	in_use	Opcode	Extended Name	Specific	variant
2		**UNALLOCATED**			
3		BAD			
4		**Branch,exception generation and syst**			
5		**Compare _ Branch (immediate)**			
6		CBZ	W		W
7		CBNZ	W		W
8		CBZ	X		X
9		CBNZ	X		X
10		**Test bit & branch (immediate)**			
11		TBZ			
12		TBNZ			
13		**Conditional branch (immediate)**			
14		B_cond			
15		**Exception generation**			
16	//	SVC			
17	//	HVC			
18	//	SMC			
19		BRK			
20	//	HLT			
21	//	DCPS1			
22	//	DCPS2			
23	//	DCPS3			
24	//	**System**			
25	//	MSR	imm	imm	
26	//	HINT			
27	//	CLREX			
28	//	DSB			
29	//	DMB			
30	//	ISB			
31	//	SYS			
32	//	MSR			
33	//	SYSL			
34	//	MRS			
35		**Unconditional branch (register)**			
36		BR			
37		BLR			

1	in_use	Opcode	Extended Name	Specific	variant
38		RET			
39	//	ERET			
40	//	DRPS			
41	//	**Unconditional branch (immediate)**			
42	//	B			
43	//	BL			
44		**Loads and stores**			
45		**Load/store exclusive**			
46		STXRB			
47		STLXRB			
48		LDXRB			
49		LDAXRB			
50		STLRB			
51		LDARB			
52		STXRH			
53		STLXRH			
54		LDXRH			
55		LDAXRH			
56		STLRH			
57		LDARH			
58		STXR	W		W
59		STLXR	W		W
60		STXP	W		W
61		STLXP	W		W
62		LDXR	W		W
63		LDAXR	W		W
64		LDXP	W		W
65		LDAXP	W		W
66		STLR	W		W
67		LDAR	W		W
68		STXR	X		X
69		STLXR	X		X
70		STXP	X		X
71		STLXP	X		X
72		LDXR	X		X
73		LDAXR	X		X
74		LDXP	X		X

1	in_use	Opcode	Extended Name	Specific	variant
75		LDAXP	X		X
76		STLR	X		X
77		LDAR	X		X
78		**Load register (literal)**			
79		LDR	W		W
80		LDR	S		S
81		LDR	X		X
82		LDR	D		D
83		LDRSW			
84		LDR	Q		Q
85		PRFM			
86		**Load/store no-allocate pair (offset)**			
87		STNP	W		W
88		LDNP	W		W
89		STNP	S		S
90		LDNP	S		S
91		STNP	D		D
92		LDNP	D		D
93		STNP	X		X
94		LDNP	X		X
95		STNP	Q		Q
96		LDNP	Q		Q
97		**Load/store register pair (post-indexed)**			
98		STP	post_W	post	W
99		LDP	post_W	post	W
100		STP	post_S	post	S
101		LDP	post_S	post	S
102		LDPSW	post	post	
103		STP	post_D	post	D
104		LDP	post_D	post	D
105		STP	post_X	post	X
106		LDP	post_X	post	X
107		STP	post_Q	post	Q
108		LDP	post_Q	post	Q
109		**Load/store register pair (offset)**			

1	in_use	Opcode	Extended Name	Specific	variant
110		STP	off_W	off	W
111		LDP	off_W	off	W
112		STP	off_S	off	S
113		LDP	off_S	off	S
114		LDPSW	off	off	
115		STP	off_D	off	D
116		LDP	off_D	off	D
117		STP	off_X	off	X
118		LDP	off_X	off	X
119		STP	off_Q	off	Q
120		LDP	off_Q	off	Q
121		**Load/store register pair (pre-indexed)**			
122		STP	pre_W	pre	W
123		LDP	pre_W	pre	W
124		STP	pre_S	pre	S
125		LDP	pre_S	pre	S
126		LDPSW	pre	pre	
127		STP	pre_D	pre	D
128		LDP	pre_D	pre	D
129		STP	pre_X	pre	X
130		LDP	pre_X	pre	X
131		STP	pre_Q	pre	Q
132		LDP	pre_Q	pre	Q
133		**Load/store register (unscaled immediate)**			
134		STURB			
135		LDURB			
136		LDURSB	X		X
137		LDURSB	W		W
138		STUR	B		B
139		LDUR	B		B
140		STUR	Q		Q
141		LDUR	Q		Q
142		STURH			
143		LDURH			
144		LDURSH	X		X
145		LDURSH	W		W
146		STUR	H		H
147		LDUR	H		H

1	in_use	Opcode	Extended Name	Specific	variant
148		STUR	W		W
149		LDUR	W		W
150		LDURSW			
151		STUR	S		S
152		LDUR	S		S
153		STUR	X		X
154		LDUR	X		X
155		PRFUM			
156		STUR	D		D
157		LDUR	D		D
158		**Load/store register (immediate post-indexe**			
159		STRB	post	post	
160		LDRB	post	post	
161		LDRSB	post_X	post	X
162		LDRSB	post_W	post	W
163		STR	post_B	post	B
164		LDR	post_B	post	B
165		STR	post_Q	post	Q
166		LDR	post_Q	post	Q
167		STRH	post	post	
168		LDRH	post	post	
169		LDRSH	post_X	post	X
170		LDRSH	post_W	post	W
171		STR	post_H	post	H
172		LDR	post_H	post	H
173		STR	post_W	post	W
174		LDR	post_W	post	W
175		LDRSW	post	post	
176		STR	post_S	post	S
177		LDR	post_S	post	S
178		STR	post_X	post	X
179		LDR	post_X	post	X
180		STR	post_D	post	D
181		LDR	post_D	post	D
182		**Load/store register (unprivileged)**			
183		STTRB			
184		LDTRB			
185		LDTRSB	X		X

1	in_use	Opcode	Extended Name	Specific	variant
186		LDTRSB	W		W
187		STTRH			
188		LDTRH			
189		LDTRSH	X		X
190		LDTRSH	W		W
191		STTR	W		W
192		LDTR	W		W
193		LDTRSW			
194		STTR	X		X
195		LDTR	X		X
196		**Load/store register (immediate pre-indexed**			
197		STRB	pre	pre	
198		LDRB	pre	pre	
199		LDRSB	pre_X	pre	X
200		LDRSB	pre_W	pre	W
201		STR	pre_B	pre	B
202		LDR	pre_B	pre	B
203		STR	pre_Q	pre	Q
204		LDR	pre_Q	pre	Q
205		STRH	pre	pre	
206		LDRH	pre	pre	
207		LDRSH	pre_X	pre	X
208		LDRSH	pre_W	pre	W
209		STR	pre_H	pre	H
210		LDR	pre_H	pre	H
211		STR	pre_W	pre	W
212		LDR	pre_W	pre	W
213		LDRSW	pre	pre	
214		STR	pre_S	pre	S
215		LDR	pre_S	pre	S
216		STR	pre_X	pre	X
217		LDR	pre_X	pre	X
218		STR	pre_D	pre	D
219		LDR	pre_D	pre	D
220		**Load/store register (register offset)**			
221		STRB	off	off	
222		LDRB	off	off	
223		LDRSB	off_X	off	X

1	in_use	Opcode	Extended Name	Specific	variant
224		LDRSB	off_W	off	W
225		STR	off_B	off	B
226		LDR	off_B	off	B
227		STR	off_Q	off	Q
228		LDR	off_Q	off	Q
229		STRH	off	off	
230		LDRH	off	off	
231		LDRSH	off_X	off	X
232		LDRSH	off_W	off	W
233		STR	off_H	off	H
234		LDR	off_H	off	H
235		STR	off_W	off	W
236		LDR	off_W	off	W
237		LDRSW	off	off	
238		STR	off_S	off	S
239		LDR	off_S	off	S
240		STR	off_D	off	D
241		LDR	off_D	off	D
243		STR	off_D	off	D
244		LDR	off_D	off	D
242		PRFM	off	off	
245		**Load/store register (unsigned immediate)**			
246		STRB	imm	imm	
247		LDRB	imm	imm	
248		LDRSB	imm_X	imm	X
249		LDRSB	imm_W	imm	W
250		STR	imm_B	imm	B
251		LDR	imm_B	imm	B
252		STR	imm_Q	imm	Q
253		LDR	imm_Q	imm	Q
254		STRH	imm	imm	
255		LDRH	imm	imm	
256		LDRSH	imm_X	imm	X
257		LDRSH	imm_W	imm	W
258		STR	imm_H	imm	H
259		LDR	imm_H	imm	H
260		STR	imm_W	imm	W
261		LDR	imm_W	imm	W

1	in_use	Opcode	Extended Name	Specific	variant
262		LDRSW	imm	imm	
263		STR	imm_S	imm	S
264		LDR	imm_S	imm	S
265		STR	imm_X	imm	X
266		LDR	imm_X	imm	X
268		STR	imm_D	imm	D
269		LDR	imm_D	imm	D
267		PRFM	imm	imm	
270	**Data processing – Immediate**				
271		**PC-rel. addressing**			
272		ADR			
273		ADRP			
274		**Add/subtract (immediate)**			
275		ADD	imm_W	imm	W
276		ADDS	imm_W	imm	W
277		SUB	imm_W	imm	W
278		SUBS	imm_W	imm	W
279		ADD	imm_X	imm	X
280		ADDS	imm_X	imm	X
281		SUB	imm_X	imm	X
282		SUBS	imm_X	imm	X
283		**Logical (immediate)**	imm	imm	
284		AND	imm_W	imm	W
285		ORR	imm_W	imm	W
286		EOR	imm_W	imm	W
287		ANDS	imm_W	imm	W
288		AND	imm_X	imm	X
289		ORR	imm_X	imm	X
290		EOR	imm_X	imm	X
291		ANDS	imm_X	imm	X
292		**Move wide (immediate)**			
293		MOVN	W		W
294		MOVZ	W		W
295		MOVK	W		W
296		MOVN	X		X
297		MOVZ	X		X
298		MOVK	X		X
299		**Bitfield**			

1	in_use	Opcode	Extended Name	Specific	variant
300		SBFM	W		W
301		BFM	W		W
302		UBFM	W		W
303		SBFM	X		X
304		BFM	X		X
305		UBFM	X		X
306		**Extract**			
307		EXTR	W		W
308		EXTR	X		X
309		**Data Processing – register**			
310		**Logical (shifted register)**			
311		AND	W		W
312		BIC	W		W
313		ORR	W		W
314		ORN	W		W
315		EOR	W		W
316		EON	W		W
317		ANDS	W		W
318		BICS	W		W
319		AND	X		X
320		BIC	X		X
321		ORR	X		X
322		ORN	X		X
323		EOR	X		X
324		EON	X		X
325		ANDS	X		X
326		BICS	X		X
327		**Add/subtract (shifted register)**			
328		ADD	W		W
329		ADDS	W		W
330		SUB	W		W
331		SUBS	W		W
332		ADD	X		X
333		ADDS	X		X
334		SUB	X		X
335		SUBS	X		X
336		**Add/subtract (extended register)**			
337		ADD	ext_W	ext	W

1	in_use	Opcode	Extended Name	Specific	variant
338		ADDS	ext_W	ext	W
339		SUB	ext_W	ext	W
340		SUBS	ext_W	ext	W
341		ADD	ext_X	ext	X
342		ADDS	ext_X	ext	X
343		SUB	ext_X	ext	X
344		SUBS	ext_X	ext	X
345		**Add/subtract (with carry)**			
346		ADC	W		W
347		ADCS	W		W
348		SBC	W		W
349		SBCS	W		W
350		ADC	X		X
351		ADCS	X		X
352		SBC	X		X
353		SBCS	X		X
354		**Conditional compare (register)**			
355		CCMN	W		W
356		CCMN	X		X
357		CCMP	W		W
358		CCMP	X		X
359		**Conditional compare (immediate)**			
360		CCMN	imm_W	imm	W
361		CCMN	imm_X	imm	X
362		CCMP	imm_W	imm	W
363		CCMP	imm_X	imm	X
364		**Conditional select**			
365		CSEL	W		W
366		CSINC	W		W
367		CSINV	W		W
368		CSNEG	W		W
369		CSEL	X		X
370		CSINC	X		X
371		CSINV	X		X
372		CSNEG	X		X
373		**Data-processing (3 source)**			
374		MADD	W		W
375		MADD	X		X

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

1	in_use	Opcode	Extended Name	Specific	variant
414		REV16	W		W
415		REV16	X		X
416		REV32			
417	//	**Data Processing – SIMD and floating p**			
418	//	**Floating-point<->fixed-point conversions**			
419	//	SCVTF	scalar_fixed_point_32_bit_to_single_pre(	scalar_fixed_point	32_bit_to_sin(
420	//	UCVTF	scalar_fixed_point_32_bit_to_single_pre(	scalar_fixed_point	32_bit_to_sin(
421	//	FCVTZS	scalar_fixed_point_Single_precision_to_	scalar_fixed_point	Single_precisi
422	//	FCVTZU	scalar_fixed_point_Single_precision_to_	scalar_fixed_point	Single_precisi
423	//	SCVTF	scalar_fixed_point_32_bit_to_double_pre	scalar_fixed_point	32_bit_to_dou
424	//	UCVTF	scalar_fixed_point_32_bit_to_double_pre	scalar_fixed_point	32_bit_to_dou
425	//	FCVTZS	scalar_fixed_point_Double_precision_to_	scalar_fixed_point	Double_precis
426	//	FCVTZU	scalar_fixed_point_Double_precision_to_	scalar_fixed_point	Double_precis
427	//	SCVTF	scalar_fixed_point_64_bit_to_single_pre(	scalar_fixed_point	64_bit_to_sin(
428	//	UCVTF	scalar_fixed_point_64_bit_to_single_pre(	scalar_fixed_point	64_bit_to_sin(
429	//	FCVTZS	scalar_fixed_point_Single_precision_to_(	scalar_fixed_point	Single_precisi
430	//	FCVTZU	scalar_fixed_point_Single_precision_to_(	scalar_fixed_point	Single_precisi
431	//	SCVTF	scalar_fixed_point_64_bit_to_double_pre	scalar_fixed_point	64_bit_to_dou
432	//	UCVTF	scalar_fixed_point_64_bit_to_double_pre	scalar_fixed_point	64_bit_to_dou
433	//	FCVTZS	scalar_fixed_point_Double_precision_to_	scalar_fixed_point	Double_precis
434	//	FCVTZU	scalar_fixed_point_Double_precision_to_	scalar_fixed_point	Double_precis
435	//	**Floating-point conditional compare**			
436	//	FCCMP	Single_precision		Single_precisi
437	//	FCCMPE	Single_precision		Single_precisi
438	//	FCCMP	Double_precision		Double_precis
439	//	FCCMPE	Double_precision		Double_precis
440	//	**Floating-point data-processing (2 source)**			
441	//	FMUL	scalar_Single_precision	scalar	Single_precisi
442	//	FDIV	scalar_Single_precision	scalar	Single_precisi
443	//	FADD	scalar_Single_precision	scalar	Single_precisi
444	//	FSUB	scalar_Single_precision	scalar	Single_precisi
445	//	FMAX	scalar_Single_precision	scalar	Single_precisi
446	//	FMIN	scalar_Single_precision	scalar	Single_precisi
447	//	FMAXNM	scalar_Single_precision	scalar	Single_precisi

1	in_use	Opcode	Extended Name	Specific	variant
448	//	FMINNM	scalar_Single_precision	scalar	Single_precisi
449	//	FNMUL	Single_precision		Single_precisi
450	//	FMUL	scalar_Double_precision	scalar	Double_precis
451	//	FDIV	scalar_Double_precision	scalar	Double_precis
452	//	FADD	scalar_Double_precision	scalar	Double_precis
453	//	FSUB	scalar_Double_precision	scalar	Double_precis
454	//	FMAX	scalar_Double_precision	scalar	Double_precis
455	//	FMIN	scalar_Double_precision	scalar	Double_precis
456	//	FMAXNM	scalar_Double_precision	scalar	Double_precis
457	//	FMINNM	scalar_Double_precision	scalar	Double_precis
458	//	FNMUL	Double_precision		Double_precis
459	//	**Floating-point conditional select**			
460	//	FCSEL	Single_precision		Single_precisi
461	//	FCSEL	Double_precision		Double_precis
462	//	**Floating-point immediate**			
463	//	FMOV	scalar_immediate_Single_precision	scalar_immediate	Single_precisi
464	//	FMOV	scalar_immediate_Double_precision	scalar_immediate	Double_precis
465	//	**Floating-point compare**			
466	//	FCMP	Single_precision		Single_precisi
467	//	FCMP	Single_precision_zero		Single_precisi
468	//	FCMPE	Single_precision		Single_precisi
469	//	FCMPE	Single_precision_zero		Single_precisi
470	//	FCMP	Double_precision		Double_precis
471	//	FCMP	Double_precision_zero		Double_precis
472	//	FCMPE	Double_precision		Double_precis
473	//	FCMPE	Double_precision_zero		Double_precis
474	//	**Floating-point data-processing (1 source)**			
475	//	FMOV	register_Single_precision	register	Single_precisi
476	//	FABS	scalar_Single_precision	scalar	Single_precisi
477	//	FNEG	scalar_Single_precision	scalar	Single_precisi
478	//	FSQRT	scalar_Single_precision	scalar	Single_precisi
479	//	FCVT	Single_precision_to_double_precision		Single_precisi
480	//	FCVT	Single_precision_to_half_precision		Single_precisi
481	//	FRINTN	scalar_Single_precision	scalar	Single_precisi

1	in_use	Opcode	Extended Name	Specific	variant
482	//	FRINTP	scalar_Single_precision	scalar	Single_precisi
483	//	FRINTM	scalar_Single_precision	scalar	Single_precisi
484	//	FRINTZ	scalar_Single_precision	scalar	Single_precisi
485	//	FRINTA	scalar_Single_precision	scalar	Single_precisi
486	//	FRINTX	scalar_Single_precision	scalar	Single_precisi
487	//	FRINTI	scalar_Single_precision	scalar	Single_precisi
488	//	FMOV	register_Double_precision	register	Double_precis
489	//	FABS	scalar_Double_precision	scalar	Double_precis
490	//	FNEG	scalar_Double_precision	scalar	Double_precis
491	//	FSQRT	scalar_Double_precision	scalar	Double_precis
492	//	FCVT	Double_precision_to_single_precision		Double_precis
493	//	FCVT	Double_precision_to_half_precision		Double_precis
494	//	FRINTN	scalar_Double_precision	scalar	Double_precis
495	//	FRINTP	scalar_Double_precision	scalar	Double_precis
496	//	FRINTM	scalar_Double_precision	scalar	Double_precis
497	//	FRINTZ	scalar_Double_precision	scalar	Double_precis
498	//	FRINTA	scalar_Double_precision	scalar	Double_precis
499	//	FRINTX	scalar_Double_precision	scalar	Double_precis
500	//	FRINTI	scalar_Double_precision	scalar	Double_precis
501	//	FCVT	Half_precision_to_single_precision		Half_precisior
502	//	FCVT	Half_precision_to_double_precision		Half_precisior
503	//	**Floating-point<->integer conversions**			
504	//	FCVTNS	scalar_Single_precision_to_32_bit	scalar	Single_precisi
505	//	FCVTNU	scalar_Single_precision_to_32_bit	scalar	Single_precisi
506	//	SCVTF	scalar_integer_32_bit_to_single_precisio	scalar_integer	32_bit_to_sing
507	//	UCVTF	scalar_integer_32_bit_to_single_precisio	scalar_integer	32_bit_to_sing
508	//	FCVTAS	scalar_Single_precision_to_32_bit	scalar	Single_precisi
509	//	FCVTAU	scalar_Single_precision_to_32_bit	scalar	Single_precisi
510	//	FMOV	general_Single_precision_to_32_bit	general	Single_precisi
511	//	FMOV	general_32_bit_to_single_precision	general	32_bit_to_sing
512	//	FCVTPS	scalar_Single_precision_to_32_bit	scalar	Single_precisi
513	//	FCVTPU	scalar_Single_precision_to_32_bit	scalar	Single_precisi
514	//	FCVTMS	scalar_Single_precision_to_32_bit	scalar	Single_precisi
515	//	FCVTMU	scalar_Single_precision_to_32_bit	scalar	Single_precisi

