_Vector */
906	//	NOT	//0x2E205800,/* NOT not */
907	//	RBIT	//0x2E605800,/* RBIT rbitvector */
908	//	FCMGE	//0x2EA0C800,/* FCMGE vfcmgezero_Vector */
909	//	FCMLE	//0x2EA0D800,/* FCMLE vfcmlezero_Vector */
910	//	FNEG	//0x2EA0F800,/* FNEG fnegvector */
911	//	FRINTI	//0x2EA19800,/* FRINTI frintivector */
912	//	FCVTPU	//0x2EA1A800,/* FCVTPU vfcvtpuvector_Vector */
913	//	FCVTZU	//0x2EA1B800,/* FCVTZU vfcvtzuvector_integer_Vector */
914	//	URSQRTE	//0x2EA1C800,/* URSQRTE ursqrte */
915	//	FRSQRTE	//0x2EA1D800,/* FRSQRTE vfrsqrteVector */
916	//	FSQRT	//0x2EA1F800,/* FSQRT fsqrtvector */
917	//	**AdvSIMD across lanes**	/* AdvSIMD across lanes */
918	//	SADDLV	//0x0E303800,/* SADDLV saddlv */
919	//	SMAXV	//0x0E30A800,/* SMAXV smaxv */
920	//	SMINV	//0x0E31A800,/* SMINV sminv */
921	//	ADDV	//0x0E31B800,/* ADDV addv */
922	//	UADDLV	//0x2E303800,/* UADDLV uaddlv */
923	//	UMAXV	//0x2E30A800,/* UMAXV umaxv */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
924	//	UMINV	//0x2E31A800,/* UMINV uminv */
925	//	FMAXNMV	//0x2E30C800,/* FMAXNMV fmaxnmv */
926	//	FMAXV	//0x2E30F800,/* FMAXV fmaxv */
927	//	FMINNMV	//0x2EB0C800,/* FMINNMV fminnmv */
928	//	FMINV	//0x2EB0F800,/* FMINV fminv */
929	//	**AdvSIMD copy**	/* AdvSIMD copy */
930	//	DUP	//0x0E000400,/* DUP vdupelement_Vector */
931	//	DUP	//0x0E000C00,/* DUP dupgeneral */
932	//	SMOV	//0x0E002C00,/* SMOV vsmov32_bit */
933	//	UMOV	//0x0E003C00,/* UMOV vumov32_bit */
934	//	INS	//0x4E001C00,/* INS insgeneral */
935	//	SMOV	//0x4E002C00,/* SMOV vsmov64_bit */
936	//	UMOV	//0x4E003C00,/* UMOV vumov64_bit */
937	//	INS	//0x6E000400,/* INS inselement */
938	//	**AdvSIMD vector x indexed ele**	/* AdvSIMD vector x indexed element */
939	//	SMLAL	//0x0F002000,/* SMLAL smlalby_element */
940	//	SMLAL2	//0x0F002000,/* SMLAL2 smlal2by_element */
941	//	SQDMLAL	//0x0F003000,/* SQDMLAL vsqdmlalby_element_Vector */
942	//	SQDMLAL2	//0x0F003000,/* SQDMLAL2 vsqdmlal2by_element_Vector */
943	//	SMLSL	//0x0F006000,/* SMLSL smlslby_element */
944	//	SMLSL2	//0x0F006000,/* SMLSL2 smlsl2by_element */
945	//	SQDMLSL	//0x0F007000,/* SQDMLSL vsqdmlslby_element_Vector */
946	//	SQDMLSL2	//0x0F007000,/* SQDMLSL2 vsqdmlsl2by_element_Vector */
947	//	MUL	//0x0F008000,/* MUL mulby_element */
948	//	SMULL	//0x0F00A000,/* SMULL smullby_element */
949	//	SMULL2	//0x0F00A000,/* SMULL2 smull2by_element */
950	//	SQDMULL	//0x0F00B000,/* SQDMULL vsqdmullby_element_Vector */
951	//	SQDMULL2	//0x0F00B000,/* SQDMULL2 vsqdmull2by_element_Vector */
952	//	SQDMULH	//0x0F00C000,/* SQDMULH vsqdmulhby_element_Vector */
953	//	SQRDMULH	//0x0F00D000,/* SQRDMULH vsqrdmulhby_element_Vector */
954	//	FMLA	//0x0F801000,/* FMLA vfmlaby_element_Vector */
955	//	FMLS	//0x0F805000,/* FMLS vfmlsby_element_Vector */
956	//	FMUL	//0x0F809000,/* FMUL vfmulby_element_Vector */