1	in_use	Opcode	Extended Name	Specific	variant
516	//	FCVTZS	scalar_integer_Single_precision_to_32_b	scalar_integer	Single_precisi
517	//	FCVTZU	scalar_integer_Single_precision_to_32_b	scalar_integer	Single_precisi
518	//	FCVTNS	scalar_Double_precision_to_32_bit	scalar	Double_precis
519	//	FCVTNU	scalar_Double_precision_to_32_bit	scalar	Double_precis
520	//	SCVTF	scalar_integer_32_bit_to_double_precisi	scalar_integer	32_bit_to_dou
521	//	UCVTF	scalar_integer_32_bit_to_double_precisi	scalar_integer	32_bit_to_dou
522	//	FCVTAS	scalar_Double_precision_to_32_bit	scalar	Double_precis
523	//	FCVTAU	scalar_Double_precision_to_32_bit	scalar	Double_precis
524	//	FCVTPS	scalar_Double_precision_to_32_bit	scalar	Double_precis
525	//	FCVTPU	scalar_Double_precision_to_32_bit	scalar	Double_precis
526	//	FCVTMS	scalar_Double_precision_to_32_bit	scalar	Double_precis
527	//	FCVTMU	scalar_Double_precision_to_32_bit	scalar	Double_precis
528	//	FCVTZS	scalar_integer_Double_precision_to_32_	scalar_integer	Double_precis
529	//	FCVTZU	scalar_integer_Double_precision_to_32_	scalar_integer	Double_precis
530	//	FCVTNS	scalar_Single_precision_to_64_bit	scalar	Single_precisi
531	//	FCVTNU	scalar_Single_precision_to_64_bit	scalar	Single_precisi
532	//	SCVTF	scalar_integer_64_bit_to_single_precisio	scalar_integer	64_bit_to_sing
533	//	UCVTF	scalar_integer_64_bit_to_single_precisio	scalar_integer	64_bit_to_sing
534	//	FCVTAS	scalar_Single_precision_to_64_bit	scalar	Single_precisi
535	//	FCVTAU	scalar_Single_precision_to_64_bit	scalar	Single_precisi
536	//	FCVTPS	scalar_Single_precision_to_64_bit	scalar	Single_precisi
537	//	FCVTPU	scalar_Single_precision_to_64_bit	scalar	Single_precisi
538	//	FCVTMS	scalar_Single_precision_to_64_bit	scalar	Single_precisi
539	//	FCVTMU	scalar_Single_precision_to_64_bit	scalar	Single_precisi
540	//	FCVTZS	scalar_integer_Single_precision_to_64_b	scalar_integer	Single_precisi
541	//	FCVTZU	scalar_integer_Single_precision_to_64_b	scalar_integer	Single_precisi
542	//	FCVTNS	scalar_Double_precision_to_64_bit	scalar	Double_precis
543	//	FCVTNU	scalar_Double_precision_to_64_bit	scalar	Double_precis
544	//	SCVTF	scalar_integer_64_bit_to_double_precisi	scalar_integer	64_bit_to_dou
545	//	UCVTF	scalar_integer_64_bit_to_double_precisi	scalar_integer	64_bit_to_dou
546	//	FCVTAS	scalar_Double_precision_to_64_bit	scalar	Double_precis
547	//	FCVTAU	scalar_Double_precision_to_64_bit	scalar	Double_precis
548	//	FMOV	general_Double_precision_to_64_bit	general	Double_precis
549	//	FMOV	general_64_bit_to_double_precision	general	64_bit_to_dou

1	in_use	Opcode	Extended Name	Specific	variant
550	//	FCVTPS	scalar_Double_precision_to_64_bit	scalar	Double_precis
551	//	FCVTPU	scalar_Double_precision_to_64_bit	scalar	Double_precis
552	//	FCVTMS	scalar_Double_precision_to_64_bit	scalar	Double_precis
553	//	FCVTMU	scalar_Double_precision_to_64_bit	scalar	Double_precis
554	//	FCVTZS	scalar_integer_Double_precision_to_64_	scalar_integer	Double_precis
555	//	FCVTZU	scalar_integer_Double_precision_to_64_	scalar_integer	Double_precis
556	//	FMOV	general_Top_half_of_128_bit_to_64_bit	general	Top_half_of_1
557	//	FMOV	general_64_bit_to_top_half_of_128_bit	general	64_bit_to_top
558	//	**Floating-point data-processing (3 source)**			
559	//	FMADD	Single_precision		Single_precisi
560	//	FMSUB	Single_precision		Single_precisi
561	//	FNMADD	Single_precision		Single_precisi
562	//	FNMSUB	Single_precision		Single_precisi
563	//	FMADD	Double_precision		Double_precis
564	//	FMSUB	Double_precision		Double_precis
565	//	FNMADD	Double_precision		Double_precis
566	//	FNMSUB	Double_precision		Double_precis
567	//	**AdvSIMD scalar three same**			
568	//	SQADD	Scalar		Scalar
569	//	SQSUB	Scalar		Scalar
570	//	CMGT	register_Scalar	register	Scalar
571	//	CMGE	register_Scalar	register	Scalar
572	//	SSHL	Scalar		Scalar
573	//	SQSHL	register_Scalar	register	Scalar
574	//	SRSHL	Scalar		Scalar
575	//	SQRSHL	Scalar		Scalar
576	//	ADD	vector_Scalar	vector	Scalar
577	//	CMTST	Scalar		Scalar
578	//	SQDMULH	vector_Scalar	vector	Scalar
579	//	FMULX	Scalar		Scalar
580	//	FCMEQ	register_Scalar	register	Scalar
581	//	FRECPS	Scalar		Scalar
582	//	FRSQRTS	Scalar		Scalar
583	//	UQADD	Scalar		Scalar

1	in_use	Opcode	Extended Name	Specific	variant
584	//	UQSUB	Scalar		Scalar
585	//	CMHI	register_Scalar	register	Scalar
586	//	CMHS	register_Scalar	register	Scalar
587	//	USHL	Scalar		Scalar
588	//	UQSHL	register_Scalar	register	Scalar
589	//	URSHL	Scalar		Scalar
590	//	UQRSHL	Scalar		Scalar
591	//	SUB	vector_Scalar	vector	Scalar
592	//	CMEQ	register_Scalar	register	Scalar
593	//	SQRDMULH	vector_Scalar	vector	Scalar
594	//	FCMGE	register_Scalar	register	Scalar
595	//	FACGE	Scalar		Scalar
596	//	FABD	Scalar		Scalar
597	//	FCMGT	register_Scalar	register	Scalar
598	//	FACGT	Scalar		Scalar
599	//	**AdvSIMD scalar three different**			
600	//	SQDMLAL	vector_Scalar	vector	Scalar
601	//	SQDMLAL2	vector_Scalar	vector	Scalar
602	//	SQDMLSL	vector_Scalar	vector	Scalar
603	//	SQDMLSL2	vector_Scalar	vector	Scalar
604	//	SQDMULL	vector_Scalar	vector	Scalar
605	//	SQDMULL2	vector_Scalar	vector	Scalar
606	//	**AdvSIMD scalar two-reg misc**			
607	//	SUQADD	Scalar		Scalar
608	//	SQABS	Scalar		Scalar
609	//	CMGT	zero_Scalar	zero	Scalar
610	//	CMEQ	zero_Scalar	zero	Scalar
611	//	CMLT	zero_Scalar	zero	Scalar
612	//	ABS	Scalar		Scalar
613	//	SQXTN	Scalar		Scalar
614	//	SQXTN2	Scalar		Scalar
615	//	FCVTNS	vector_Scalar	vector	Scalar
616	//	FCVTMS	vector_Scalar	vector	Scalar
617	//	FCVTAS	vector_Scalar	vector	Scalar

1	in_use	Opcode	Extended Name	Specific	variant
618	//	SCVTF	vector_integer_Scalar	vector_integer	Scalar
619	//	FCMGT	zero_Scalar	zero	Scalar
620	//	FCMEQ	zero_Scalar	zero	Scalar
621	//	FCMLT	zero_Scalar	zero	Scalar
622	//	FCVTPS	vector_Scalar	vector	Scalar
623	//	FCVTZS	vector_integer_Scalar	vector_integer	Scalar
624	//	FRECPE	Scalar		Scalar
625	//	FRECPX			
626	//	USQADD	Scalar		Scalar
627	//	SQNEG	Scalar		Scalar
628	//	CMGE	zero_Scalar	zero	Scalar
629	//	CMLE	zero_Scalar	zero	Scalar
630	//	NEG	vector_Scalar	vector	Scalar
631	//	SQXTUN	Scalar		Scalar
632	//	SQXTUN2	Scalar		Scalar
633	//	UQXTN	Scalar		Scalar
634	//	UQXTN2	Scalar		Scalar
635	//	FCVTXN	Scalar		Scalar
636	//	FCVTXN2	Scalar		Scalar
637	//	FCVTNU	vector_Scalar	vector	Scalar
638	//	FCVTMU	vector_Scalar	vector	Scalar
639	//	FCVTAU	vector_Scalar	vector	Scalar
640	//	UCVTF	vector_integer_Scalar	vector_integer	Scalar
641	//	FCMGE	zero_Scalar	zero	Scalar
642	//	FCMLE	zero_Scalar	zero	Scalar
643	//	FCVTPU	vector_Scalar	vector	Scalar
644	//	FCVTZU	vector_integer_Scalar	vector_integer	Scalar
645	//	FRSQRTE	Scalar		Scalar
646	//	**AdvSIMD scalar pairwise**			
647	//	ADDP	scalar	scalar	
648	//	FMAXNMP	scalar	scalar	
649	//	FADDP	scalar	scalar	
650	//	FMAXP	scalar	scalar	
651	//	FMINNMP	scalar	scalar	

1	in_use	Opcode	Extended Name	Specific	variant
652	//	FMINP	scalar	scalar	
653	//	**AdvSIMD scalar copy**			
654	//	DUP	element_Scalar	element	Scalar
655	//	**AdvSIMD scalar x indexed element**			
656	//	SQDMLAL	by_element_Scalar	by_element	Scalar
657	//	SQDMLAL2	by_element_Scalar	by_element	Scalar
658	//	SQDMLSL	by_element_Scalar	by_element	Scalar
659	//	SQDMLSL2	by_element_Scalar	by_element	Scalar
660	//	SQDMULL	by_element_Scalar	by_element	Scalar
661	//	SQDMULL2	by_element_Scalar	by_element	Scalar
662	//	SQDMULH	by_element_Scalar	by_element	Scalar
663	//	SQRDMULH	by_element_Scalar	by_element	Scalar
664	//	FMLA	by_element_Scalar	by_element	Scalar
665	//	FMLS	by_element_Scalar	by_element	Scalar
666	//	FMUL	by_element_Scalar	by_element	Scalar
667	//	FMULX	by_element_Scalar	by_element	Scalar
668	//	**AdvSIMD scalar shift by immediate**			
669	//	SSHR	Scalar		Scalar
670	//	SSRA	Scalar		Scalar
671	//	SRSHR	Scalar		Scalar
672	//	SRSRA	Scalar		Scalar
673	//	SHL	Scalar		Scalar
674	//	SQSHL	immediate_Scalar	immediate	Scalar
675	//	SQSHRN	Scalar		Scalar
676	//	SQSHRN2	Scalar		Scalar
677	//	SQRSHRN	Scalar		Scalar
678	//	SQRSHRN2	Scalar		Scalar
679	//	SCVTF	vector_fixed_point_Scalar	vector_fixed_point	Scalar
680	//	FCVTZS	vector_fixed_point_Scalar	vector_fixed_point	Scalar
681	//	USHR	Scalar		Scalar
682	//	USRA	Scalar		Scalar
683	//	URSHR	Scalar		Scalar
684	//	URSRA	Scalar		Scalar
685	//	SRI	Scalar		Scalar

1	in_use	Opcode	Extended Name	Specific	variant
686	//	SLI	Scalar		Scalar
687	//	SQSHLU	Scalar		Scalar
688	//	UQSHL	immediate_Scalar	immediate	Scalar
689	//	SQSHRUN	Scalar		Scalar
690	//	SQSHRUN2	Scalar		Scalar
691	//	SQRSHRUN	Scalar		Scalar
692	//	SQRSHRUN2	Scalar		Scalar
693	//	UQSHRN	Scalar		Scalar
694	//	UQRSHRN	Scalar		Scalar
695	//	UQRSHRN2	Scalar		Scalar
696	//	UCVTF	vector_fixed_point_Scalar	vector_fixed_point	Scalar
697	//	FCVTZU	vector_fixed_point_Scalar	vector_fixed_point	Scalar
698	//	**Crypto three-reg SHA**			
699	//	SHA1C			
700	//	SHA1P			
701	//	SHA1M			
702	//	SHA1SU0			
703	//	SHA256H			
704	//	SHA256H2			
705	//	SHA256SU1			
706	//	**Crypto two-reg SHA**			
707	//	SHA1H			
708	//	SHA1SU1			
709	//	SHA256SU0			
710	//	**Crypto AES**			
711	//	AESE			
712	//	AESD			
713	//	AESMC			
714	//	AESIMC			
715	//	**AdvSIMD three same**			
716	//	SHADD			
717	//	SQADD	Vector		Vector
718	//	SRHADD			
719	//	SHSUB			

1	in_use	Opcode	Extended Name	Specific	variant
720	//	SQSUB	Vector		Vector
721	//	CMGT	register_Vector	register	Vector
722	//	CMGE	register_Vector	register	Vector
723	//	SSHL Vector			
724	//	SQSHL	register_Vector	register	Vector
725	//	SRSHL	Vector		Vector
726	//	SQRSHL	Vector		Vector
727	//	SMAX			
728	//	SMIN			
729	//	SABD			
730	//	SABA			
731	//	ADD	vector_Vector	vector	Vector
732	//	CMTST	Vector		Vector
733	//	MLA	vector	vector	
734	//	MUL	vector	vector	
735	//	SMAXP			
736	//	SMINP			
737	//	SQDMULH	vector_Vector	vector	Vector
738	//	ADDP	vector	vector	
739	//	FMAXNM	vector	vector	
740	//	FMLA	vector	vector	
741	//	FADD	vector	vector	
742	//	FMULX	Vector		Vector
743	//	FCMEQ	register_Vector	register	Vector
744	//	FMAX	vector	vector	
745	//	FRECPS	Vector		Vector
746	//	AND	vector	vector	
747	//	BIC	vector_register	vector_register	
748	//	FMINNM	vector	vector	
749	//	FMLS	vector	vector	
750	//	FSUB	vector	vector	
751	//	FMIN	vector	vector	
752	//	FRSQRTS	Vector		Vector
753	//	ORR	vector_register	vector_register	

1	in_use	Opcode	Extended Name	Specific	variant
754	//	ORN	vector	vector	
755	//	UHADD			
756	//	UQADD	Vector		Vector
757	//	URHADD			
758	//	UHSUB			
759	//		Vector		Vector
760	//	CMHI	register_Vector	register	Vector
761	//	CMHS	register_Vector	register	Vector
762	//	USHL	Vector		Vector
763	//	UQSHL	register_Vector	register	Vector
764	//	URSHL	Vector		Vector
765	//	UQRSHL	Vector		Vector
766	//	UMAX			
767	//	UMIN			
768	//	UABD			
769	//	UABA			
770	//	SUB	vector_Vector	vector	Vector
771	//	CMEQ	register_Vector	register	Vector
772	//	MLS	vector	vector	
773	//	PMUL			
774	//	UMAXP			
775	//	UMINP			
776	//	SQRDMULH	vector_Vector	vector	Vector
777	//	FMAXNMP	vector	vector	
778	//	FADDP	vector	vector	
779	//	FMUL	vector	vector	
780	//	FCMGE	register_Vector	register	Vector
781	//	FACGE	Vector		Vector
782	//	FMAXP	vector	vector	
783	//	FDIV	vector	vector	
784	//	EOR	vector	vector	
785	//	BSL			
786	//	FMINNMP	vector	vector	
787	//	FABD	Vector		Vector

1	in_use	Opcode	Extended Name	Specific	variant
788	//	FCMGT	register_Vector	register	Vector
789	//	FACGT	Vector		Vector
790	//	FMINP	vector	vector	
791	//	BIT			
792	//	BIF			
793	//	**AdvSIMD three different**			
794	//	SADDL			
795	//	SADDL2			
796	//	SADDW			
797	//	SADDW2			
798	//	SSUBL			
799	//	SSUBL2			
800	//	SSUBW			
801	//	SSUBW2			
802	//	ADDHN			
803	//	ADDHN2			
804	//	SABAL			
805	//	SABAL2			
806	//	SUBHN			
807	//	SUBHN2			
808	//	SABDL			
809	//	SABDL2			
810	//	SMLAL	vector	vector	
811	//	SMLAL2	vector	vector	
812	//	SQDMLAL	vector_Vector	vector	Vector
813	//	SQDMLAL2	vector_Vector	vector	Vector
814	//	SMLSL	vector	vector	
815	//	SMLSL2	vector	vector	
816	//	SQDMLSL	vector_Vector	vector	Vector
817	//	SQDMLSL2	vector_Vector	vector	Vector
818	//	SMULL	vector	vector	
819	//	SMULL2	vector	vector	
820	//	SQDMULL	vector_Vector	vector	Vector
821	//	SQDMULL2	vector_Vector	vector	Vector

1	in_use	Opcode	Extended Name	Specific	variant
822	//	PMULL			
823	//	PMULL2			
824	//	UADDL			
825	//	UADDL2			
826	//	UADDW			
827	//	UADDW2			
828	//	USUBL			
829	//	USUBL2			
830	//	USUBW			
831	//	USUBW2			
832	//	RADDHN			
833	//	RADDHN2			
834	//	UABAL			
835	//	UABAL2			
836	//	RSUBHN			
837	//	RSUBHN2			
838	//	UABDL			
839	//	UABDL2			
840	//	UMLAL	vector	vector	
841	//	UMLAL2	vector	vector	
842	//	UMLSL	vector	vector	
843	//	UMLSL2	vector	vector	
844	//	UMULL	vector	vector	
845	//	UMULL2	vector	vector	
846	//	**AdvSIMD two-reg misc**			
847	//	REV64			
848	//	REV16	vector	vector	
849	//	SADDLP			
850	//	SUQADD	Vector		Vector
851	//	CLS	vector	vector	
852	//	CNT			
853	//	SADALP			
854	//	SQABS	Vector		Vector
855	//	CMGT	zero_Vector	zero	Vector

1	in_use	Opcode	Extended Name	Specific	variant
856	//	CMEQ	zero_Vector	zero	Vector
857	//	CMLT	zero_Vector	zero	Vector
858	//	ABS	Vector		Vector
859	//	XTN			
860	//	XTN2			
861	//	SQXTN	Vector		Vector
862	//	SQXTN2	Vector		Vector
863	//	FCVTN			
864	//	FCVTN2			
865	//	FCVTL			
866	//	FCVTL2			
867	//	FRINTN	vector	vector	
868	//	FRINTM	vector	vector	
869	//	FCVTNS	vector_Vector	vector	Vector
870	//	FCVTMS	vector_Vector	vector	Vector
871	//	FCVTAS	vector_Vector	vector	Vector
872	//	SCVTF	vector_integer_Vector	vector_integer	Vector
873	//	FCMGT	zero_Vector	zero	Vector
874	//	FCMEQ	zero_Vector	zero	Vector
875	//	FCMLT	zero_Vector	zero	Vector
876	//	FABS	vector	vector	
877	//	FRINTP	vector	vector	
878	//	FRINTZ	vector	vector	
879	//	FCVTPS	vector_Vector	vector	Vector
880	//	FCVTZS	vector_integer_Vector	vector_integer	Vector
881	//	URECPE			
882	//	FRECPE	Vector		Vector
883	//	REV32	vector	vector	
884	//	UADDLP			
885	//	USQADD	Vector		Vector
886	//	CLZ	vector	vector	
887	//	UADALP			
888	//	SQNEG	Vector		Vector
889	//	CMGE	zero_Vector	zero	Vector

1	in_use	Opcode	Extended Name	Specific	variant
890	//	CMLE	zero_Vector	zero	Vector
891	//	NEG	vector_Vector	vector	Vector
892	//	SQXTUN	Vector		Vector
893	//	SQXTUN2	Vector		Vector
894	//	SHLL			
895	//	SHLL2			
896	//	UQXTN	Vector		Vector
897	//	UQXTN2	Vector		Vector
898	//	FCVTXN	Vector		Vector
899	//	FCVTXN2	Vector		Vector
900	//	FRINTA	vector	vector	
901	//	FRINTX	vector	vector	
902	//	FCVTNU	vector_Vector	vector	Vector
903	//	FCVTMU	vector_Vector	vector	Vector
904	//	FCVTAU	vector_Vector	vector	Vector
905	//	UCVTF	vector_integer_Vector	vector_integer	Vector
906	//	NOT			
907	//	RBIT	vector	vector	
908	//	FCMGE	zero_Vector	zero	Vector
909	//	FCMLE	zero_Vector	zero	Vector
910	//	FNEG	vector	vector	
911	//	FRINTI	vector	vector	
912	//	FCVTPU	vector_Vector	vector	Vector
913	//	FCVTZU	vector_integer_Vector	vector_integer	Vector
914	//	URSQRTE			
915	//	FRSQRTE	Vector		Vector
916	//	FSQRT	vector	vector	
917	//	**AdvSIMD across lanes**			
918	//	SADDLV			
919	//	SMAXV			
920	//	SMINV			
921	//	ADDV			
922	//	UADDLV			
923	//	UMAXV			

1	in_use	Opcode	Extended Name	Specific	variant
924	//	UMINV			
925	//	FMAXNMV			
926	//	FMAXV			
927	//	FMINNMV			
928	//	FMINV			
929	//	**AdvSIMD copy**			
930	//	DUP	element_Vector	element	Vector
931	//	DUP	general	general	
932	//	SMOV	32_bit		32_bit
933	//	UMOV	32_bit		32_bit
934	//	INS	general	general	
935	//	SMOV	64_bit		64_bit
936	//	UMOV	64_bit		64_bit
937	//	INS	element	element	
938	//	**AdvSIMD vector x indexed element**			
939	//	SMLAL	by_element	by_element	
940	//	SMLAL2	by_element	by_element	
941	//	SQDMLAL	by_element_Vector	by_element	Vector
942	//	SQDMLAL2	by_element_Vector	by_element	Vector
943	//	SMLSL	by_element	by_element	
944	//	SMLSL2	by_element	by_element	
945	//	SQDMLSL	by_element_Vector	by_element	Vector
946	//	SQDMLSL2	by_element_Vector	by_element	Vector
947	//	MUL	by_element	by_element	
948	//	SMULL	by_element	by_element	
949	//	SMULL2	by_element	by_element	
950	//	SQDMULL	by_element_Vector	by_element	Vector
951	//	SQDMULL2	by_element_Vector	by_element	Vector
952	//	SQDMULH	by_element_Vector	by_element	Vector
953	//	SQRDMULH	by_element_Vector	by_element	Vector
954	//	FMLA	by_element_Vector	by_element	Vector
955	//	FMLS	by_element_Vector	by_element	Vector
956	//	FMUL	by_element_Vector	by_element	Vector
957	//	MLA	by_element	by_element	

1	in_use	Opcode	Extended Name	Specific	variant
958	//	UMLAL	by_element	by_element	
959	//	UMLAL2	by_element	by_element	
960	//	MLS	by_element	by_element	
961	//	UMLSL	by_element	by_element	
962	//	UMLSL2	by_element	by_element	
963	//	UMULL	by_element	by_element	
964	//	UMULL2	by_element	by_element	
965	//	FMULX	by_element_Vector	by_element	Vector
966	//	**AdvSIMD modified immediate**			
967	//	MOVI	32_bit_shifted_immediate		32_bit_shifted
968	//	ORR	vector_immediate_32_bit	vector_immediate	32_bit
969	//	MOVI	16_bit_shifted_immediate		16_bit_shifted
970	//	ORR	vector_immediate_16_bit	vector_immediate	16_bit
971	//	MOVI	32_bit_shifting_ones		32_bit_shiftin
972	//	MOVI	8_bit		8_bit
973	//	FMOV	vector_immediate_Single_precision	vector_immediate	Single_precisi
974	//	MVNI	32_bit_shifted_immediate		32_bit_shifted
975	//	BIC	vector_immediate_32_bit	vector_immediate	32_bit
976	//	MVNI	16_bit_shifted_immediate		16_bit_shifted
977	//	BIC	vector_immediate_16_bit	vector_immediate	16_bit
978	//	MVNI	32_bit_shifting_ones		32_bit_shiftin
979	//	MOVI	64_bit_scalar		64_bit_scalar
980	//	MOVI	64_bit_vector		64_bit_vector
981	//	FMOV	vector_immediate_Double_precision	vector_immediate	Double_precis
982	//	**AdvSIMD shift by immediate**			
983	//	SSHR	Vector		Vector
984	//	SSRA	Vector		Vector
985	//	SRSHR	Vector		Vector
986	//	SRSRA	Vector		Vector
987	//	SHL	Vector		Vector
988	//	SQSHL	immediate_Vector	immediate	Vector
989	//	SHRN			
990	//	SHRN2			
991	//	RSHRN			

1	in_use	Opcode	Extended Name	Specific	variant
992	//	RSHRN2			
993	//	SQSHRN	Vector		Vector
994	//	SQSHRN2	Vector		Vector
995	//	SQRSHRN	Vector		Vector
996	//	SQRSHRN2	Vector		Vector
997	//	SSHLL			
998	//	SSHLL2			
999	//	SCVTF	vector_fixed_point_Vector	vector_fixed_point	Vector
1000	//	FCVTZS	vector_fixed_point_Vector	vector_fixed_point	Vector
1001	//	USHR	Vector		Vector
1002	//	USRA	Vector		Vector
1003	//	URSHR	Vector		Vector
1004	//	URSRA	Vector		Vector
1005	//	SRI	Vector		Vector
1006	//	SLI	Vector		Vector
1007	//	SQSHLU	Vector		Vector
1008	//	UQSHL	immediate_Vector	immediate	Vector
1009	//	SQSHRUN	Vector		Vector
1010	//	SQSHRUN2	Vector		Vector
1011	//	SQRSHRUN	Vector		Vector
1012	//	SQRSHRUN2	Vector		Vector
1013	//	UQSHRN	Vector		Vector
1014	//	UQRSHRN	Vector		Vector
1015	//	UQRSHRN2	Vector		Vector
1016	//	USHLL			
1017	//	USHLL2			
1018	//	UCVTF	vector_fixed_point_Vector	vector_fixed_point	Vector
1019	//	FCVTZU	vector_fixed_point_Vector	vector_fixed_point	Vector
1020	//	**AdvSIMD TBL/TBX**			
1021	//	TBL	Single_register_table		Single_registe
1022	//	TBX	Single_register_table		Single_registe
1023	//	TBL	Two_register_table		Two_register_
1024	//	TBX	Two_register_table		Two_register_
1025	//	TBL	Three_register_table		Three_registe

1	in_use	Opcode	Extended Name	Specific	variant
1026	//	TBX	Three_register_table		Three_registe
1027	//	TBL	Four_register_table		Four_register_
1028	//	TBX	Four_register_table		Four_register_
1029	//	**AdvSIMD ZIP/UZP/TRN**			
1030	//	UZP1			
1031	//	TRN1			
1032	//	ZIP1			
1033	//	UZP2			
1034	//	TRN2			
1035	//	ZIP2			
1036	//	**AdvSIMD EXT**			
1037	//	EXT			
1038	//	**Loads and stores**			
1039	//	**AdvSIMD load/store multiple structures**			
1040	//	ST4	multiple_structures_No_offset	multiple_structures	No_offset
1041	//	ST1	multiple_structures_Four_registers	multiple_structures	Four_registers
1042	//	ST3	multiple_structures_No_offset	multiple_structures	No_offset
1043	//	ST1	multiple_structures_Three_registers	multiple_structures	Three_registe
1044	//	ST1	multiple_structures_One_register	multiple_structures	One_register
1045	//	ST2	multiple_structures_No_offset	multiple_structures	No_offset
1046	//	ST1	multiple_structures_Two_registers	multiple_structures	Two_registers
1047	//	LD4	multiple_structures_No_offset	multiple_structures	No_offset
1048	//	LD1	multiple_structures_Four_registers	multiple_structures	Four_registers
1049	//	LD3	multiple_structures_No_offset	multiple_structures	No_offset
1050	//	LD1	multiple_structures_Three_registers	multiple_structures	Three_registe
1051	//	LD1	multiple_structures_One_register	multiple_structures	One_register
1052	//	LD2	multiple_structures_No_offset	multiple_structures	No_offset
1053	//	LD1	multiple_structures_Two_registers	multiple_structures	Two_registers
1054	//	**AdvSIMD load/store multiple structures (pc**			
1055	//	ST4	multiple_structures_Register_offset	multiple_structures	Register_offse
1056	//	ST1	multiple_structures_Four_registers_regis	multiple_structures	Four_registers
1057	//	ST3	multiple_structures_Register_offset	multiple_structures	Register_offse
1058	//	ST1	multiple_structures_Three_registers_regi	multiple_structures	Three_registe
1059	//	ST1	multiple_structures_One_register_registe	multiple_structures	One_register_

1	in_use	Opcode	Extended Name	Specific	variant
1060	//	ST2	multiple_structures_Register_offset	multiple_structures	Register_offse
1061	//	ST1	multiple_structures_Two_registers_regist	multiple_structures	Two_registers
1062	//	ST4	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1063	//	ST1	multiple_structures_Four_registers_imme	multiple_structures	Four_registers
1064	//	ST3	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1065	//	ST1	multiple_structures_Three_registers_imn	multiple_structures	Three_registe
1066	//	ST1	multiple_structures_One_register_immed	multiple_structures	One_register_
1067	//	ST2	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1068	//	ST1	multiple_structures_Two_registers_imme	multiple_structures	Two_registers
1069	//	LD4	multiple_structures_Register_offset	multiple_structures	Register_offse
1070	//	LD1	multiple_structures_Four_registers_regis	multiple_structures	Four_registers
1071	//	LD3	multiple_structures_Register_offset	multiple_structures	Register_offse
1072	//	LD1	multiple_structures_Three_registers_regi	multiple_structures	Three_registe
1073	//	LD1	multiple_structures_One_register_registe	multiple_structures	One_register_
1074	//	LD2	multiple_structures_Register_offset	multiple_structures	Register_offse
1075	//	LD1	multiple_structures_Two_registers_regist	multiple_structures	Two_registers
1076	//	LD4	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1077	//	LD1	multiple_structures_Four_registers_imme	multiple_structures	Four_registers
1078	//	LD3	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1079	//	LD1	multiple_structures_Three_registers_imn	multiple_structures	Three_registe
1080	//	LD1	multiple_structures_One_register_immed	multiple_structures	One_register_
1081	//	LD2	multiple_structures_Immediate_offset	multiple_structures	Immediate_of
1082	//	LD1	multiple_structures_Two_registers_imme	multiple_structures	Two_registers
1083	//	**AdvSIMD load/store single structure**			
1084	//	ST1	single_structure_8_bit	single_structure	8_bit
1085	//	ST3	single_structure_8_bit	single_structure	8_bit
1086	//	ST1	single_structure_16_bit	single_structure	16_bit
1087	//	ST3	single_structure_16_bit	single_structure	16_bit
1088	//	ST1	single_structure_32_bit	single_structure	32_bit
1089	//	ST1	single_structure_64_bit	single_structure	64_bit
1090	//	ST3	single_structure_32_bit	single_structure	32_bit
1091	//	ST3	single_structure_64_bit	single_structure	64_bit
1092	//	ST2	single_structure_8_bit	single_structure	8_bit
1093	//	ST4	single_structure_8_bit	single_structure	8_bit

1	in_use	Opcode	Extended Name	Specific	variant
1094	//	ST2	single_structure_16_bit	single_structure	16_bit
1095	//	ST4	single_structure_16_bit	single_structure	16_bit
1096	//	ST2	single_structure_32_bit	single_structure	32_bit
1097	//	ST2	single_structure_64_bit	single_structure	64_bit
1098	//	ST4	single_structure_32_bit	single_structure	32_bit
1099	//	ST4	single_structure_64_bit	single_structure	64_bit
1100	//	LD1	single_structure_8_bit	single_structure	8_bit
1101	//	LD3	single_structure_8_bit	single_structure	8_bit
1102	//	LD1	single_structure_16_bit	single_structure	16_bit
1103	//	LD3	single_structure_16_bit	single_structure	16_bit
1104	//	LD1	single_structure_32_bit	single_structure	32_bit
1105	//	LD1	single_structure_64_bit	single_structure	64_bit
1106	//	LD3	single_structure_32_bit	single_structure	32_bit
1107	//	LD3	single_structure_64_bit	single_structure	64_bit
1108	//	LD1R	No_offset		No_offset
1109	//	LD3R	No_offset		No_offset
1110	//	LD2	single_structure_8_bit	single_structure	8_bit
1111	//	LD4	single_structure_8_bit	single_structure	8_bit
1112	//	LD2	single_structure_16_bit	single_structure	16_bit
1113	//	LD4	single_structure_16_bit	single_structure	16_bit
1114	//	LD2	single_structure_32_bit	single_structure	32_bit
1115	//	LD2	single_structure_64_bit	single_structure	64_bit
1116	//	LD4	single_structure_32_bit	single_structure	32_bit
1117	//	LD4	single_structure_64_bit	single_structure	64_bit
1118	//	LD2R	No_offset		No_offset
1119	//	LD4R	No_offset		No_offset
1120	//	**AdvSIMD load/store single structure (post-**			
1121	//	ST1	single_structure_8_bit_register_offset	single_structure	8_bit_register
1122	//	ST3	single_structure_8_bit_register_offset	single_structure	8_bit_register
1123	//	ST1	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1124	//	ST3	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1125	//	ST1	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1126	//	ST1	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1127	//	ST3	single_structure_32_bit_register_offset	single_structure	32_bit_registe

1	in_use	Opcode	Extended Name	Specific	variant
1128	//	ST3	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1129	//	ST1	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1130	//	ST3	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1131	//	ST1	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1132	//	ST3	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1133	//	ST1	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1134	//	ST1	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1135	//	ST3	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1136	//	ST3	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1137	//	ST2	single_structure_8_bit_register_offset	single_structure	8_bit_register
1138	//	ST4	single_structure_8_bit_register_offset	single_structure	8_bit_register
1139	//	ST2	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1140	//	ST4	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1141	//	ST2	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1142	//	ST2	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1143	//	ST4	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1144	//	ST4	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1145	//	ST2	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1146	//	ST4	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1147	//	ST2	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1148	//	ST4	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1149	//	ST2	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1150	//	ST2	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1151	//	ST4	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1152	//	ST4	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1153	//	LD1	single_structure_8_bit_register_offset	single_structure	8_bit_register
1154	//	LD3	single_structure_8_bit_register_offset	single_structure	8_bit_register
1155	//	LD1	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1156	//	LD3	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1157	//	LD1	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1158	//	LD1	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1159	//	LD3	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1160	//	LD3	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1161	//	LD1R	Register_offset		Register_offse

1	in_use	Opcode	Extended Name	Specific	variant
1162	//	LD3R	Register_offset		Register_offse
1163	//	LD1	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1164	//	LD3	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1165	//	LD1	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1166	//	LD3	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1167	//	LD1	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1168	//	LD1	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1169	//	LD3	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1170	//	LD3	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1171	//	LD1R	Immediate_offset		Immediate_of
1172	//	LD3R	Immediate_offset		Immediate_of
1173	//	LD2	single_structure_8_bit_register_offset	single_structure	8_bit_register
1174	//	LD4	single_structure_8_bit_register_offset	single_structure	8_bit_register
1175	//	LD2	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1176	//	LD4	single_structure_16_bit_register_offset	single_structure	16_bit_registe
1177	//	LD2	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1178	//	LD2	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1179	//	LD4	single_structure_32_bit_register_offset	single_structure	32_bit_registe
1180	//	LD4	single_structure_64_bit_register_offset	single_structure	64_bit_registe
1181	//	LD2R	Register_offset		Register_offse
1182	//	LD4R	Register_offset		Register_offse
1183	//	LD2	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1184	//	LD4	single_structure_8_bit_immediate_offset	single_structure	8_bit_immedi
1185	//	LD2	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1186	//	LD4	single_structure_16_bit_immediate_offse	single_structure	16_bit_immec
1187	//	LD2	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1188	//	LD2	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1189	//	LD4	single_structure_32_bit_immediate_offse	single_structure	32_bit_immec
1190	//	LD4	single_structure_64_bit_immediate_offse	single_structure	64_bit_immec
1191	//	LD2R	Immediate_offset		Immediate_of
1192	//	LD4R	Immediate_offset		Immediate_of