957	//	MLA	//0x2F000000,/* MLA mlaby_element */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
958	//	UMLAL	//0x2F002000,/*	UMLAL	umlalby_element */
959	//	UMLAL2	//0x2F002000,/*	UMLAL2	umlal2by_element */
960	//	MLS	//0x2F004000,/*	MLS	mlsby_element */
961	//	UMLSL	//0x2F006000,/*	UMLSL	umlslby_element */
962	//	UMLSL2	//0x2F006000,/*	UMLSL2	umlsl2by_element */
963	//	UMULL	//0x2F00A000,/*	UMULL	umullby_element */
964	//	UMULL2	//0x2F00A000,/*	UMULL2	umull2by_element */
965	//	FMULX	//0x2F809000,/*	FMULX	vfmulxby_element_Vector */
966	//	**AdvSIMD modified immediate**	/* AdvSIMD modified immediate */		
967	//	MOVI	//0x0F000400,/*	MOVI	vmovi32_bit_shifted_immediate */
968	//	ORR	//0x0F001400,/*	ORR	vorrvector_immediate_32_bit */
969	//	MOVI	//0x0F008400,/*	MOVI	vmovi16_bit_shifted_immediate */
970	//	ORR	//0x0F009400,/*	ORR	vorrvector_immediate_16_bit */
971	//	MOVI	//0x0F00C400,/*	MOVI	vmovi32_bit_shifting_ones */
972	//	MOVI	//0x0F00E400,/*	MOVI	vmovi8_bit */
973	//	FMOV	//0x0F00F400,/*	FMOV	vfmovvector_immediate_Single_precision */
974	//	MVNI	//0x2F000400,/*	MVNI	vmvni32_bit_shifted_immediate */
975	//	BIC	//0x2F001400,/*	BIC	vbicvector_immediate_32_bit */
976	//	MVNI	//0x2F008400,/*	MVNI	vmvni16_bit_shifted_immediate */
977	//	BIC	//0x2F009400,/*	BIC	vbicvector_immediate_16_bit */
978	//	MVNI	//0x2F00C400,/*	MVNI	vmvni32_bit_shifting_ones */
979	//	MOVI	//0x2F00E400,/*	MOVI	vmovi64_bit_scalar */
980	//	MOVI	//0x6F00E400,/*	MOVI	vmovi64_bit_vector */
981	//	FMOV	//0x6F00F400,/*	FMOV	vfmovvector_immediate_Double_precision */
982	//	**AdvSIMD shift by immediate**	/* AdvSIMD shift by immediate */		
983	//	SSHR	//0x0F000400,/*	SSHR	vsshrVector */
984	//	SSRA	//0x0F001400,/*	SSRA	vssraVector */
985	//	SRSHR	//0x0F002400,/*	SRSHR	vsrshrVector */
986	//	SRSRA	//0x0F003400,/*	SRSRA	vsrsraVector */
987	//	SHL	//0x0F005400,/*	SHL	vshlVector */
988	//	SQSHL	//0x0F007400,/*	SQSHL	vsqshlimmediate_Vector */
989	//	SHRN	//0x0F008400,/*	SHRN	shrn */
990	//	SHRN2	//0x0F008400,/*	SHRN2	shrn2 */
991	//	RSHRN	//0x0F008C00,/*	RSHRN	rshrn */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
992	//	RSHRN2	//0x0F008C00,/* RSHRN2 rshrn2 */
993	//	SQSHRN	//0x0F009400,/* SQSHRN vsqshrnVector */
994	//	SQSHRN2	//0x0F009400,/* SQSHRN2 vsqshrn2Vector */
995	//	SQRSHRN	//0x0F009C00,/* SQRSHRN vsqrshrnVector */
996	//	SQRSHRN2	//0x0F009C00,/* SQRSHRN2 vsqrshrn2Vector */
997	//	SSHLL	//0x0F00A400,/* SSHLL sshll */
998	//	SSHLL2	//0x0F00A400,/* SSHLL2 sshll2 */
999	//	SCVTF	//0x0F00E400,/* SCVTF vscvtfvector_fixed_point_Vector */
1000	//	FCVTZS	//0x0F00FC00,/* FCVTZS vfcvtzsvector_fixed_point_Vector */
1001	//	USHR	//0x2F000400,/* USHR vushrVector */
1002	//	USRA	//0x2F001400,/* USRA vusraVector */
1003	//	URSHR	//0x2F002400,/* URSHR vurshrVector */