	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
		UNALLOCATED					**0**	**0**								
		BAD	invalid operation	0	0	0	0	0	0	0	0	0	0	0	0	0
		Branch,exception generation and syst					**1**	**0**	**1**							
		Compare _ Branch (immediate)		**-**	**0**	**1**	**1**	**0**	**1**	**0**	**-**					
		CBZ		0	0	1	1	0	1	0	0					
		CBNZ		0	0	1	1	0	1	0	1					
		CBZ		1	0	1	1	0	1	0	0					
		CBNZ		1	0	1	1	0	1	0	1					
		Test bit & branch (immediate)		**b5**	**0**	**1**	**1**	**0**	**1**	**1**	**-**			**b40**		
		TBZ		b5	0	1	1	0	1	1	0			b40		
		TBNZ		b5	0	1	1	0	1	1	1			b40		
		Conditional branch (immediate)		**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	0	0		
	//	HVC		1	1	0	1	0	1	0	0	0	0	0		
	//	SMC		1	1	0	1	0	1	0	0	0	0	0		
		BRK		1	1	0	1	0	1	0	0	0	0	1		
	//	HLT		1	1	0	1	0	1	0	0	0	1	0		
	//	DCPS1		1	1	0	1	0	1	0	0	1	0	1		
	//	DCPS2		1	1	0	1	0	1	0	0	1	0	1		
	//	DCPS3		1	1	0	1	0	1	0	0	1	0	1		
	//	**System**		**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**-**	**-**	**-**
	//	MSR		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	HINT		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	CLREX		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	DSB		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	DMB		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	ISB		1	1	0	1	0	1	0	1	0	0	0	0	0
	//	SYS		1	1	0	1	0	1	0	1	0	0	0	0	1
	//	MSR		1	1	0	1	0	1	0	1	0	0	0	1	-
	//	SYSL		1	1	0	1	0	1	0	1	0	0	1	0	1
	//	MRS		1	1	0	1	0	1	0	1	0	0	1	1	-
		Unconditional branch (register)		**1**	**1**	**0**	**1**	**0**	**1**	**1**		**opc**				
		BR		1	1	0	1	0	1	1	0	0	0	0	1	1
		BLR		1	1	0	1	0	1	1	0	0	0	1	1	1

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
38		RET		1	1	0	1	0	1	1	0	0	1	0	1	1
39	//	ERET		1	1	0	1	0	1	1	0	1	0	0	1	1
40	//	DRPS		1	1	0	1	0	1	1	0	1	0	1	1	1
41	//	**Unconditional branch (immediate)**		**-**	**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	0	0		
47		STLXRB		0	0	0	0	1	0	0	0	0	0	0		
48		LDXRB		0	0	0	0	1	0	0	0	0	1	0		
49		LDAXRB		0	0	0	0	1	0	0	0	0	1	0		
50		STLRB		0	0	0	0	1	0	0	0	1	0	0		
51		LDARB		0	0	0	0	1	0	0	0	1	1	0		
52		STXRH		0	1	0	0	1	0	0	0	0	0	0		
53		STLXRH		0	1	0	0	1	0	0	0	0	0	0		
54		LDXRH		0	1	0	0	1	0	0	0	0	1	0		
55		LDAXRH		0	1	0	0	1	0	0	0	0	1	0		
56		STLRH		0	1	0	0	1	0	0	0	1	0	0		
57		LDARH		0	1	0	0	1	0	0	0	1	1	0		
58		STXR		1	0	0	0	1	0	0	0	0	0	0		
59		STLXR		1	0	0	0	1	0	0	0	0	0	0		
60		STXP		1	0	0	0	1	0	0	0	0	0	1		
61		STLXP		1	0	0	0	1	0	0	0	0	0	1		
62		LDXR		1	0	0	0	1	0	0	0	0	1	0		
63		LDAXR		1	0	0	0	1	0	0	0	0	1	0		
64		LDXP		1	0	0	0	1	0	0	0	0	1	1		
65		LDAXP		1	0	0	0	1	0	0	0	0	1	1		
66		STLR		1	0	0	0	1	0	0	0	1	0	0		
67		LDAR		1	0	0	0	1	0	0	0	1	1	0		
68		STXR		1	1	0	0	1	0	0	0	0	0	0		
69		STLXR		1	1	0	0	1	0	0	0	0	0	0		
70		STXP		1	1	0	0	1	0	0	0	0	0	1		
71		STLXP		1	1	0	0	1	0	0	0	0	0	1		
72		LDXR		1	1	0	0	1	0	0	0	0	1	0		
73		LDAXR		1	1	0	0	1	0	0	0	0	1	0		
74		LDXP		1	1	0	0	1	0	0	0	0	1	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
75		LDAXP		1	1	0	0	1	0	0	0	0	1	1		
76		STLR		1	1	0	0	1	0	0	0	1	0	0		
77		LDAR		1	1	0	0	1	0	0	0	1	1	0		
78		**Load register (literal)**		**-**	**-**	**0**	**1**	**1**	**-**	**0**	**0**					
79		LDR		0	0	0	1	1	0	0	0					
80		LDR		0	0	0	1	1	1	0	0					
81		LDR		0	1	0	1	1	0	0	0					
82		LDR		0	1	0	1	1	1	0	0					
83		LDRSW		1	0	0	1	1	0	0	0					
84		LDR		1	0	0	1	1	1	0	0					
85		PRFM		1	1	0	1	1	0	0	0					
86		**Load/store no-allocate pair (offset)**		**-**	**-**	**1**	**0**	**1**	**-**	**0**	**0**	**0**	**-**			**ii**
87		STNP		0	0	1	0	1	0	0	0	0	0			ii
88		LDNP		0	0	1	0	1	0	0	0	0	1			ii
89		STNP		0	0	1	0	1	1	0	0	0	0			ii
90		LDNP		0	0	1	0	1	1	0	0	0	1			ii
91		STNP		0	1	1	0	1	1	0	0	0	0			ii
92		LDNP		0	1	1	0	1	1	0	0	0	1			ii
93		STNP		1	0	1	0	1	0	0	0	0	0			ii
94		LDNP		1	0	1	0	1	0	0	0	0	1			ii
95		STNP		1	0	1	0	1	1	0	0	0	0			ii
96		LDNP		1	0	1	0	1	1	0	0	0	1			ii
97		**Load/store register pair (post-indexed)**		**-**	**-**	**1**	**0**	**1**	**-**	**0**	**0**	**1**	**-**			**ii**
98		STP		0	0	1	0	1	0	0	0	1	0			ii
99		LDP		0	0	1	0	1	0	0	0	1	1			ii
100		STP		0	0	1	0	1	1	0	0	1	0			ii
101		LDP		0	0	1	0	1	1	0	0	1	1			ii
102		LDPSW		0	1	1	0	1	0	0	0	1	1			ii
103		STP		0	1	1	0	1	1	0	0	1	0			ii
104		LDP		0	1	1	0	1	1	0	0	1	1			ii
105		STP		1	0	1	0	1	0	0	0	1	0			ii
106		LDP		1	0	1	0	1	0	0	0	1	1			ii
107		STP		1	0	1	0	1	1	0	0	1	0			ii
108		LDP		1	0	1	0	1	1	0	0	1	1			ii
109		**Load/store register pair (offset)**		**opc**		**1**	**0**	**1**	**V**	**0**	**1**	**0**	**L**			**ii**

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
110		STP		0	0	1	0	1	0	0	1	0	0			ii
111		LDP		0	0	1	0	1	0	0	1	0	1			ii
112		STP		0	0	1	0	1	1	0	1	0	0			ii
113		LDP		0	0	1	0	1	1	0	1	0	1			ii
114		LDPSW		0	1	1	0	1	0	0	1	0	1			ii
115		STP		0	1	1	0	1	1	0	1	0	0			ii
116		LDP		0	1	1	0	1	1	0	1	0	1			ii
117		STP		1	0	1	0	1	0	0	1	0	0			ii
118		LDP		1	0	1	0	1	0	0	1	0	1			ii
119		STP		1	0	1	0	1	1	0	1	0	0			ii
120		LDP		1	0	1	0	1	1	0	1	0	1			ii
121		**Load/store register pair (pre-indexed)**		**opc**		**1**	**0**	**1**	**V**	**0**	**1**	**1**	**L**			**ii**
122		STP		0	0	1	0	1	0	0	1	1	0			ii
123		LDP		0	0	1	0	1	0	0	1	1	1			ii
124		STP		0	0	1	0	1	1	0	1	1	0			ii
125		LDP		0	0	1	0	1	1	0	1	1	1			ii
126		LDPSW		0	1	1	0	1	0	0	1	1	1			ii
127		STP		0	1	1	0	1	1	0	1	1	0			ii
128		LDP		0	1	1	0	1	1	0	1	1	1			ii
129		STP		1	0	1	0	1	0	0	1	1	0			ii
130		LDP		1	0	1	0	1	0	0	1	1	1			ii
131		STP		1	0	1	0	1	1	0	1	1	0			ii
132		LDP		1	0	1	0	1	1	0	1	1	1			ii
133		**Load/store register (unscaled immediate)**		**size**		**1**	**1**	**1**	**V**	**0**	**0**	**opc**		**0**		
134		STURB		0	0	1	1	1	0	0	0	0	0	0		
135		LDURB		0	0	1	1	1	0	0	0	0	1	0		
136		LDURSB		0	0	1	1	1	0	0	0	1	0	0		
137		LDURSB		0	0	1	1	1	0	0	0	1	1	0		
138		STUR		0	0	1	1	1	1	0	0	0	0	0		
139		LDUR		0	0	1	1	1	1	0	0	0	1	0		
140		STUR		0	0	1	1	1	1	0	0	1	0	0		
141		LDUR		0	0	1	1	1	1	0	0	1	1	0		
142		STURH		0	1	1	1	1	0	0	0	0	0	0		
143		LDURH		0	1	1	1	1	0	0	0	0	1	0		
144		LDURSH		0	1	1	1	1	0	0	0	1	0	0		
145		LDURSH		0	1	1	1	1	0	0	0	1	1	0		
146		STUR		0	1	1	1	1	1	0	0	0	0	0		
147		LDUR		0	1	1	1	1	1	0	0	0	1	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
148		STUR		1	0	1	1	1	0	0	0	0	0	0		
149		LDUR		1	0	1	1	1	0	0	0	0	1	0		
150		LDURSW		1	0	1	1	1	0	0	0	1	0	0		
151		STUR		1	0	1	1	1	1	0	0	0	0	0		
152		LDUR		1	0	1	1	1	1	0	0	0	1	0		
153		STUR		1	1	1	1	1	0	0	0	0	0	0		
154		LDUR		1	1	1	1	1	0	0	0	0	1	0		
155		PRFUM		1	1	1	1	1	0	0	0	1	0	0		
156		STUR		1	1	1	1	1	1	0	0	0	0	0		
157		LDUR		1	1	1	1	1	1	0	0	0	1	0		
158		**Load/store register (immediate post-indexe**		**size**		**1**	**1**	**1**	**V**	**0**	**0**	**opc**		**0**		
159		STRB		0	0	1	1	1	0	0	0	0	0	0		
160		LDRB		0	0	1	1	1	0	0	0	0	1	0		
161		LDRSB		0	0	1	1	1	0	0	0	1	0	0		
162		LDRSB		0	0	1	1	1	0	0	0	1	1	0		
163		STR		0	0	1	1	1	1	0	0	0	0	0		
164		LDR		0	0	1	1	1	1	0	0	0	1	0		
165		STR		0	0	1	1	1	1	0	0	1	0	0		
166		LDR		0	0	1	1	1	1	0	0	1	1	0		
167		STRH		0	1	1	1	1	0	0	0	0	0	0		
168		LDRH		0	1	1	1	1	0	0	0	0	1	0		
169		LDRSH		0	1	1	1	1	0	0	0	1	0	0		
170		LDRSH		0	1	1	1	1	0	0	0	1	1	0		
171		STR		0	1	1	1	1	1	0	0	0	0	0		
172		LDR		0	1	1	1	1	1	0	0	0	1	0		
173		STR		1	0	1	1	1	0	0	0	0	0	0		
174		LDR		1	0	1	1	1	0	0	0	0	1	0		
175		LDRSW		1	0	1	1	1	0	0	0	1	0	0		
176		STR		1	0	1	1	1	1	0	0	0	0	0		
177		LDR		1	0	1	1	1	1	0	0	0	1	0		
178		STR		1	1	1	1	1	0	0	0	0	0	0		
179		LDR		1	1	1	1	1	0	0	0	0	1	0		
180		STR		1	1	1	1	1	1	0	0	0	0	0		
181		LDR		1	1	1	1	1	1	0	0	0	1	0		
182		**Load/store register (unprivileged)**		**size**		**1**	**1**	**1**	**V**	**0**	**0**	**opc**		**0**		
183		STTRB		0	0	1	1	1	0	0	0	0	0	0		
184		LDTRB		0	0	1	1	1	0	0	0	0	1	0		
185		LDTRSB		0	0	1	1	1	0	0	0	1	0	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
186		LDTRSB		0	0	1	1	1	0	0	0	1	1	0		
187		STTRH		0	1	1	1	1	0	0	0	0	0	0		
188		LDTRH		0	1	1	1	1	0	0	0	0	1	0		
189		LDTRSH		0	1	1	1	1	0	0	0	1	0	0		
190		LDTRSH		0	1	1	1	1	0	0	0	1	1	0		
191		STTR		1	0	1	1	1	0	0	0	0	0	0		
192		LDTR		1	0	1	1	1	0	0	0	0	1	0		
193		LDTRSW		1	0	1	1	1	0	0	0	1	0	0		
194		STTR		1	1	1	1	1	0	0	0	0	0	0		
195		LDTR		1	1	1	1	1	0	0	0	0	1	0		
196		**Load/store register (immediate pre-indexec**		**size**		**1**	**1**	**1**	**V**	**0**	**0**	**opc**		**0**		
197		STRB		0	0	1	1	1	0	0	0	0	0	0		
198		LDRB		0	0	1	1	1	0	0	0	0	1	0		
199		LDRSB		0	0	1	1	1	0	0	0	1	0	0		
200		LDRSB		0	0	1	1	1	0	0	0	1	1	0		
201		STR		0	0	1	1	1	1	0	0	0	0	0		
202		LDR		0	0	1	1	1	1	0	0	0	1	0		
203		STR		0	0	1	1	1	1	0	0	1	0	0		
204		LDR		0	0	1	1	1	1	0	0	1	1	0		
205		STRH		0	1	1	1	1	0	0	0	0	0	0		
206		LDRH		0	1	1	1	1	0	0	0	0	1	0		
207		LDRSH		0	1	1	1	1	0	0	0	1	0	0		
208		LDRSH		0	1	1	1	1	0	0	0	1	1	0		
209		STR		0	1	1	1	1	1	0	0	0	0	0		
210		LDR		0	1	1	1	1	1	0	0	0	1	0		
211		STR		1	0	1	1	1	0	0	0	0	0	0		
212		LDR		1	0	1	1	1	0	0	0	0	1	0		
213		LDRSW		1	0	1	1	1	0	0	0	1	0	0		
214		STR		1	0	1	1	1	1	0	0	0	0	0		
215		LDR		1	0	1	1	1	1	0	0	0	1	0		
216		STR		1	1	1	1	1	0	0	0	0	0	0		
217		LDR		1	1	1	1	1	0	0	0	0	1	0		
218		STR		1	1	1	1	1	1	0	0	0	0	0		
219		LDR		1	1	1	1	1	1	0	0	0	1	0		
220		**Load/store register (register offset)**		**size**		**1**	**1**	**1**	**v**	**0**	**0**	**opc**		**1**		
221		STRB		0	0	1	1	1	0	0	0	0	0	1		
222		LDRB		0	0	1	1	1	0	0	0	0	1	1		
223		LDRSB		0	0	1	1	1	0	0	0	1	0	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
224		LDRSB		0	0	1	1	1	0	0	0	1	1	1		
225		STR		0	0	1	1	1	1	0	0	0	0	1		
226		LDR		0	0	1	1	1	1	0	0	0	1	1		
227		STR		0	0	1	1	1	1	0	0	1	0	1		
228		LDR		0	0	1	1	1	1	0	0	1	1	1		
229		STRH		0	1	1	1	1	0	0	0	0	0	1		
230		LDRH		0	1	1	1	1	0	0	0	0	1	1		
231		LDRSH		0	1	1	1	1	0	0	0	1	0	1		
232		LDRSH		0	1	1	1	1	0	0	0	1	1	1		
233		STR		0	1	1	1	1	1	0	0	0	0	1		
234		LDR		0	1	1	1	1	1	0	0	0	1	1		
235		STR		1	0	1	1	1	0	0	0	0	0	1		
236		LDR		1	0	1	1	1	0	0	0	0	1	1		
237		LDRSW		1	0	1	1	1	0	0	0	1	0	1		
238		STR		1	0	1	1	1	1	0	0	0	0	1		
239		LDR		1	0	1	1	1	1	0	0	0	1	1		
240		STR		1	1	1	1	1	0	0	0	0	0	1		
241		LDR		1	1	1	1	1	0	0	0	0	1	1		
243		STR		1	1	1	1	1	1	0	0	0	0	1		
244		LDR		1	1	1	1	1	1	0	0	0	1	1		
242		PRFM		1	1	1	1	1	0	0	0	1	0	1		
245		**Load/store register (unsigned immediate)**		**size**		**1**	**1**	**1**	**v**	**0**	**1**	**opc**				
246		STRB		0	0	1	1	1	0	0	1	0	0			
247		LDRB		0	0	1	1	1	0	0	1	0	1			
248		LDRSB		0	0	1	1	1	0	0	1	1	0			
249		LDRSB		0	0	1	1	1	0	0	1	1	1			
250		STR		0	0	1	1	1	1	0	1	0	0			
251		LDR		0	0	1	1	1	1	0	1	0	1			
252		STR		0	0	1	1	1	1	0	1	1	0			
253		LDR		0	0	1	1	1	1	0	1	1	1			
254		STRH		0	1	1	1	1	0	0	1	0	0			
255		LDRH		0	1	1	1	1	0	0	1	0	1			
256		LDRSH		0	1	1	1	1	0	0	1	1	0			
257		LDRSH		0	1	1	1	1	0	0	1	1	1			
258		STR		0	1	1	1	1	1	0	1	0	0			
259		LDR		0	1	1	1	1	1	0	1	0	1			
260		STR		1	0	1	1	1	0	0	1	0	0			
261		LDR		1	0	1	1	1	0	0	1	0	1			

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
262		LDRSW		1	0	1	1	1	0	0	1	1	0			
263		STR		1	0	1	1	1	1	0	1	0	0			
264		LDR		1	0	1	1	1	1	0	1	0	1			
265		STR		1	1	1	1	1	0	0	1	0	0			
266		LDR		1	1	1	1	1	0	0	1	0	1			
268		STR		1	1	1	1	1	1	0	1	0	0			
269		LDR		1	1	1	1	1	1	0	1	0	1			
267		PRFM		1	1	1	1	1	0	0	1	1	0			
270		**Data processing – Immediate**					**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**	**shift**				
275		ADD		0	0	0	1	0	0	0	1	-	-			
276		ADDS		0	0	1	1	0	0	0	1	-	-			
277		SUB		0	1	0	1	0	0	0	1	-	-			
278		SUBS		0	1	1	1	0	0	0	1	-	-			
279		ADD		1	0	0	1	0	0	0	1	-	-			
280		ADDS		1	0	1	1	0	0	0	1	-	-			
281		SUB		1	1	0	1	0	0	0	1	-	-			
282		SUBS		1	1	1	1	0	0	0	1	-	-			
283		**Logical (immediate)**		**sf**	**opc**		**1**	**0**	**0**	**1**	**0**	**0**	**N**			**imr**
284		AND		0	0	0	1	0	0	1	0	0	0			imr
285		ORR		0	0	1	1	0	0	1	0	0	0			imr
286		EOR		0	1	0	1	0	0	1	0	0	0			imr
287		ANDS		0	1	1	1	0	0	1	0	0	0			imr
288		AND		1	0	0	1	0	0	1	0	0	-			imr
289		ORR		1	0	1	1	0	0	1	0	0	-			imr
290		EOR		1	1	0	1	0	0	1	0	0	-			imr
291		ANDS		1	1	1	1	0	0	1	0	0	-			imr
292		**Move wide (immediate)**		**sf**	**opc**		**1**	**0**	**0**	**1**	**0**	**1**	**hw**			
293		MOVN		0	0	0	1	0	0	1	0	1	-	-		
294		MOVZ		0	1	0	1	0	0	1	0	1	-	-		
295		MOVK		0	1	1	1	0	0	1	0	1	-	-		
296		MOVN		1	0	0	1	0	0	1	0	1	-	-		
297		MOVZ		1	1	0	1	0	0	1	0	1	-	-		
298		MOVK		1	1	1	1	0	0	1	0	1	-	-		
299		**Bitfield**		**sf**	**opc**		**1**	**0**	**0**	**1**	**1**	**0**	**N**			**imr**

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
300		SBFM		0	0	0	1	0	0	1	1	0	0			im
301		BFM		0	0	1	1	0	0	1	1	0	0			im
302		UBFM		0	1	0	1	0	0	1	1	0	0			im
303		SBFM		1	0	0	1	0	0	1	1	0	1			im
304		BFM		1	0	1	1	0	0	1	1	0	1			im
305		UBFM		1	1	0	1	0	0	1	1	0	1			im
306		**Extract**		**sf**	**op21**		**1**	**0**	**0**	**1**	**1**	**1**	**N**	**o0**		
307		EXTR		0	0	0	1	0	0	1	1	1	0	0		
308		EXTR		1	0	0	1	0	0	1	1	1	1	0		
309		**Data Processing – register**						**1**	**0**	**1**						
310		**Logical (shifted register)**		**sf**	**opc**		**0**	**1**	**0**	**1**	**0**	**shift**		**N**		
311		AND		0	0	0	0	1	0	1	0	shift		0		
312		BIC		0	0	0	0	1	0	1	0	shift		1		
313		ORR		0	0	1	0	1	0	1	0	shift		0		
314		ORN		0	0	1	0	1	0	1	0	shift		1		
315		EOR		0	1	0	0	1	0	1	0	shift		0		
316		EON		0	1	0	0	1	0	1	0	shift		1		
317		ANDS		0	1	1	0	1	0	1	0	shift		0		
318		BICS		0	1	1	0	1	0	1	0	shift		1		
319		AND		1	0	0	0	1	0	1	0	shift		0		
320		BIC		1	0	0	0	1	0	1	0	shift		1		
321		ORR		1	0	1	0	1	0	1	0	shift		0		
322		ORN		1	0	1	0	1	0	1	0	shift		1		
323		EOR		1	1	0	0	1	0	1	0	shift		0		
324		EON		1	1	0	0	1	0	1	0	shift		1		
325		ANDS		1	1	1	0	1	0	1	0	shift		0		
326		BICS		1	1	1	0	1	0	1	0	shift		1		
327		**Add/subtract (shifted register)**		**sf**	**op**	**S**	**0**	**1**	**0**	**1**	**1**	**shift**		**0**		
328		ADD		0	0	0	0	1	0	1	1	-	-	0		
329		ADDS		0	0	1	0	1	0	1	1	-	-	0		
330		SUB		0	1	0	0	1	0	1	1	-	-	0		
331		SUBS		0	1	1	0	1	0	1	1	-	-	0		
332		ADD		1	0	0	0	1	0	1	1	-	-	0		
333		ADDS		1	0	1	0	1	0	1	1	-	-	0		
334		SUB		1	1	0	0	1	0	1	1	-	-	0		
335		SUBS		1	1	1	0	1	0	1	1	-	-	0		
336		**Add/subtract (extended register)**		**sf**	**op**	**S**	**0**	**1**	**0**	**1**	**1**	**opt**		**1**		
337		ADD		0	0	0	0	1	0	1	1	0	0	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
338		ADDS		0	0	1	0	1	0	1	1	0	0	1		
339		SUB		0	1	0	0	1	0	1	1	0	0	1		
340		SUBS		0	1	1	0	1	0	1	1	0	0	1		
341		ADD		1	0	0	0	1	0	1	1	0	0	1		
342		ADDS		1	0	1	0	1	0	1	1	0	0	1		
343		SUB		1	1	0	0	1	0	1	1	0	0	1		
344		SUBS		1	1	1	0	1	0	1	1	0	0	1		
345		**Add/subtract (with carry)**		**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**		
346		ADC		0	0	0	1	1	0	1	0	0	0	0		
347		ADCS		0	0	1	1	1	0	1	0	0	0	0		
348		SBC		0	1	0	1	1	0	1	0	0	0	0		
349		SBCS		0	1	1	1	1	0	1	0	0	0	0		
350		ADC		1	0	0	1	1	0	1	0	0	0	0		
351		ADCS		1	0	1	1	1	0	1	0	0	0	0		
352		SBC		1	1	0	1	1	0	1	0	0	0	0		
353		SBCS		1	1	1	1	1	0	1	0	0	0	0		
354		**Conditional compare (register)**		**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**		**i**
355		CCMN		0	0	1	1	1	0	1	0	0	1	0		i
356		CCMN		1	0	1	1	1	0	1	0	0	1	0		i
357		CCMP		0	1	1	1	1	0	1	0	0	1	0		i
358		CCMP		1	1	1	1	1	0	1	0	0	1	0		i
359		**Conditional compare (immediate)**		**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**		**i**
360		CCMN		0	0	1	1	1	0	1	0	0	1	0		i
361		CCMN		1	0	1	1	1	0	1	0	0	1	0		i
362		CCMP		0	1	1	1	1	0	1	0	0	1	0		i
363		CCMP		1	1	1	1	1	0	1	0	0	1	0		i
364		**Conditional select**		**sf**	**op**	**S**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**		
365		CSEL		0	0	0	1	1	0	1	0	1	0	0		
366		CSINC		0	0	0	1	1	0	1	0	1	0	0		
367		CSINV		0	1	0	1	1	0	1	0	1	0	0		
368		CSNEG		0	1	0	1	1	0	1	0	1	0	0		
369		CSEL		1	0	0	1	1	0	1	0	1	0	0		
370		CSINC		1	0	0	1	1	0	1	0	1	0	0		
371		CSINV		1	1	0	1	1	0	1	0	1	0	0		
372		CSNEG		1	1	0	1	1	0	1	0	1	0	0		
373		**Data-processing (3 source)**		**sf**	**op54**		**1**	**1**	**0**	**1**	**1**	**op31**				
374		MADD		0	0	0	1	1	0	1	1	0	0	0		
375		MADD		1	0	0	1	1	0	1	1	0	0	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
376		SMADDL		1	0	0	1	1	0	1	1	0	0	1		
377		UMADDL		1	0	0	1	1	0	1	1	1	0	1		
378		MSUB		0	0	0	1	1	0	1	1	0	0	0		
379		MSUB		1	0	0	1	1	0	1	1	0	0	0		
380		SMSUBL		1	0	0	1	1	0	1	1	0	0	1		
381		UMSUBL		1	0	0	1	1	0	1	1	1	0	1		
382		SMULH		1	0	0	1	1	0	1	1	0	1	0		
383		UMULH		1	0	0	1	1	0	1	1	1	1	0		
384		**Data-processing (2 source)**		**sf**	**0**	**S**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**		
385		CRC32X		1	0	0	1	1	0	1	0	1	1	0		
386		CRC32CX		1	0	0	1	1	0	1	0	1	1	0		
387		CRC32B		0	0	0	1	1	0	1	0	1	1	0		
388		CRC32CB		0	0	0	1	1	0	1	0	1	1	0		
389		CRC32H		0	0	0	1	1	0	1	0	1	1	0		
390		CRC32CH		0	0	0	1	1	0	1	0	1	1	0		
391		CRC32W		0	0	0	1	1	0	1	0	1	1	0		
392		CRC32CW		0	0	0	1	1	0	1	0	1	1	0		
393		UDIV		0	0	0	1	1	0	1	0	1	1	0		
394		UDIV		1	0	0	1	1	0	1	0	1	1	0		
395		SDIV		0	0	0	1	1	0	1	0	1	1	0		
396		SDIV		1	0	0	1	1	0	1	0	1	1	0		
397		LSLV		0	0	0	1	1	0	1	0	1	1	0		
398		LSLV		1	0	0	1	1	0	1	0	1	1	0		
399		LSRV		0	0	0	1	1	0	1	0	1	1	0		
400		LSRV		1	0	0	1	1	0	1	0	1	1	0		
401		ASRV		0	0	0	1	1	0	1	0	1	1	0		
402		ASRV		1	0	0	1	1	0	1	0	1	1	0		
403		RORV		0	0	0	1	1	0	1	0	1	1	0		
404		RORV		1	0	0	1	1	0	1	0	1	1	0		
405		**Data-processing (1 source)**		**sf**	**1**	**S**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**		**op**
406		RBIT		0	1	0	1	1	0	1	0	1	1	0	0	0
407		RBIT		1	1	0	1	1	0	1	0	1	1	0	0	0
408		CLZ		0	1	0	1	1	0	1	0	1	1	0	0	0
409		CLZ		1	1	0	1	1	0	1	0	1	1	0	0	0
410		CLS		0	1	0	1	1	0	1	0	1	1	0	0	0
411		CLS		1	1	0	1	1	0	1	0	1	1	0	0	0
412		REV		0	1	0	1	1	0	1	0	1	1	0	0	0
413		REV		1	1	0	1	1	0	1	0	1	1	0	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
414		REV16		1	1	0	1	1	0	1	0	1	1	0	0	0
415		REV16		0	1	0	1	1	0	1	0	1	1	0	0	0
416		REV32		1	1	0	1	1	0	1	0	1	1	0	0	0
417	//	**Data Processing – SIMD and floating p**						**1**	**1**	**1**						
418	//	**Floating-point<->fixed-point conversions**		**sf**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**0**	**rmode**	
419	//	SCVTF		0	0	0	1	1	1	1	0	0	0	0	0	0
420	//	UCVTF		0	0	0	1	1	1	1	0	0	0	0	0	0
421	//	FCVTZS		0	0	0	1	1	1	1	0	1	1	0	1	1
422	//	FCVTZU		0	0	0	1	1	1	1	0	1	1	0	1	1
423	//	SCVTF		0	0	0	1	1	1	1	0	0	0	0	0	0
424	//	UCVTF		0	0	0	1	1	1	1	0	0	0	0	0	0
425	//	FCVTZS		0	0	0	1	1	1	1	0	1	1	0	1	1
426	//	FCVTZU		0	0	0	1	1	1	1	0	1	1	0	1	1
427	//	SCVTF		1	0	0	1	1	1	1	0	0	0	0	0	0
428	//	UCVTF		1	0	0	1	1	1	1	0	0	0	0	0	0
429	//	FCVTZS		1	0	0	1	1	1	1	0	1	1	0	1	1
430	//	FCVTZU		1	0	0	1	1	1	1	0	1	1	0	1	1
431	//	SCVTF		1	0	0	1	1	1	1	0	0	0	0	0	0
432	//	UCVTF		1	0	0	1	1	1	1	0	0	0	0	0	0
433	//	FCVTZS		1	0	0	1	1	1	1	0	1	1	0	1	1
434	//	FCVTZU		1	0	0	1	1	1	1	0	1	1	0	1	1
435	//	**Floating-point conditional compare**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
436	//	FCCMP		0	0	0	1	1	1	1	0	0	0	1		
437	//	FCCMPE		0	0	0	1	1	1	1	0	0	0	1		
438	//	FCCMP		0	0	0	1	1	1	1	0	0	1	1		
439	//	FCCMPE		0	0	0	1	1	1	1	0	0	1	1		
440	//	**Floating-point data-processing (2 source)**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
441	//	FMUL		0	0	0	1	1	1	1	0	0	0	1		
442	//	FDIV		0	0	0	1	1	1	1	0	0	0	1		
443	//	FADD		0	0	0	1	1	1	1	0	0	0	1		
444	//	FSUB		0	0	0	1	1	1	1	0	0	0	1		
445	//	FMAX		0	0	0	1	1	1	1	0	0	0	1		
446	//	FMIN		0	0	0	1	1	1	1	0	0	0	1		
447	//	FMAXNM		0	0	0	1	1	1	1	0	0	0	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
448	//	FMINNM		0	0	0	1	1	1	1	0	0	0	1		
449	//	FNMUL		0	0	0	1	1	1	1	0	0	0	1		
450	//	FMUL		0	0	0	1	1	1	1	0	0	1	1		
451	//	FDIV		0	0	0	1	1	1	1	0	0	1	1		
452	//	FADD		0	0	0	1	1	1	1	0	0	1	1		
453	//	FSUB		0	0	0	1	1	1	1	0	0	1	1		
454	//	FMAX		0	0	0	1	1	1	1	0	0	1	1		
455	//	FMIN		0	0	0	1	1	1	1	0	0	1	1		
456	//	FMAXNM		0	0	0	1	1	1	1	0	0	1	1		
457	//	FMINNM		0	0	0	1	1	1	1	0	0	1	1		
458	//	FNMUL		0	0	0	1	1	1	1	0	0	1	1		
459	//	**Floating-point conditional select**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
460	//	FCSEL		0	0	0	1	1	1	1	0	0	0	1		
461	//	FCSEL		0	0	0	1	1	1	1	0	0	1	1		
462	//	**Floating-point immediate**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
463	//	FMOV		0	0	0	1	1	1	1	0	0	0	1		
464	//	FMOV		0	0	0	1	1	1	1	0	0	1	1		
465	//	**Floating-point compare**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
466	//	FCMP		0	0	0	1	1	1	1	0	0	0	1		
467	//	FCMP		0	0	0	1	1	1	1	0	0	0	1		
468	//	FCMPE		0	0	0	1	1	1	1	0	0	0	1		
469	//	FCMPE		0	0	0	1	1	1	1	0	0	0	1		
470	//	FCMP		0	0	0	1	1	1	1	0	0	1	1		
471	//	FCMP		0	0	0	1	1	1	1	0	0	1	1		
472	//	FCMPE		0	0	0	1	1	1	1	0	0	1	1		
473	//	FCMPE		0	0	0	1	1	1	1	0	0	1	1		
474	//	**Floating-point data-processing (1 source)**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**		
475	//	FMOV		0	0	0	1	1	1	1	0	0	0	1	0	0
476	//	FABS		0	0	0	1	1	1	1	0	0	0	1	0	0
477	//	FNEG		0	0	0	1	1	1	1	0	0	0	1	0	0
478	//	FSQRT		0	0	0	1	1	1	1	0	0	0	1	0	0
479	//	FCVT		0	0	0	1	1	1	1	0	0	0	1	0	0
480	//	FCVT		0	0	0	1	1	1	1	0	0	0	1	0	0
481	//	FRINTN		0	0	0	1	1	1	1	0	0	0	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
482	//	FRINTP		0	0	0	1	1	1	1	0	0	0	1	0	0
483	//	FRINTM		0	0	0	1	1	1	1	0	0	0	1	0	0
484	//	FRINTZ		0	0	0	1	1	1	1	0	0	0	1	0	0
485	//	FRINTA		0	0	0	1	1	1	1	0	0	0	1	0	0
486	//	FRINTX		0	0	0	1	1	1	1	0	0	0	1	0	0
487	//	FRINTI		0	0	0	1	1	1	1	0	0	0	1	0	0
488	//	FMOV		0	0	0	1	1	1	1	0	0	1	1	0	0
489	//	FABS		0	0	0	1	1	1	1	0	0	1	1	0	0
490	//	FNEG		0	0	0	1	1	1	1	0	0	1	1	0	0
491	//	FSQRT		0	0	0	1	1	1	1	0	0	1	1	0	0
492	//	FCVT		0	0	0	1	1	1	1	0	0	1	1	0	0
493	//	FCVT		0	0	0	1	1	1	1	0	0	1	1	0	0
494	//	FRINTN		0	0	0	1	1	1	1	0	0	1	1	0	0
495	//	FRINTP		0	0	0	1	1	1	1	0	0	1	1	0	0
496	//	FRINTM		0	0	0	1	1	1	1	0	0	1	1	0	0
497	//	FRINTZ		0	0	0	1	1	1	1	0	0	1	1	0	0
498	//	FRINTA		0	0	0	1	1	1	1	0	0	1	1	0	0
499	//	FRINTX		0	0	0	1	1	1	1	0	0	1	1	0	0
500	//	FRINTI		0	0	0	1	1	1	1	0	0	1	1	0	0
501	//	FCVT		0	0	0	1	1	1	1	0	1	1	1	0	0
502	//	FCVT		0	0	0	1	1	1	1	0	1	1	1	0	0
503	//	**Floating-point<->integer conversions**		**sf**	**0**	**S**	**1**	**1**	**1**	**1**	**0**	**type**		**1**	**rmode**	
504	//	FCVTNS		0	0	0	1	1	1	1	0	0	0	1	0	0
505	//	FCVTNU		0	0	0	1	1	1	1	0	0	0	1	0	0
506	//	SCVTF		0	0	0	1	1	1	1	0	0	0	1	0	0
507	//	UCVTF		0	0	0	1	1	1	1	0	0	0	1	0	0
508	//	FCVTAS		0	0	0	1	1	1	1	0	0	0	1	0	0
509	//	FCVTAU		0	0	0	1	1	1	1	0	0	0	1	0	0
510	//	FMOV		0	0	0	1	1	1	1	0	0	0	1	0	0
511	//	FMOV		0	0	0	1	1	1	1	0	0	0	1	0	0
512	//	FCVTPS		0	0	0	1	1	1	1	0	0	0	1	0	1
513	//	FCVTPU		0	0	0	1	1	1	1	0	0	0	1	0	1
514	//	FCVTMS		0	0	0	1	1	1	1	0	0	0	1	1	0
515	//	FCVTMU		0	0	0	1	1	1	1	0	0	0	1	1	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
516	//	FCVTZS		0	0	0	1	1	1	1	0	0	0	1	1	1
517	//	FCVTZU		0	0	0	1	1	1	1	0	0	0	1	1	1
518	//	FCVTNS		0	0	0	1	1	1	1	0	0	1	1	0	0
519	//	FCVTNU		0	0	0	1	1	1	1	0	0	1	1	0	0
520	//	SCVTF		0	0	0	1	1	1	1	0	0	1	1	0	0
521	//	UCVTF		0	0	0	1	1	1	1	0	0	1	1	0	0
522	//	FCVTAS		0	0	0	1	1	1	1	0	0	1	1	0	0
523	//	FCVTAU		0	0	0	1	1	1	1	0	0	1	1	0	0
524	//	FCVTPS		0	0	0	1	1	1	1	0	0	1	1	0	1
525	//	FCVTPU		0	0	0	1	1	1	1	0	0	1	1	0	1
526	//	FCVTMS		0	0	0	1	1	1	1	0	0	1	1	1	0
527	//	FCVTMU		0	0	0	1	1	1	1	0	0	1	1	1	0
528	//	FCVTZS		0	0	0	1	1	1	1	0	0	1	1	1	1
529	//	FCVTZU		0	0	0	1	1	1	1	0	0	1	1	1	1
530	//	FCVTNS		1	0	0	1	1	1	1	0	0	0	1	0	0
531	//	FCVTNU		1	0	0	1	1	1	1	0	0	0	1	0	0
532	//	SCVTF		1	0	0	1	1	1	1	0	0	0	1	0	0
533	//	UCVTF		1	0	0	1	1	1	1	0	0	0	1	0	0
534	//	FCVTAS		1	0	0	1	1	1	1	0	0	0	1	0	0
535	//	FCVTAU		1	0	0	1	1	1	1	0	0	0	1	0	0
536	//	FCVTPS		1	0	0	1	1	1	1	0	0	0	1	0	1
537	//	FCVTPU		1	0	0	1	1	1	1	0	0	0	1	0	1
538	//	FCVTMS		1	0	0	1	1	1	1	0	0	0	1	1	0
539	//	FCVTMU		1	0	0	1	1	1	1	0	0	0	1	1	0
540	//	FCVTZS		1	0	0	1	1	1	1	0	0	0	1	1	1
541	//	FCVTZU		1	0	0	1	1	1	1	0	0	0	1	1	1
542	//	FCVTNS		1	0	0	1	1	1	1	0	0	0	1	0	0
543	//	FCVTNU		1	0	0	1	1	1	1	0	0	0	1	0	0
544	//	SCVTF		1	0	0	1	1	1	1	0	0	1	1	0	0
545	//	UCVTF		1	0	0	1	1	1	1	0	0	1	1	0	0
546	//	FCVTAS		1	0	0	1	1	1	1	0	0	1	1	0	0
547	//	FCVTAU		1	0	0	1	1	1	1	0	0	1	1	0	0
548	//	FMOV		1	0	0	1	1	1	1	0	0	1	1	0	0
549	//	FMOV		1	0	0	1	1	1	1	0	0	1	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
550	//	FCVTPS		1	0	0	1	1	1	1	0	0	1	1	0	1
551	//	FCVTPU		1	0	0	1	1	1	1	0	0	1	1	0	1
552	//	FCVTMS		1	0	0	1	1	1	1	0	0	1	1	1	0
553	//	FCVTMU		1	0	0	1	1	1	1	0	0	1	1	1	0
554	//	FCVTZS		1	0	0	1	1	1	1	0	0	1	1	1	1
555	//	FCVTZU		1	0	0	1	1	1	1	0	0	1	1	1	1
556	//	FMOV		1	0	0	1	1	1	1	0	1	0	1	0	1
557	//	FMOV		1	0	0	1	1	1	1	0	1	0	1	0	1
558	//	**Floating-point data-processing (3 source)**		**M**	**0**	**S**	**1**	**1**	**1**	**1**	**1**	**type**		**o1**		
559	//	FMADD		0	0	0	1	1	1	1	1	0	0	0		
560	//	FMSUB		0	0	0	1	1	1	1	1	0	0	0		
561	//	FNMADD		0	0	0	1	1	1	1	1	0	0	1		
562	//	FNMSUB		0	0	0	1	1	1	1	1	0	0	1		
563	//	FMADD		0	0	0	1	1	1	1	1	0	1	0		
564	//	FMSUB		0	0	0	1	1	1	1	1	0	1	0		
565	//	FNMADD		0	0	0	1	1	1	1	1	0	1	1		
566	//	FNMSUB		0	0	0	1	1	1	1	1	0	1	1		
567	//	**AdvSIMD scalar three same**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**size**		**1**		
568	//	SQADD		0	1	0	1	1	1	1	0	-	-	1		
569	//	SQSUB		0	1	0	1	1	1	1	0	-	-	1		
570	//	CMGT		0	1	0	1	1	1	1	0	-	-	1		
571	//	CMGE		0	1	0	1	1	1	1	0	-	-	1		
572	//	SSHL		0	1	0	1	1	1	1	0	-	-	1		
573	//	SQSHL		0	1	0	1	1	1	1	0	-	-	1		
574	//	SRSHL		0	1	0	1	1	1	1	0	-	-	1		
575	//	SQRSHL		0	1	0	1	1	1	1	0	-	-	1		
576	//	ADD		0	1	0	1	1	1	1	0	-	-	1		
577	//	CMTST		0	1	0	1	1	1	1	0	-	-	1		
578	//	SQDMULH		0	1	0	1	1	1	1	0	-	-	1		
579	//	FMULX		0	1	0	1	1	1	1	0	0	x	1		
580	//	FCMEQ		0	1	0	1	1	1	1	0	0	x	1		
581	//	FRECPS		0	1	0	1	1	1	1	0	0	x	1		
582	//	FRSQRTS		0	1	0	1	1	1	1	0	1	x	1		
583	//	UQADD		0	1	1	1	1	1	1	0	-	-	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
584	//	UQSUB		0	1	1	1	1	1	1	0	-	-	1		
585	//	CMHI		0	1	1	1	1	1	1	0	-	-	1		
586	//	CMHS		0	1	1	1	1	1	1	0	-	-	1		
587	//	USHL		0	1	1	1	1	1	1	0	-	-	1		
588	//	UQSHL		0	1	1	1	1	1	1	0	-	-	1		
589	//	URSHL		0	1	1	1	1	1	1	0	-	-	1		
590	//	UQRSHL		0	1	1	1	1	1	1	0	-	-	1		
591	//	SUB		0	1	1	1	1	1	1	0	-	-	1		
592	//	CMEQ		0	1	1	1	1	1	1	0	-	-	1		
593	//	SQRDMULH		0	1	1	1	1	1	1	0	-	-	1		
594	//	FCMGE		0	1	1	1	1	1	1	0	0	x	1		
595	//	FACGE		0	1	1	1	1	1	1	0	0	x	1		
596	//	FABD		0	1	1	1	1	1	1	0	1	x	1		
597	//	FCMGT		0	1	1	1	1	1	1	0	1	x	1		
598	//	FACGT		0	1	1	1	1	1	1	0	1	x	1		
599	//	**AdvSIMD scalar three different**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**size**		**1**		
600	//	SQDMLAL	writes to low half of the dest. register	0	1	0	1	1	1	1	0	size		1		
601	//	SQDMLAL2	writes to high half of the dest. registe	0	1	0	1	1	1	1	0	size		1		
602	//	SQDMLSL	writes to low half of the dest. register	0	1	0	1	1	1	1	0	size		1		
603	//	SQDMLSL2	writes to high half of the dest. registe	0	1	0	1	1	1	1	0	size		1		
604	//	SQDMULL	writes to low half of the dest. register	0	1	0	1	1	1	1	0	size		1		
605	//	SQDMULL2	writes to high half of the dest. registe	0	1	0	1	1	1	1	0	size		1		
606	//	**AdvSIMD scalar two-reg misc**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**size**		**1**	**0**	**0**
607	//	SUQADD		0	1	0	1	1	1	1	0	-	-	1	0	0
608	//	SQABS		0	1	0	1	1	1	1	0	-	-	1	0	0
609	//	CMGT		0	1	0	1	1	1	1	0	-	-	1	0	0
610	//	CMEQ		0	1	0	1	1	1	1	0	-	-	1	0	0
611	//	CMLT		0	1	0	1	1	1	1	0	-	-	1	0	0
612	//	ABS		0	1	0	1	1	1	1	0	-	-	1	0	0
613	//	SQXTN	writes to low half of the dest. register	0	1	0	1	1	1	1	0	-	-	1	0	0
614	//	SQXTN2	writes to high half of the dest. registe	0	1	0	1	1	1	1	0	-	-	1	0	0
615	//	FCVTNS		0	1	0	1	1	1	1	0	0	x	1	0	0
616	//	FCVTMS		0	1	0	1	1	1	1	0	0	x	1	0	0
617	//	FCVTAS		0	1	0	1	1	1	1	0	0	x	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
618	//	SCVTF		0	1	0	1	1	1	1	0	0	x	1	0	0
619	//	FCMGT		0	1	0	1	1	1	1	0	1	x	1	0	0
620	//	FCMEQ		0	1	0	1	1	1	1	0	1	x	1	0	0
621	//	FCMLT		0	1	0	1	1	1	1	0	1	x	1	0	0
622	//	FCVTPS		0	1	0	1	1	1	1	0	1	x	1	0	0
623	//	FCVTZS		0	1	0	1	1	1	1	0	1	x	1	0	0
624	//	FRECPE		0	1	0	1	1	1	1	0	1	x	1	0	0
625	//	FRECPX		0	1	0	1	1	1	1	0	1	x	1	0	0
626	//	USQADD		0	1	1	1	1	1	1	0	-	-	1	0	0
627	//	SQNEG		0	1	1	1	1	1	1	0	-	-	1	0	0
628	//	CMGE		0	1	1	1	1	1	1	0	-	-	1	0	0
629	//	CMLE		0	1	1	1	1	1	1	0	-	-	1	0	0
630	//	NEG		0	1	1	1	1	1	1	0	-	-	1	0	0
631	//	SQXTUN	writes to low half of the dest. register	0	1	1	1	1	1	1	0	-	-	1	0	0
632	//	SQXTUN2	writes to high half of the dest. registe	0	1	1	1	1	1	1	0	-	-	1	0	0
633	//	UQXTN	writes to low half of the dest. register	0	1	1	1	1	1	1	0	-	-	1	0	0
634	//	UQXTN2	writes to high half of the dest. registe	0	1	1	1	1	1	1	0	-	-	1	0	0
635	//	FCVTXN	writes to low half of the dest. register	0	1	1	1	1	1	1	0	0	x	1	0	0
636	//	FCVTXN2	writes to high half of the dest. registe	0	1	1	1	1	1	1	0	0	x	1	0	0
637	//	FCVTNU		0	1	1	1	1	1	1	0	0	x	1	0	0
638	//	FCVTMU		0	1	1	1	1	1	1	0	0	x	1	0	0
639	//	FCVTAU		0	1	1	1	1	1	1	0	0	x	1	0	0
640	//	UCVTF		0	1	1	1	1	1	1	0	0	x	1	0	0
641	//	FCMGE		0	1	1	1	1	1	1	0	1	x	1	0	0
642	//	FCMLE		0	1	1	1	1	1	1	0	1	x	1	0	0
643	//	FCVTPU		0	1	1	1	1	1	1	0	1	x	1	0	0
644	//	FCVTZU		0	1	1	1	1	1	1	0	1	x	1	0	0
645	//	FRSQRTE		0	1	1	1	1	1	1	0	1	x	1	0	0
646	//	**AdvSIMD scalar pairwise**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**0**	**size**		**1**	**1**	**0**
647	//	ADDP		0	1	0	1	1	1	1	0	-	-	1	1	0
648	//	FMAXNMP		0	1	1	1	1	1	1	0	0	x	1	1	0
649	//	FADDP		0	1	1	1	1	1	1	0	0	x	1	1	0
650	//	FMAXP		0	1	1	1	1	1	1	0	0	x	1	1	0
651	//	FMINNMP		0	1	1	1	1	1	1	0	1	x	1	1	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
652	//	FMINP		0	1	1	1	1	1	1	0	1	x	1	1	0
653	//	**AdvSIMD scalar copy**		**0**	**1**	**op**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**		**ii**
654	//	DUP		0	1	0	1	1	1	1	0	0	0	0	-	-
655	//	**AdvSIMD scalar x indexed element**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**1**	**size**		**L**	**M**	
656	//	SQDMLAL		0	1	0	1	1	1	1	1	-	-	L	M	
657	//	SQDMLAL2		0	1	0	1	1	1	1	1	-	-	L	M	
658	//	SQDMLSL		0	1	0	1	1	1	1	1	-	-	L	M	
659	//	SQDMLSL2		0	1	0	1	1	1	1	1	-	-	L	M	
660	//	SQDMULL		0	1	0	1	1	1	1	1	-	-	L	M	
661	//	SQDMULL2		0	1	0	1	1	1	1	1	-	-	L	M	
662	//	SQDMULH		0	1	0	1	1	1	1	1	-	-	L	M	
663	//	SQRDMULH		0	1	0	1	1	1	1	1	-	-	L	M	
664	//	FMLA		0	1	0	1	1	1	1	1	1	x	L	M	
665	//	FMLS		0	1	0	1	1	1	1	1	1	x	L	M	
666	//	FMUL		0	1	0	1	1	1	1	1	1	x	L	M	
667	//	FMULX		0	1	1	1	1	1	1	1	1	x	L	M	
668	//	**AdvSIMD scalar shift by immediate**		**0**	**1**	**U**	**1**	**1**	**1**	**1**	**1**	**0**		**immh**		
669	//	SSHR	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
670	//	SSRA	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
671	//	SRSHR	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
672	//	SRSRA	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
673	//	SHL	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
674	//	SQSHL	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
675	//	SQSHRN	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
676	//	SQSHRN2	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
677	//	SQRSHRN	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
678	//	SQRSHRN2	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
679	//	SCVTF	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
680	//	FCVTZS	immh != 0000	0	1	0	1	1	1	1	1	0		immh		
681	//	USHR	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
682	//	USRA	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
683	//	URSHR	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
684	//	URSRA	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
685	//	SRI	immh != 0000	0	1	1	1	1	1	1	1	0		immh		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
686	//	SLI	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
687	//	SQSHLU	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
688	//	UQSHL	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
689	//	SQSHRUN	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
690	//	SQSHRUN2	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
691	//	SQRSHRUN	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
692	//	SQRSHRUN2	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
693	//	UQSHRN	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
694	//	UQRSHRN	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
695	//	UQRSHRN2	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
696	//	UCVTF	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
697	//	FCVTZU	immh != 0000	0	1	1	1	1	1	1	1	0		immh		
698	//	**Crypto three-reg SHA**		**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**size**		**0**		
699	//	SHA1C		0	1	0	1	1	1	1	0	0	0	0		
700	//	SHA1P		0	1	0	1	1	1	1	0	0	0	0		
701	//	SHA1M		0	1	0	1	1	1	1	0	0	0	0		
702	//	SHA1SU0		0	1	0	1	1	1	1	0	0	0	0		
703	//	SHA256H		0	1	0	1	1	1	1	0	0	0	0		
704	//	SHA256H2		0	1	0	1	1	1	1	0	0	0	0		
705	//	SHA256SU1		0	1	0	1	1	1	1	0	0	0	0		
706	//	**Crypto two-reg SHA**		**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**size**		**1**	**0**	**1**
707	//	SHA1H		0	1	0	1	1	1	1	0	0	0	1	0	1
708	//	SHA1SU1		0	1	0	1	1	1	1	0	0	0	1	0	1
709	//	SHA256SU0		0	1	0	1	1	1	1	0	0	0	1	0	1
710	//	**Crypto AES**		**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**size**		**1**	**0**	**1**
711	//	AESE		0	1	0	0	1	1	1	0	0	0	1	0	1
712	//	AESD		0	1	0	0	1	1	1	0	0	0	1	0	1
713	//	AESMC		0	1	0	0	1	1	1	0	0	0	1	0	1
714	//	AESIMC		0	1	0	0	1	1	1	0	0	0	1	0	1
715	//	**AdvSIMD three same**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**size**		**1**		
716	//	SHADD		0	Q	0	0	1	1	1	0	-	-	1		
717	//	SQADD		0	Q	0	0	1	1	1	0	-	-	1		
718	//	SRHADD		0	Q	0	0	1	1	1	0	-	-	1		
719	//	SHSUB		0	Q	0	0	1	1	1	0	-	-	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
720	//	SQSUB		0	Q	0	0	1	1	1	0	-	-	1		
721	//	CMGT		0	Q	0	0	1	1	1	0	-	-	1		
722	//	CMGE		0	Q	0	0	1	1	1	0	-	-	1		
723	//	SSHL Vector		0	Q	0	0	1	1	1	0	-	-	1		
724	//	SQSHL		0	Q	0	0	1	1	1	0	-	-	1		
725	//	SRSHL		0	Q	0	0	1	1	1	0	-	-	1		
726	//	SQRSHL		0	Q	0	0	1	1	1	0	-	-	1		
727	//	SMAX		0	Q	0	0	1	1	1	0	-	-	1		
728	//	SMIN		0	Q	0	0	1	1	1	0	-	-	1		
729	//	SABD		0	Q	0	0	1	1	1	0	-	-	1		
730	//	SABA		0	Q	0	0	1	1	1	0	-	-	1		
731	//	ADD		0	Q	0	0	1	1	1	0	-	-	1		
732	//	CMTST		0	Q	0	0	1	1	1	0	-	-	1		
733	//	MLA		0	Q	0	0	1	1	1	0	-	-	1		
734	//	MUL		0	Q	0	0	1	1	1	0	-	-	1		
735	//	SMAXP		0	Q	0	0	1	1	1	0	-	-	1		
736	//	SMINP		0	Q	0	0	1	1	1	0	-	-	1		
737	//	SQDMULH		0	Q	0	0	1	1	1	0	-	-	1		
738	//	ADDP		0	Q	0	0	1	1	1	0	-	-	1		
739	//	FMAXNM		0	Q	0	0	1	1	1	0	0	x	1		
740	//	FMLA		0	Q	0	0	1	1	1	0	0	x	1		
741	//	FADD		0	Q	0	0	1	1	1	0	0	x	1		
742	//	FMULX		0	Q	0	0	1	1	1	0	0	x	1		
743	//	FCMEQ		0	Q	0	0	1	1	1	0	0	x	1		
744	//	FMAX		0	Q	0	0	1	1	1	0	0	x	1		
745	//	FRECPS		0	Q	0	0	1	1	1	0	0	x	1		
746	//	AND		0	Q	0	0	1	1	1	0	0	0	1		
747	//	BIC		0	Q	0	0	1	1	1	0	0	1	1		
748	//	FMINNM		0	Q	0	0	1	1	1	0	1	x	1		
749	//	FMLS		0	Q	0	0	1	1	1	0	1	x	1		
750	//	FSUB		0	Q	0	0	1	1	1	0	1	x	1		
751	//	FMIN		0	Q	0	0	1	1	1	0	1	x	1		
752	//	FRSQRTS		0	Q	0	0	1	1	1	0	1	x	1		
753	//	ORR		0	Q	0	0	1	1	1	0	1	0	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
754	//	ORN		0	Q	0	0	1	1	1	0	1	1	1		
755	//	UHADD		0	Q	1	0	1	1	1	0	-	-	1		
756	//	UQADD		0	Q	1	0	1	1	1	0	-	-	1		
757	//	URHADD		0	Q	1	0	1	1	1	0	-	-	1		
758	//	UHSUB		0	Q	1	0	1	1	1	0	-	-	1		
759	//			0	Q	1	0	1	1	1	0	-	-	1		
760	//	CMHI		0	Q	1	0	1	1	1	0	-	-	1		
761	//	CMHS		0	Q	1	0	1	1	1	0	-	-	1		
762	//	USHL		0	Q	1	0	1	1	1	0	-	-	1		
763	//	UQSHL		0	Q	1	0	1	1	1	0	-	-	1		
764	//	URSHL		0	Q	1	0	1	1	1	0	-	-	1		
765	//	UQRSHL		0	Q	1	0	1	1	1	0	-	-	1		
766	//	UMAX		0	Q	1	0	1	1	1	0	-	-	1		
767	//	UMIN		0	Q	1	0	1	1	1	0	-	-	1		
768	//	UABD		0	Q	1	0	1	1	1	0	-	-	1		
769	//	UABA		0	Q	1	0	1	1	1	0	-	-	1		
770	//	SUB		0	Q	1	0	1	1	1	0	-	-	1		
771	//	CMEQ		0	Q	1	0	1	1	1	0	-	-	1		
772	//	MLS		0	Q	1	0	1	1	1	0	-	-	1		
773	//	PMUL		0	Q	1	0	1	1	1	0	-	-	1		
774	//	UMAXP		0	Q	1	0	1	1	1	0	-	-	1		
775	//	UMINP		0	Q	1	0	1	1	1	0	-	-	1		
776	//	SQRDMULH		0	Q	1	0	1	1	1	0	-	-	1		
777	//	FMAXNMP		0	Q	1	0	1	1	1	0	0	x	1		
778	//	FADDP		0	Q	1	0	1	1	1	0	0	x	1		
779	//	FMUL		0	Q	1	0	1	1	1	0	0	x	1		
780	//	FCMGE		0	Q	1	0	1	1	1	0	0	x	1		
781	//	FACGE		0	Q	1	0	1	1	1	0	0	x	1		
782	//	FMAXP		0	Q	1	0	1	1	1	0	0	x	1		
783	//	FDIV		0	Q	1	0	1	1	1	0	0	x	1		
784	//	EOR		0	Q	1	0	1	1	1	0	0	0	1		
785	//	BSL		0	Q	1	0	1	1	1	0	0	1	1		
786	//	FMINNMP		0	Q	1	0	1	1	1	0	1	x	1		
787	//	FABD		0	Q	1	0	1	1	1	0	1	x	1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
788	//	FCMGT		0	Q	1	0	1	1	1	0	1	x	1		
789	//	FACGT		0	Q	1	0	1	1	1	0	1	x	1		
790	//	FMINP		0	Q	1	0	1	1	1	0	1	x	1		
791	//	BIT		0	Q	1	0	1	1	1	0	1	0	1		
792	//	BIF		0	Q	1	0	1	1	1	0	1	1	1		
793	//	**AdvSIMD three different**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**size**		**1**		
794	//	SADDL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
795	//	SADDL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
796	//	SADDW	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
797	//	SADDW2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
798	//	SSUBL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
799	//	SSUBL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
800	//	SSUBW	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
801	//	SSUBW2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
802	//	ADDHN	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
803	//	ADDHN2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
804	//	SABAL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
805	//	SABAL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
806	//	SUBHN	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
807	//	SUBHN2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
808	//	SABDL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
809	//	SABDL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
810	//	SMLAL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
811	//	SMLAL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
812	//	SQDMLAL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
813	//	SQDMLAL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
814	//	SMLSL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
815	//	SMLSL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
816	//	SQDMLSL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
817	//	SQDMLSL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
818	//	SMULL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
819	//	SMULL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
820	//	SQDMULL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
821	//	SQDMULL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
822	//	PMULL	writes to low half of the dest. register	0	0	0	0	1	1	1	0	size		1		
823	//	PMULL2	writes to high half of the dest. registe	0	1	0	0	1	1	1	0	size		1		
824	//	UADDL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
825	//	UADDL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
826	//	UADDW	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
827	//	UADDW2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
828	//	USUBL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
829	//	USUBL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
830	//	USUBW	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
831	//	USUBW2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
832	//	RADDHN	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
833	//	RADDHN2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
834	//	UABAL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
835	//	UABAL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
836	//	RSUBHN	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
837	//	RSUBHN2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
838	//	UABDL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
839	//	UABDL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
840	//	UMLAL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
841	//	UMLAL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
842	//	UMLSL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
843	//	UMLSL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
844	//	UMULL	writes to low half of the dest. register	0	0	1	0	1	1	1	0	size		1		
845	//	UMULL2	writes to high half of the dest. registe	0	1	1	0	1	1	1	0	size		1		
846	//	**AdvSIMD two-reg misc**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**size**		**1**	**0**	**0**
847	//	REV64		0	Q	0	0	1	1	1	0	-	-	1	0	0
848	//	REV16		0	Q	0	0	1	1	1	0	-	-	1	0	0
849	//	SADDLP		0	Q	0	0	1	1	1	0	-	-	1	0	0
850	//	SUQADD		0	Q	0	0	1	1	1	0	-	-	1	0	0
851	//	CLS		0	Q	0	0	1	1	1	0	-	-	1	0	0
852	//	CNT		0	Q	0	0	1	1	1	0	-	-	1	0	0
853	//	SADALP		0	Q	0	0	1	1	1	0	-	-	1	0	0
854	//	SQABS		0	Q	0	0	1	1	1	0	-	-	1	0	0
855	//	CMGT		0	Q	0	0	1	1	1	0	-	-	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
856	//	CMEQ		0	Q	0	0	1	1	1	0	-	-	1	0	0
857	//	CMLT		0	Q	0	0	1	1	1	0	-	-	1	0	0
858	//	ABS		0	Q	0	0	1	1	1	0	-	-	1	0	0
859	//	XTN		0	Q	0	0	1	1	1	0	-	-	1	0	0
860	//	XTN2		0	Q	0	0	1	1	1	0	-	-	1	0	0
861	//	SQXTN		0	Q	0	0	1	1	1	0	-	-	1	0	0
862	//	SQXTN2		0	Q	0	0	1	1	1	0	-	-	1	0	0
863	//	FCVTN		0	Q	0	0	1	1	1	0	0	x	1	0	0
864	//	FCVTN2		0	Q	0	0	1	1	1	0	0	x	1	0	0
865	//	FCVTL		0	Q	0	0	1	1	1	0	0	x	1	0	0
866	//	FCVTL2		0	Q	0	0	1	1	1	0	0	x	1	0	0
867	//	FRINTN		0	Q	0	0	1	1	1	0	0	x	1	0	0
868	//	FRINTM		0	Q	0	0	1	1	1	0	0	x	1	0	0
869	//	FCVTNS		0	Q	0	0	1	1	1	0	0	x	1	0	0
870	//	FCVTMS		0	Q	0	0	1	1	1	0	0	x	1	0	0
871	//	FCVTAS		0	Q	0	0	1	1	1	0	0	x	1	0	0
872	//	SCVTF		0	Q	0	0	1	1	1	0	0	x	1	0	0
873	//	FCMGT		0	Q	0	0	1	1	1	0	1	x	1	0	0
874	//	FCMEQ		0	Q	0	0	1	1	1	0	1	x	1	0	0
875	//	FCMLT		0	Q	0	0	1	1	1	0	1	x	1	0	0
876	//	FABS		0	Q	0	0	1	1	1	0	1	x	1	0	0
877	//	FRINTP		0	Q	0	0	1	1	1	0	1	x	1	0	0
878	//	FRINTZ		0	Q	0	0	1	1	1	0	1	x	1	0	0
879	//	FCVTPS		0	Q	0	0	1	1	1	0	1	x	1	0	0
880	//	FCVTZS		0	Q	0	0	1	1	1	0	1	x	1	0	0
881	//	URECPE		0	Q	0	0	1	1	1	0	1	x	1	0	0
882	//	FRECPE		0	Q	0	0	1	1	1	0	1	x	1	0	0
883	//	REV32		0	Q	1	0	1	1	1	0	-	-	1	0	0
884	//	UADDLP		0	Q	1	0	1	1	1	0	-	-	1	0	0
885	//	USQADD		0	Q	1	0	1	1	1	0	-	-	1	0	0
886	//	CLZ		0	Q	1	0	1	1	1	0	-	-	1	0	0
887	//	UADALP		0	Q	1	0	1	1	1	0	-	-	1	0	0
888	//	SQNEG		0	Q	1	0	1	1	1	0	-	-	1	0	0
889	//	CMGE		0	Q	1	0	1	1	1	0	-	-	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
890	//	CMLE		0	Q	1	0	1	1	1	0	-	-	1	0	0
891	//	NEG		0	Q	1	0	1	1	1	0	-	-	1	0	0
892	//	SQXTUN		0	Q	1	0	1	1	1	0	-	-	1	0	0
893	//	SQXTUN2		0	Q	1	0	1	1	1	0	-	-	1	0	0
894	//	SHLL		0	Q	1	0	1	1	1	0	-	-	1	0	0
895	//	SHLL2		0	Q	1	0	1	1	1	0	-	-	1	0	0
896	//	UQXTN		0	Q	1	0	1	1	1	0	-	-	1	0	0
897	//	UQXTN2		0	Q	1	0	1	1	1	0	-	-	1	0	0
898	//	FCVTXN		0	Q	1	0	1	1	1	0	0	x	1	0	0
899	//	FCVTXN2		0	Q	1	0	1	1	1	0	0	x	1	0	0
900	//	FRINTA		0	Q	1	0	1	1	1	0	0	x	1	0	0
901	//	FRINTX		0	Q	1	0	1	1	1	0	0	x	1	0	0
902	//	FCVTNU		0	Q	1	0	1	1	1	0	0	x	1	0	0
903	//	FCVTMU		0	Q	1	0	1	1	1	0	0	x	1	0	0
904	//	FCVTAU		0	Q	1	0	1	1	1	0	0	x	1	0	0
905	//	UCVTF		0	Q	1	0	1	1	1	0	0	x	1	0	0
906	//	NOT		0	Q	1	0	1	1	1	0	0	0	1	0	0
907	//	RBIT		0	Q	1	0	1	1	1	0	0	1	1	0	0
908	//	FCMGE		0	Q	1	0	1	1	1	0	1	x	1	0	0
909	//	FCMLE		0	Q	1	0	1	1	1	0	1	x	1	0	0
910	//	FNEG		0	Q	1	0	1	1	1	0	1	x	1	0	0
911	//	FRINTI		0	Q	1	0	1	1	1	0	1	x	1	0	0
912	//	FCVTPU		0	Q	1	0	1	1	1	0	1	x	1	0	0
913	//	FCVTZU		0	Q	1	0	1	1	1	0	1	x	1	0	0
914	//	URSQRTE		0	Q	1	0	1	1	1	0	1	x	1	0	0
915	//	FRSQRTE		0	Q	1	0	1	1	1	0	1	x	1	0	0
916	//	FSQRT		0	Q	1	0	1	1	1	0	1	x	1	0	0
917	//	**AdvSIMD across lanes**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**0**	**size**		**1**	**1**	**0**
918	//	SADDLV		0	Q	0	0	1	1	1	0	-	-	1	1	0
919	//	SMAXV		0	Q	0	0	1	1	1	0	-	-	1	1	0
920	//	SMINV		0	Q	0	0	1	1	1	0	-	-	1	1	0
921	//	ADDV		0	Q	0	0	1	1	1	0	-	-	1	1	0
922	//	UADDLV		0	Q	1	0	1	1	1	0	-	-	1	1	0
923	//	UMAXV		0	Q	1	0	1	1	1	0	-	-	1	1	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
924	//	UMINV		0	Q	1	0	1	1	1	0	-	-	1	1	0
925	//	FMAXNMV		0	Q	1	0	1	1	1	0	0	x	1	1	0
926	//	FMAXV		0	Q	1	0	1	1	1	0	0	x	1	1	0
927	//	FMINNMV		0	Q	1	0	1	1	1	0	1	x	1	1	0
928	//	FMINV		0	Q	1	0	1	1	1	0	1	x	1	1	0
929	//	**AdvSIMD copy**		**0**	**Q**	**op**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**		**i**
930	//	DUP		0	-	0	0	1	1	1	0	0	0	0	-	-
931	//	DUP		0	-	0	0	1	1	1	0	0	0	0	-	-
932	//	SMOV		0	0	0	0	1	1	1	0	0	0	0	-	-
933	//	UMOV		0	0	0	0	1	1	1	0	0	0	0	-	-
934	//	INS		0	1	0	0	1	1	1	0	0	0	0	-	-
935	//	SMOV		0	1	0	0	1	1	1	0	0	0	0	-	-
936	//	UMOV		0	1	0	0	1	1	1	0	0	0	0	-	-
937	//	INS		0	1	1	0	1	1	1	0	0	0	0	-	-
938	//	**AdvSIMD vector x indexed element**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**1**	**size**		**L**	**M**	
939	//	SMLAL		0	Q	0	0	1	1	1	1	-	-	L	M	
940	//	SMLAL2		0	Q	0	0	1	1	1	1	-	-	L	M	
941	//	SQDMLAL		0	Q	0	0	1	1	1	1	-	-	L	M	
942	//	SQDMLAL2		0	Q	0	0	1	1	1	1	-	-	L	M	
943	//	SMLSL		0	Q	0	0	1	1	1	1	-	-	L	M	
944	//	SMLSL2		0	Q	0	0	1	1	1	1	-	-	L	M	
945	//	SQDMLSL		0	Q	0	0	1	1	1	1	-	-	L	M	
946	//	SQDMLSL2		0	Q	0	0	1	1	1	1	-	-	L	M	
947	//	MUL		0	Q	0	0	1	1	1	1	-	-	L	M	
948	//	SMULL		0	Q	0	0	1	1	1	1	-	-	L	M	
949	//	SMULL2		0	Q	0	0	1	1	1	1	-	-	L	M	
950	//	SQDMULL		0	Q	0	0	1	1	1	1	-	-	L	M	
951	//	SQDMULL2		0	Q	0	0	1	1	1	1	-	-	L	M	
952	//	SQDMULH		0	Q	0	0	1	1	1	1	-	-	L	M	
953	//	SQRDMULH		0	Q	0	0	1	1	1	1	-	-	L	M	
954	//	FMLA		0	Q	0	0	1	1	1	1	1	x	L	M	
955	//	FMLS		0	Q	0	0	1	1	1	1	1	x	L	M	
956	//	FMUL		0	Q	0	0	1	1	1	1	1	x	L	M	
957	//	MLA		0	Q	1	0	1	1	1	1	-	-	L	M	