1004	//	URSRA	//0x2F003400,/* URSRA vursraVector */
1005	//	SRI	//0x2F004400,/* SRI vsriVector */
1006	//	SLI	//0x2F005400,/* SLI vsliVector */
1007	//	SQSHLU	//0x2F006400,/* SQSHLU vsqshluVector */
1008	//	UQSHL	//0x2F007400,/* UQSHL vuqshlimmediate_Vector */
1009	//	SQSHRUN	//0x2F008400,/* SQSHRUN vsqshrunVector */
1010	//	SQSHRUN2	//0x2F008400,/* SQSHRUN2 vsqshrun2Vector */
1011	//	SQRSHRUN	//0x2F008C00,/* SQRSHRUN vsqrshrunVector */
1012	//	SQRSHRUN2	//0x2F008C00,/* SQRSHRUN2 vsqrshrun2Vector */
1013	//	UQSHRN	//0x2F009400,/* UQSHRN vuqshrnVector */
1014	//	UQRSHRN	//0x2F009C00,/* UQRSHRN vuqrshrnVector */
1015	//	UQRSHRN2	//0x2F009C00,/* UQRSHRN2 vuqrshrn2Vector */
1016	//	USHLL	//0x2F00A400,/* USHLL ushll */
1017	//	USHLL2	//0x2F00A400,/* USHLL2 ushll2 */
1018	//	UCVTF	//0x2F00E400,/* UCVTF vucvtfvector_fixed_point_Vector */
1019	//	FCVTZU	//0x2F00FC00,/* FCVTZU vfcvtzuvector_fixed_point_Vector */
1020	//	**AdvSIMD TBL/TBX**	/* AdvSIMD TBL/TBX */
1021	//	TBL	//0x0E000000,/* TBL vtblSingle_register_table */
1022	//	TBX	//0x0E001000,/* TBX vtbxSingle_register_table */
1023	//	TBL	//0x0E002000,/* TBL vtblTwo_register_table */
1024	//	TBX	//0x0E003000,/* TBX vtbxTwo_register_table */
1025	//	TBL	//0x0E004000,/* TBL vtblThree_register_table */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1026	//	TBX	//0x0E005000,/* TBX	vtbxThree_register_table */	
1027	//	TBL	//0x0E006000,/* TBL	vtblFour_register_table */	
1028	//	TBX	//0x0E007000,/* TBX	vtbxFour_register_table */	
1029	//	**AdvSIMD ZIP/UZP/TRN**	/* AdvSIMD ZIP/UZP/TRN */		
1030	//	UZP1	//0x0E001800,/* UZP1	uzp1 */	
1031	//	TRN1	//0x0E002800,/* TRN1	trn1 */	
1032	//	ZIP1	//0x0E003800,/* ZIP1	zip1 */	
1033	//	UZP2	//0x0E005800,/* UZP2	uzp2 */	
1034	//	TRN2	//0x0E006800,/* TRN2	trn2 */	
1035	//	ZIP2	//0x0E007800,/* ZIP2	zip2 */	
1036	//	**AdvSIMD EXT**	/* AdvSIMD EXT */		
1037	//	EXT	//0x2E000000,/* EXT	ext */	
1038	//	**Loads and stores**	/* Loads and stores */		
1039	//	**AdvSIMD load/store multiple s**	/* AdvSIMD load/store multiple structures */		
1040	//	ST4	//0x0C000000,/* ST4	vst4multiple_structures_No_offset */	
1041	//	ST1	//0x0C002000,/* ST1	vst1multiple_structures_Four_registers */	
1042	//	ST3	//0x0C004000,/* ST3	vst3multiple_structures_No_offset */	
1043	//	ST1	//0x0C006000,/* ST1	vst1multiple_structures_Three_registers */	
1044	//	ST1	//0x0C007000,/* ST1	vst1multiple_structures_One_register */	
1045	//	ST2	//0x0C008000,/* ST2	vst2multiple_structures_No_offset */	
1046	//	ST1	//0x0C00A000,/* ST1	vst1multiple_structures_Two_registers */	
1047	//	LD4	//0x0C400000,/* LD4	vld4multiple_structures_No_offset */	
1048	//	LD1	//0x0C402000,/* LD1	vld1multiple_structures_Four_registers */	
1049	//	LD3	//0x0C404000,/* LD3	vld3multiple_structures_No_offset */	
1050	//	LD1	//0x0C406000,/* LD1	vld1multiple_structures_Three_registers */	