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
958	//	UMLAL		0	Q	1	0	1	1	1	1	-	-	L	M	
959	//	UMLAL2		0	Q	1	0	1	1	1	1	-	-	L	M	
960	//	MLS		0	Q	1	0	1	1	1	1	-	-	L	M	
961	//	UMLSL		0	Q	1	0	1	1	1	1	-	-	L	M	
962	//	UMLSL2		0	Q	1	0	1	1	1	1	-	-	L	M	
963	//	UMULL		0	Q	1	0	1	1	1	1	-	-	L	M	
964	//	UMULL2		0	Q	1	0	1	1	1	1	-	-	L	M	
965	//	FMULX		0	Q	1	0	1	1	1	1	1	x	L	M	
966	//	**AdvSIMD modified immediate**		**0**	**Q**	**op**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
967	//	MOVI		0	-	0	0	1	1	1	1	0	0	0	0	0
968	//	ORR		0	-	0	0	1	1	1	1	0	0	0	0	0
969	//	MOVI		0	-	0	0	1	1	1	1	0	0	0	0	0
970	//	ORR		0	-	0	0	1	1	1	1	0	0	0	0	0
971	//	MOVI		0	-	0	0	1	1	1	1	0	0	0	0	0
972	//	MOVI		0	-	0	0	1	1	1	1	0	0	0	0	0
973	//	FMOV		0	-	0	0	1	1	1	1	0	0	0	0	0
974	//	MVNI		0	-	1	0	1	1	1	1	0	0	0	0	0
975	//	BIC		0	-	1	0	1	1	1	1	0	0	0	0	0
976	//	MVNI		0	-	1	0	1	1	1	1	0	0	0	0	0
977	//	BIC		0	-	1	0	1	1	1	1	0	0	0	0	0
978	//	MVNI		0	-	1	0	1	1	1	1	0	0	0	0	0
979	//	MOVI		0	0	1	0	1	1	1	1	0	0	0	0	0
980	//	MOVI		0	1	1	0	1	1	1	1	0	0	0	0	0
981	//	FMOV		0	1	1	0	1	1	1	1	0	0	0	0	0
982	//	**AdvSIMD shift by immediate**		**0**	**Q**	**U**	**0**	**1**	**1**	**1**	**1**	**0**		**immh**		
983	//	SSHR		0	Q	0	0	1	1	1	1	0		immh		
984	//	SSRA		0	Q	0	0	1	1	1	1	0		immh		
985	//	SRSHR		0	Q	0	0	1	1	1	1	0		immh		
986	//	SRSRA		0	Q	0	0	1	1	1	1	0		immh		
987	//	SHL		0	Q	0	0	1	1	1	1	0		immh		
988	//	SQSHL		0	Q	0	0	1	1	1	1	0		immh		
989	//	SHRN		0	Q	0	0	1	1	1	1	0		immh		
990	//	SHRN2		0	Q	0	0	1	1	1	1	0		immh		
991	//	RSHRN		0	Q	0	0	1	1	1	1	0		immh		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
992	//	RSHRN2		0	Q	0	0	1	1	1	1	0		immh		
993	//	SQSHRN		0	Q	0	0	1	1	1	1	0		immh		
994	//	SQSHRN2		0	Q	0	0	1	1	1	1	0		immh		
995	//	SQRSHRN		0	Q	0	0	1	1	1	1	0		immh		
996	//	SQRSHRN2		0	Q	0	0	1	1	1	1	0		immh		
997	//	SSHLL		0	Q	0	0	1	1	1	1	0		immh		
998	//	SSHLL2		0	Q	0	0	1	1	1	1	0		immh		
999	//	SCVTF		0	Q	0	0	1	1	1	1	0		immh		
1000	//	FCVTZS		0	Q	0	0	1	1	1	1	0		immh		
1001	//	USHR		0	Q	1	0	1	1	1	1	0		immh		
1002	//	USRA		0	Q	1	0	1	1	1	1	0		immh		
1003	//	URSHR		0	Q	1	0	1	1	1	1	0		immh		
1004	//	URSRA		0	Q	1	0	1	1	1	1	0		immh		
1005	//	SRI		0	Q	1	0	1	1	1	1	0		immh		
1006	//	SLI		0	Q	1	0	1	1	1	1	0		immh		
1007	//	SQSHLU		0	Q	1	0	1	1	1	1	0		immh		
1008	//	UQSHL		0	Q	1	0	1	1	1	1	0		immh		
1009	//	SQSHRUN		0	Q	1	0	1	1	1	1	0		immh		
1010	//	SQSHRUN2		0	Q	1	0	1	1	1	1	0		immh		
1011	//	SQRSHRUN		0	Q	1	0	1	1	1	1	0		immh		
1012	//	SQRSHRUN2		0	Q	1	0	1	1	1	1	0		immh		
1013	//	UQSHRN		0	Q	1	0	1	1	1	1	0		immh		
1014	//	UQRSHRN		0	Q	1	0	1	1	1	1	0		immh		
1015	//	UQRSHRN2		0	Q	1	0	1	1	1	1	0		immh		
1016	//	USHLL		0	Q	1	0	1	1	1	1	0		immh		
1017	//	USHLL2		0	Q	1	0	1	1	1	1	0		immh		
1018	//	UCVTF		0	Q	1	0	1	1	1	1	0		immh		
1019	//	FCVTZU		0	Q	1	0	1	1	1	1	0		immh		
1020	//	**AdvSIMD TBL/TBX**		**0**	**Q**	**0**	**0**	**1**	**1**	**1**	**0**	**op2**		**0**		
1021	//	TBL		0	Q	0	0	1	1	1	0	0	0	0		
1022	//	TBX		0	Q	0	0	1	1	1	0	0	0	0		
1023	//	TBL		0	Q	0	0	1	1	1	0	0	0	0		
1024	//	TBX		0	Q	0	0	1	1	1	0	0	0	0		
1025	//	TBL		0	Q	0	0	1	1	1	0	0	0	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
1026	//	TBX		0	Q	0	0	1	1	1	0	0	0	0		
1027	//	TBL		0	Q	0	0	1	1	1	0	0	0	0		
1028	//	TBX		0	Q	0	0	1	1	1	0	0	0	0		
1029	//	**AdvSIMD ZIP/UZP/TRN**		**0**	**Q**	**0**	**0**	**1**	**1**	**1**	**0**	**size**		**0**		
1030	//	UZP1		0	Q	0	0	1	1	1	0	size		0		
1031	//	TRN1		0	Q	0	0	1	1	1	0	size		0		
1032	//	ZIP1		0	Q	0	0	1	1	1	0	size		0		
1033	//	UZP2		0	Q	0	0	1	1	1	0	size		0		
1034	//	TRN2		0	Q	0	0	1	1	1	0	size		0		
1035	//	ZIP2		0	Q	0	0	1	1	1	0	size		0		
1036	//	**AdvSIMD EXT**		**0**	**Q**	**1**	**0**	**1**	**1**	**1**	**0**	**op2**		**0**		
1037	//	EXT		0	Q	1	0	1	1	1	0	0	0	0		
1038	//	**Loads and stores**						**1**		**0**						
1039	//	**AdvSIMD load/store multiple structures**		**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**L**	**0**	**0**	**0**
1040	//	ST4		0	Q	0	0	1	1	0	0	0	0	0	0	0
1041	//	ST1		0	Q	0	0	1	1	0	0	0	0	0	0	0
1042	//	ST3		0	Q	0	0	1	1	0	0	0	0	0	0	0
1043	//	ST1		0	Q	0	0	1	1	0	0	0	0	0	0	0
1044	//	ST1		0	Q	0	0	1	1	0	0	0	0	0	0	0
1045	//	ST2		0	Q	0	0	1	1	0	0	0	0	0	0	0
1046	//	ST1		0	Q	0	0	1	1	0	0	0	0	0	0	0
1047	//	LD4		0	Q	0	0	1	1	0	0	0	1	0	0	0
1048	//	LD1		0	Q	0	0	1	1	0	0	0	1	0	0	0
1049	//	LD3		0	Q	0	0	1	1	0	0	0	1	0	0	0
1050	//	LD1		0	Q	0	0	1	1	0	0	0	1	0	0	0
1051	//	LD1		0	Q	0	0	1	1	0	0	0	1	0	0	0
1052	//	LD2		0	Q	0	0	1	1	0	0	0	1	0	0	0
1053	//	LD1		0	Q	0	0	1	1	0	0	0	1	0	0	0
1054	//	**AdvSIMD load/store multiple structures (pc**		**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**L**	**0**		
1055	//	ST4	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
1056	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
1057	//	ST3	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
1058	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
1059	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
106(	//	ST2	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
1061	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	0	1	0	0		
106:	//	ST4		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	ST1		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	ST3		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	ST1		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	ST1		0	Q	0	0	1	1	0	0	1	0	0	1	1
106;	//	ST2		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	ST1		0	Q	0	0	1	1	0	0	1	0	0	1	1
106:	//	LD4	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107(	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
1071	//	LD3	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107:	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107:	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107:	//	LD2	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107;	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	0	1	1	0		
107:	//	LD4		0	Q	0	0	1	1	0	0	1	1	0	1	1
1077	//	LD1		0	Q	0	0	1	1	0	0	1	1	0	1	1
107:	//	LD3		0	Q	0	0	1	1	0	0	1	1	0	1	1
107;	//	LD1		0	Q	0	0	1	1	0	0	1	1	0	1	1
108(	//	LD1		0	Q	0	0	1	1	0	0	1	1	0	1	1
1081	//	LD2		0	Q	0	0	1	1	0	0	1	1	0	1	1
108:	//	LD1		0	Q	0	0	1	1	0	0	1	1	0	1	1
108:	//	**AdvSIMD load/store single structure**		**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**L**	**R**	**0**	**0**
108:	//	ST1		0	Q	0	0	1	1	0	1	0	0	0	0	0
108;	//	ST3		0	Q	0	0	1	1	0	1	0	0	0	0	0
108:	//	ST1		0	Q	0	0	1	1	0	1	0	0	0	0	0
1087	//	ST3		0	Q	0	0	1	1	0	1	0	0	0	0	0
108:	//	ST1		0	Q	0	0	1	1	0	1	0	0	0	0	0
108;	//	ST1		0	Q	0	0	1	1	0	1	0	0	0	0	0
109(	//	ST3		0	Q	0	0	1	1	0	1	0	0	0	0	0
1091	//	ST3		0	Q	0	0	1	1	0	1	0	0	0	0	0
109:	//	ST2		0	Q	0	0	1	1	0	1	0	0	1	0	0
109:	//	ST4		0	Q	0	0	1	1	0	1	0	0	1	0	0

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
1094	//	ST2		0	Q	0	0	1	1	0	1	0	0	1	0	0
1095	//	ST4		0	Q	0	0	1	1	0	1	0	0	1	0	0
1096	//	ST2		0	Q	0	0	1	1	0	1	0	0	1	0	0
1097	//	ST2		0	Q	0	0	1	1	0	1	0	0	1	0	0
1098	//	ST4		0	Q	0	0	1	1	0	1	0	0	1	0	0
1099	//	ST4		0	Q	0	0	1	1	0	1	0	0	1	0	0
1100	//	LD1		0	Q	0	0	1	1	0	1	0	1	0	0	0
1101	//	LD3		0	Q	0	0	1	1	0	1	0	1	0	0	0
1102	//	LD1		0	Q	0	0	1	1	0	1	0	1	0	0	0
1103	//	LD3		0	Q	0	0	1	1	0	1	0	1	0	0	0
1104	//	LD1		0	Q	0	0	1	1	0	1	0	1	0	0	0
1105	//	LD1		0	Q	0	0	1	1	0	1	0	1	0	0	0
1106	//	LD3		0	Q	0	0	1	1	0	1	0	1	0	0	0
1107	//	LD3		0	Q	0	0	1	1	0	1	0	1	0	0	0
1108	//	LD1R		0	Q	0	0	1	1	0	1	0	1	0	0	0
1109	//	LD3R		0	Q	0	0	1	1	0	1	0	1	0	0	0
1110	//	LD2		0	Q	0	0	1	1	0	1	0	1	1	0	0
1111	//	LD4		0	Q	0	0	1	1	0	1	0	1	1	0	0
1112	//	LD2		0	Q	0	0	1	1	0	1	0	1	1	0	0
1113	//	LD4		0	Q	0	0	1	1	0	1	0	1	1	0	0
1114	//	LD2		0	Q	0	0	1	1	0	1	0	1	1	0	0
1115	//	LD2		0	Q	0	0	1	1	0	1	0	1	1	0	0
1116	//	LD4		0	Q	0	0	1	1	0	1	0	1	1	0	0
1117	//	LD4		0	Q	0	0	1	1	0	1	0	1	1	0	0
1118	//	LD2R		0	Q	0	0	1	1	0	1	0	1	1	0	0
1119	//	LD4R		0	Q	0	0	1	1	0	1	0	1	1	0	0
1120	//	**AdvSIMD load/store single structure (post-**		**0**	**Q**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**L**	**R**		
1121	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1122	//	ST3	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1123	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1124	//	ST3	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1125	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1126	//	ST1	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1127	//	ST3	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
1128	//	ST3	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	0		
1129	//	ST1		0	Q	0	0	1	1	0	1	1	0	0	1	1
1130	//	ST3		0	Q	0	0	1	1	0	1	1	0	0	1	1
1131	//	ST1		0	Q	0	0	1	1	0	1	1	0	0	1	1
1132	//	ST3		0	Q	0	0	1	1	0	1	1	0	0	1	1
1133	//	ST1		0	Q	0	0	1	1	0	1	1	0	0	1	1
1134	//	ST1		0	Q	0	0	1	1	0	1	1	0	0	1	1
1135	//	ST3		0	Q	0	0	1	1	0	1	1	0	0	1	1
1136	//	ST3		0	Q	0	0	1	1	0	1	1	0	0	1	1
1137	//	ST2	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1138	//	ST4	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1139	//	ST2	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1140	//	ST4	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1141	//	ST2	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1142	//	ST2	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1143	//	ST4	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1144	//	ST4	Rm != 11111	0	Q	0	0	1	1	0	1	1	0	1		
1145	//	ST2		0	Q	0	0	1	1	0	1	1	0	1	1	1
1146	//	ST4		0	Q	0	0	1	1	0	1	1	0	1	1	1
1147	//	ST2		0	Q	0	0	1	1	0	1	1	0	1	1	1
1148	//	ST4		0	Q	0	0	1	1	0	1	1	0	1	1	1
1149	//	ST2		0	Q	0	0	1	1	0	1	1	0	1	1	1
1150	//	ST2		0	Q	0	0	1	1	0	1	1	0	1	1	1
1151	//	ST4		0	Q	0	0	1	1	0	1	1	0	1	1	1
1152	//	ST4		0	Q	0	0	1	1	0	1	1	0	1	1	1
1153	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1154	//	LD3	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1155	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1156	//	LD3	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1157	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1158	//	LD1	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1159	//	LD3	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1160	//	LD3	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1161	//	LD1R	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		