1051	//	LD1	//0x0C407000,/* LD1	vld1multiple_structures_One_register */	
1052	//	LD2	//0x0C408000,/* LD2	vld2multiple_structures_No_offset */	
1053	//	LD1	//0x0C40A000,/* LD1	vld1multiple_structures_Two_registers */	
1054	//	**AdvSIMD load/store multiple s**	/* AdvSIMD load/store multiple structures (post-indexed) */		
1055	//	ST4	//0x0C800000,/* ST4	vst4multiple_structures_Register_offsetRm != 11111 */	
1056	//	ST1	//0x0C802000,/* ST1	vst1multiple_structures_Four_registers_register_offsetR	
1057	//	ST3	//0x0C804000,/* ST3	vst3multiple_structures_Register_offsetRm != 11111 */	
1058	//	ST1	//0x0C806000,/* ST1	vst1multiple_structures_Three_registers_register_offsetl	
1059	//	ST1	//0x0C807000,/* ST1	vst1multiple_structures_One_register_register_offsetRm	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1060	//	ST2	//0x0C808000,/* ST2	vst2multiple_structures_Register_offsetRm != 11111 */	
1061	//	ST1	//0x0C80A000,/* ST1	vst1multiple_structures_Two_registers_register_offsetR	
1062	//	ST4	//0x0C9F0000,/* ST4	vst4multiple_structures_Immediate_offset */	
1063	//	ST1	//0x0C9F2000,/* ST1	vst1multiple_structures_Four_registers_immediate_offse	
1064	//	ST3	//0x0C9F4000,/* ST3	vst3multiple_structures_Immediate_offset */	
1065	//	ST1	//0x0C9F6000,/* ST1	vst1multiple_structures_Three_registers_immediate_offs	
1066	//	ST1	//0x0C9F7000,/* ST1	vst1multiple_structures_One_register_immediate_offset	
1067	//	ST2	//0x0C9F8000,/* ST2	vst2multiple_structures_Immediate_offset */	
1068	//	ST1	//0x0C9FA000,/* ST1	vst1multiple_structures_Two_registers_immediate_offse	
1069	//	LD4	//0x0CC00000,/* LD4	vld4multiple_structures_Register_offsetRm != 11111 */	
1070	//	LD1	//0x0CC02000,/* LD1	vld1multiple_structures_Four_registers_register_offsetR	
1071	//	LD3	//0x0CC04000,/* LD3	vld3multiple_structures_Register_offsetRm != 11111 */	
1072	//	LD1	//0x0CC06000,/* LD1	vld1multiple_structures_Three_registers_register_offset	
1073	//	LD1	//0x0CC07000,/* LD1	vld1multiple_structures_One_register_register_offsetRn	
1074	//	LD2	//0x0CC08000,/* LD2	vld2multiple_structures_Register_offsetRm != 11111 */	
1075	//	LD1	//0x0CC0A000,/* LD1	vld1multiple_structures_Two_registers_register_offsetR	
1076	//	LD4	//0x0CDF0000,/* LD4	vld4multiple_structures_Immediate_offset */	
1077	//	LD1	//0x0CDF2000,/* LD1	vld1multiple_structures_Four_registers_immediate_offs	
1078	//	LD3	//0x0CDF4000,/* LD3	vld3multiple_structures_Immediate_offset */	
1079	//	LD1	//0x0CDF6000,/* LD1	vld1multiple_structures_Three_registers_immediate_off	
1080	//	LD1	//0x0CDF7000,/* LD1	vld1multiple_structures_One_register_immediate_offset	
1081	//	LD2	//0x0CDF8000,/* LD2	vld2multiple_structures_Immediate_offset */	
1082	//	LD1	//0x0CDFA000,/* LD1	vld1multiple_structures_Two_registers_immediate_offse	
1083	//	**AdvSIMD load/store single st**	/* AdvSIMD load/store single structure */		
1084	//	ST1	//0x0D000000,/* ST1	vst1single_structure_8_bit */	
1085	//	ST3	//0x0D002000,/* ST3	vst3single_structure_8_bit */	
1086	//	ST1	//0x0D004000,/* ST1	vst1single_structure_16_bit */	
1087	//	ST3	//0x0D006000,/* ST3	vst3single_structure_16_bit */	
1088	//	ST1	//0x0D008000,/* ST1	vst1single_structure_32_bit */	
1089	//	ST1	//0x0D008400,/* ST1	vst1single_structure_64_bit */	
1090	//	ST3	//0x0D00A000,/* ST3	vst3single_structure_32_bit */	
1091	//	ST3	//0x0D00A400,/* ST3	vst3single_structure_64_bit */	
1092	//	ST2	//0x0D200000,/* ST2	vst2single_structure_8_bit */	
1093	//	ST4	//0x0D202000,/* ST4	vst4single_structure_8_bit */	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1094	//	ST2	//0x0D204000,/* ST2	vst2single_structure_16_bit */	
1095	//	ST4	//0x0D206000,/* ST4	vst4single_structure_16_bit */	
1096	//	ST2	//0x0D208000,/* ST2	vst2single_structure_32_bit */	
1097	//	ST2	//0x0D208400,/* ST2	vst2single_structure_64_bit */	
1098	//	ST4	//0x0D20A000,/* ST4	vst4single_structure_32_bit */	
1099	//	ST4	//0x0D20A400,/* ST4	vst4single_structure_64_bit */	
1100	//	LD1	//0x0D400000,/* LD1	vld1single_structure_8_bit */	
1101	//	LD3	//0x0D402000,/* LD3	vld3single_structure_8_bit */	
1102	//	LD1	//0x0D404000,/* LD1	vld1single_structure_16_bit */	
1103	//	LD3	//0x0D406000,/* LD3	vld3single_structure_16_bit */	
1104	//	LD1	//0x0D408000,/* LD1	vld1single_structure_32_bit */	
1105	//	LD1	//0x0D408400,/* LD1	vld1single_structure_64_bit */	
1106	//	LD3	//0x0D40A000,/* LD3	vld3single_structure_32_bit */	
1107	//	LD3	//0x0D40A400,/* LD3	vld3single_structure_64_bit */	
1108	//	LD1R	//0x0D40C000,/* LD1R	vld1rNo_offset */	
1109	//	LD3R	//0x0D40E000,/* LD3R	vld3rNo_offset */	
1110	//	LD2	//0x0D600000,/* LD2	vld2single_structure_8_bit */	
1111	//	LD4	//0x0D602000,/* LD4	vld4single_structure_8_bit */	
1112	//	LD2	//0x0D604000,/* LD2	vld2single_structure_16_bit */	
1113	//	LD4	//0x0D606000,/* LD4	vld4single_structure_16_bit */	
1114	//	LD2	//0x0D608000,/* LD2	vld2single_structure_32_bit */	
1115	//	LD2	//0x0D608400,/* LD2	vld2single_structure_64_bit */	
1116	//	LD4	//0x0D60A000,/* LD4	vld4single_structure_32_bit */	
1117	//	LD4	//0x0D60A400,/* LD4	vld4single_structure_64_bit */	
1118	//	LD2R	//0x0D60C000,/* LD2R	vld2rNo_offset */	
1119	//	LD4R	//0x0D60E000,/* LD4R	vld4rNo_offset */	
1120	//	**AdvSIMD load/store single st**	/* AdvSIMD load/store single structure (post-indexed) */		
1121	//	ST1	//0x0D800000,/* ST1	vst1single_structure_8_bit_register_offsetRm != 11111 *	
1122	//	ST3	//0x0D802000,/* ST3	vst3single_structure_8_bit_register_offsetRm != 11111 *	
1123	//	ST1	//0x0D804000,/* ST1	vst1single_structure_16_bit_register_offsetRm != 11111	
1124	//	ST3	//0x0D806000,/* ST3	vst3single_structure_16_bit_register_offsetRm != 11111	
1125	//	ST1	//0x0D808000,/* ST1	vst1single_structure_32_bit_register_offsetRm != 11111	
1126	//	ST1	//0x0D808400,/* ST1	vst1single_structure_64_bit_register_offsetRm != 11111	
1127	//	ST3	//0x0D80A000,/* ST3	vst3single_structure_32_bit_register_offsetRm != 11111	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1128	//	ST3	//0x0D80A400,/*	ST3	vst3single_structure_64_bit_register_offsetRm != 11111
1129	//	ST1	//0x0D9F0000,/*	ST1	vst1single_structure_8_bit_immediate_offset */
1130	//	ST3	//0x0D9F2000,/*	ST3	vst3single_structure_8_bit_immediate_offset */
1131	//	ST1	//0x0D9F4000,/*	ST1	vst1single_structure_16_bit_immediate_offset */
1132	//	ST3	//0x0D9F6000,/*	ST3	vst3single_structure_16_bit_immediate_offset */
1133	//	ST1	//0x0D9F8000,/*	ST1	vst1single_structure_32_bit_immediate_offset */
1134	//	ST1	//0x0D9F8400,/*	ST1	vst1single_structure_64_bit_immediate_offset */
1135	//	ST3	//0x0D9FA000,/*	ST3	vst3single_structure_32_bit_immediate_offset */
1136	//	ST3	//0x0D9FA400,/*	ST3	vst3single_structure_64_bit_immediate_offset */
1137	//	ST2	//0x0DA00000,/*	ST2	vst2single_structure_8_bit_register_offsetRm != 11111
1138	//	ST4	//0x0DA02000,/*	ST4	vst4single_structure_8_bit_register_offsetRm != 11111
1139	//	ST2	//0x0DA04000,/*	ST2	vst2single_structure_16_bit_register_offsetRm != 11111
1140	//	ST4	//0x0DA06000,/*	ST4	vst4single_structure_16_bit_register_offsetRm != 11111
1141	//	ST2	//0x0DA08000,/*	ST2	vst2single_structure_32_bit_register_offsetRm != 11111
1142	//	ST2	//0x0DA08400,/*	ST2	vst2single_structure_64_bit_register_offsetRm != 11111
1143	//	ST4	//0x0DA0A000,/*	ST4	vst4single_structure_32_bit_register_offsetRm != 11111
1144	//	ST4	//0x0DA0A400,/*	ST4	vst4single_structure_64_bit_register_offsetRm != 11111
1145	//	ST2	//0x0DBF0000,/*	ST2	vst2single_structure_8_bit_immediate_offset */
1146	//	ST4	//0x0DBF2000,/*	ST4	vst4single_structure_8_bit_immediate_offset */
1147	//	ST2	//0x0DBF4000,/*	ST2	vst2single_structure_16_bit_immediate_offset */
1148	//	ST4	//0x0DBF6000,/*	ST4	vst4single_structure_16_bit_immediate_offset */
1149	//	ST2	//0x0DBF8000,/*	ST2	vst2single_structure_32_bit_immediate_offset */
1150	//	ST2	//0x0DBF8400,/*	ST2	vst2single_structure_64_bit_immediate_offset */
1151	//	ST4	//0x0DBFA000,/*	ST4	vst4single_structure_32_bit_immediate_offset */
1152	//	ST4	//0x0DBFA400,/*	ST4	vst4single_structure_64_bit_immediate_offset */
1153	//	LD1	//0x0DC00000,/*	LD1	vld1single_structure_8_bit_register_offsetRm != 11111
1154	//	LD3	//0x0DC02000,/*	LD3	vld3single_structure_8_bit_register_offsetRm != 11111
1155	//	LD1	//0x0DC04000,/*	LD1	vld1single_structure_16_bit_register_offsetRm != 11111
1156	//	LD3	//0x0DC06000,/*	LD3	vld3single_structure_16_bit_register_offsetRm != 11111
1157	//	LD1	//0x0DC08000,/*	LD1	vld1single_structure_32_bit_register_offsetRm != 11111
1158	//	LD1	//0x0DC08400,/*	LD1	vld1single_structure_64_bit_register_offsetRm != 11111
1159	//	LD3	//0x0DC0A000,/*	LD3	vld3single_structure_32_bit_register_offsetRm != 1111
1160	//	LD3	//0x0DC0A400,/*	LD3	vld3single_structure_64_bit_register_offsetRm != 1111
1161	//	LD1R	//0x0DC0C000,/*	LD1R	vld1rRegister_offsetRm != 11111 */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1162	//	LD3R	//0x0DC0E000,/*	LD3R	vld3rRegister_offsetRm != 11111 */
1163	//	LD1	//0x0DDF0000,/*	LD1	vld1single_structure_8_bit_immediate_offset */
1164	//	LD3	//0x0DDF2000,/*	LD3	vld3single_structure_8_bit_immediate_offset */
1165	//	LD1	//0x0DDF4000,/*	LD1	vld1single_structure_16_bit_immediate_offset */
1166	//	LD3	//0x0DDF6000,/*	LD3	vld3single_structure_16_bit_immediate_offset */
1167	//	LD1	//0x0DDF8000,/*	LD1	vld1single_structure_32_bit_immediate_offset */
1168	//	LD1	//0x0DDF8400,/*	LD1	vld1single_structure_64_bit_immediate_offset */
1169	//	LD3	//0x0DDFA000,/*	LD3	vld3single_structure_32_bit_immediate_offset */
1170	//	LD3	//0x0DDFA400,/*	LD3	vld3single_structure_64_bit_immediate_offset */
1171	//	LD1R	//0x0DDFC000,/*	LD1R	vld1rImmediate_offset */
1172	//	LD3R	//0x0DDFE000,/*	LD3R	vld3rImmediate_offset */
1173	//	LD2	//0x0DE00000,/*	LD2	vld2single_structure_8_bit_register_offsetRm != 11111
1174	//	LD4	//0x0DE02000,/*	LD4	vld4single_structure_8_bit_register_offsetRm != 11111
1175	//	LD2	//0x0DE04000,/*	LD2	vld2single_structure_16_bit_register_offsetRm != 11111
1176	//	LD4	//0x0DE06000,/*	LD4	vld4single_structure_16_bit_register_offsetRm != 11111
1177	//	LD2	//0x0DE08000,/*	LD2	vld2single_structure_32_bit_register_offsetRm != 11111
1178	//	LD2	//0x0DE08400,/*	LD2	vld2single_structure_64_bit_register_offsetRm != 11111
1179	//	LD4	//0x0DE0A000,/*	LD4	vld4single_structure_32_bit_register_offsetRm != 11111
1180	//	LD4	//0x0DE0A400,/*	LD4	vld4single_structure_64_bit_register_offsetRm != 11111
1181	//	LD2R	//0x0DE0C000,/*	LD2R	vld2rRegister_offsetRm != 11111 */
1182	//	LD4R	//0x0DE0E000,/*	LD4R	vld4rRegister_offsetRm != 11111 */
1183	//	LD2	//0x0DFF0000,/*	LD2	vld2single_structure_8_bit_immediate_offset */
1184	//	LD4	//0x0DFF2000,/*	LD4	vld4single_structure_8_bit_immediate_offset */
1185	//	LD2	//0x0DFF4000,/*	LD2	vld2single_structure_16_bit_immediate_offset */
1186	//	LD4	//0x0DFF6000,/*	LD4	vld4single_structure_16_bit_immediate_offset */
1187	//	LD2	//0x0DFF8000,/*	LD2	vld2single_structure_32_bit_immediate_offset */
1188	//	LD2	//0x0DFF8400,/*	LD2	vld2single_structure_64_bit_immediate_offset */
1189	//	LD4	//0x0DFFA000,/*	LD4	vld4single_structure_32_bit_immediate_offset */
1190	//	LD4	//0x0DFFA400,/*	LD4	vld4single_structure_64_bit_immediate_offset */
1191	//	LD2R	//0x0DFFC000,/*	LD2R	vld2rImmediate_offset */
1192	//	LD4R	//0x0DFFE000,/*	LD4R	vld4rImmediate_offset */