1	in_use	Opcode	comments	31	30	29	28	27	26	25	24	23	22	21	20	19
1162	//	LD3R	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	0		
1163	//	LD1		0	Q	0	0	1	1	0	1	1	1	0	1	1
1164	//	LD3		0	Q	0	0	1	1	0	1	1	1	0	1	1
1165	//	LD1		0	Q	0	0	1	1	0	1	1	1	0	1	1
1166	//	LD3		0	Q	0	0	1	1	0	1	1	1	0	1	1
1167	//	LD1		0	Q	0	0	1	1	0	1	1	1	0	1	1
1168	//	LD1		0	Q	0	0	1	1	0	1	1	1	0	1	1
1169	//	LD3		0	Q	0	0	1	1	0	1	1	1	0	1	1
1170	//	LD3		0	Q	0	0	1	1	0	1	1	1	0	1	1
1171	//	LD1R		0	Q	0	0	1	1	0	1	1	1	0	1	1
1172	//	LD3R		0	Q	0	0	1	1	0	1	1	1	0	1	1
1173	//	LD2	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1174	//	LD4	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1175	//	LD2	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1176	//	LD4	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1177	//	LD2	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1178	//	LD2	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1179	//	LD4	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1180	//	LD4	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1181	//	LD2R	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1182	//	LD4R	Rm != 11111	0	Q	0	0	1	1	0	1	1	1	1		
1183	//	LD2		0	Q	0	0	1	1	0	1	1	1	1	1	1
1184	//	LD4		0	Q	0	0	1	1	0	1	1	1	1	1	1
1185	//	LD2		0	Q	0	0	1	1	0	1	1	1	1	1	1
1186	//	LD4		0	Q	0	0	1	1	0	1	1	1	1	1	1
1187	//	LD2		0	Q	0	0	1	1	0	1	1	1	1	1	1
1188	//	LD2		0	Q	0	0	1	1	0	1	1	1	1	1	1
1189	//	LD4		0	Q	0	0	1	1	0	1	1	1	1	1	1
1190	//	LD4		0	Q	0	0	1	1	0	1	1	1	1	1	1
1191	//	LD2R		0	Q	0	0	1	1	0	1	1	1	1	1	1
1192	//	LD4R		0	Q	0	0	1	1	0	1	1	1	1	1	1

in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
	UNALLOCATED																				
	BAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00000000
	Branch,exception generation and syst																				
	Compare _ Branch (immediate)				**imm19**													**Rt**			
	CBZ				imm19													Rt			0x34000000
	CBNZ				imm19													Rt			0x35000000
	CBZ				imm19													Rt			0xB4000000
	CBNZ				imm19													Rt			0xB5000000
	Test bit & branch (immediate)							**imm14**										**Rt**			
	TBZ							imm14										Rt			0x36000000
	TBNZ							imm14										Rt			0x37000000
	Conditional branch (immediate)				**imm19**											**-**		**cond**			
	B_cond				imm19											0		cond			0x54000000
	Exception generation						**imm16**									**-**	**-**	**-**	**-**	**-**	
//	SVC						imm16									0	0	0	0	1	0xD4000001
//	HVC						imm16									0	0	0	1	0	0xD4000002
//	SMC						imm16									0	0	0	1	1	0xD4000003
	BRK						imm16									0	0	0	0	0	0xD4200000
//	HLT						imm16									0	0	0	0	0	0xD4400000
//	DCPS1						imm16									0	0	0	0	1	0xD4A00001
//	DCPS2						imm16									0	0	0	1	0	0xD4A00002
//	DCPS3						imm16									0	0	0	1	1	0xD4A00003
//	**System**			**op1**			**CRn**			**CRm**			**op2**					**Rt**			
//	MSR			op1		0	1	0	0	CR_m			op2			1	1	1	1	1	0xD500401F
//	HINT	0	1	1	0	0	1	0		CR_m			op2			1	1	1	1	1	0xD503201F
//	CLREX	0	1	1	0	0	1	1		CR_m		0	1	0		1	1	1	1	1	0xD503305F
//	DSB	0	1	1	0	0	1	1		CR_m		1	0	0		1	1	1	1	1	0xD503309F
//	DMB	0	1	1	0	0	1	1		CR_m		1	0	1		1	1	1	1	1	0xD50330BF
//	ISB	0	1	1	0	0	1	1		CR_m		1	1	0		1	1	1	1	1	0xD50330DF
//	SYS			op1			CR_n			CR_m			op2					Rt			0xD5080000
//	MSR			op1			CR_n			CR_m			op2					Rt			0xD5100000
//	SYSL			op1			CR_n			CR_m			op2					Rt			0xD5280000
//	MRS			op1			CR_n			CR_m			op2					Rt			0xD5300000
	Unconditional branch (register)	**op2**						**op3**					**Rn**					**op4**			
	BR	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0	0xD61F0000
	BLR	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0	0xD63F0000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
38		RET	1	1	1	0	0	0	0	0	0			Rn			0	0	0	0	0	0xD65F0000
39	//	ERET	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0xD69F03E0
40	//	DRPS	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0xD6BF03E0
41	//	**Unconditional branch (immediate)**						**imm26**														
42	//	B						imm26														0x14000000
43	//	BL						imm26														0x94000000
44		**Loads and stores**																				
45		**Load/store exclusive**	**Rs**			**-**		**Rt2**						**Rn**					**Rt**			
46		STXRB	Rs			0		Rt2						Rn					Rt			0x08000000
47		STLXRB	Rs			1		Rt2						Rn					Rt			0x08008000
48		LDXRB	Rs			0		Rt2						Rn					Rt			0x08400000
49		LDAXRB	Rs			1		Rt2						Rn					Rt			0x08408000
50		STLRB	Rs			1		Rt2						Rn					Rt			0x08808000
51		LDARB	Rs			1		Rt2						Rn					Rt			0x08C08000
52		STXRH	Rs			0		Rt2						Rn					Rt			0x48000000
53		STLXRH	Rs			1		Rt2						Rn					Rt			0x48008000
54		LDXRH	Rs			0		Rt2						Rn					Rt			0x48400000
55		LDAXRH	Rs			1		Rt2						Rn					Rt			0x48408000
56		STLRH	Rs			1		Rt2						Rn					Rt			0x48808000
57		LDARH	Rs			1		Rt2						Rn					Rt			0x48C08000
58		STXR	Rs			0		Rt2						Rn					Rt			0x88000000
59		STLXR	Rs			1		Rt2						Rn					Rt			0x88008000
60		STXP	Rs			0		Rt2						Rn					Rt			0x88200000
61		STLXP	Rs			1		Rt2						Rn					Rt			0x88208000
62		LDXR	Rs			0		Rt2						Rn					Rt			0x88400000
63		LDAXR	Rs			1		Rt2						Rn					Rt			0x88408000
64		LDXP	Rs			0		Rt2						Rn					Rt			0x88600000
65		LDAXP	Rs			1		Rt2						Rn					Rt			0x88608000
66		STLR	Rs			1		Rt2						Rn					Rt			0x88808000
67		LDAR	Rs			1		Rt2						Rn					Rt			0x88C08000
68		STXR	Rs			0		Rt2						Rn					Rt			0xC8000000
69		STLXR	Rs			1		Rt2						Rn					Rt			0xC8008000
70		STXP	Rs			0		Rt2						Rn					Rt			0xC8200000
71		STLXP	Rs			1		Rt2						Rn					Rt			0xC8208000
72		LDXR	Rs			0		Rt2						Rn					Rt			0xC8400000
73		LDAXR	Rs			1		Rt2						Rn					Rt			0xC8408000
74		LDXP	Rs			0		Rt2						Rn					Rt			0xC8600000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
75		LDAXP	Rs			1			Rt2					Rn					Rt			0xC8608000
76		STLR	Rs			1			Rt2					Rn					Rt			0xC8808000
77		LDAR	Rs			1			Rt2					Rn					Rt			0xC8C08000
78		**Load register (literal)**				**imm19**													**Rt**			
79		LDR				imm19													Rt			0x18000000
80		LDR				imm19													Rt			0x1C000000
81		LDR				imm19													Rt			0x58000000
82		LDR				imm19													Rt			0x5C000000
83		LDRSW				imm19													Rt			0x98000000
84		LDR				imm19													Rt			0x9C000000
85		PRFM				imm19													Rt			0xD8000000
86		**Load/store no-allocate pair (offset)**	**mm7**						**Rt2**					**Rn**					**Rt**			
87		STNP	mm7						Rt2					Rn					Rt			0x28000000
88		LDNP	mm7						Rt2					Rn					Rt			0x28400000
89		STNP	mm7						Rt2					Rn					Rt			0x2C000000
90		LDNP	mm7						Rt2					Rn					Rt			0x2C400000
91		STNP	mm7						Rt2					Rn					Rt			0x6C000000
92		LDNP	mm7						Rt2					Rn					Rt			0x6C400000
93		STNP	mm7						Rt2					Rn					Rt			0xA8000000
94		LDNP	mm7						Rt2					Rn					Rt			0xA8400000
95		STNP	mm7						Rt2					Rn					Rt			0xAC000000
96		LDNP	mm7						Rt2					Rn					Rt			0xAC400000
97		**Load/store register pair (post-indexed)**	**mm7**						**Rt2**					**Rn**					**Rt**			
98		STP	mm7						Rt2					Rn					Rt			0x28800000
99		LDP	mm7						Rt2					Rn					Rt			0x28C00000
100		STP	mm7						Rt2					Rn					Rt			0x2C800000
101		LDP	mm7						Rt2					Rn					Rt			0x2CC00000
102		LDPSW	mm7						Rt2					Rn					Rt			0x68C00000
103		STP	mm7						Rt2					Rn					Rt			0x6C800000
104		LDP	mm7						Rt2					Rn					Rt			0x6CC00000
105		STP	mm7						Rt2					Rn					Rt			0xA8800000
106		LDP	mm7						Rt2					Rn					Rt			0xA8C00000
107		STP	mm7						Rt2					Rn					Rt			0xAC800000
108		LDP	mm7						Rt2					Rn					Rt			0xACC00000
109		**Load/store register pair (offset)**	**mm7**						**Rt2**					**Rn**					**Rt**			

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
110		STP	mm7						Rt2					Rn					Rt			0x29000000
111		LDP	mm7						Rt2					Rn					Rt			0x29400000
112		STP	mm7						Rt2					Rn					Rt			0x2D000000
113		LDP	mm7						Rt2					Rn					Rt			0x2D400000
114		LDPSW	mm7						Rt2					Rn					Rt			0x69400000
115		STP	mm7						Rt2					Rn					Rt			0x6D000000
116		LDP	mm7						Rt2					Rn					Rt			0x6D400000
117		STP	mm7						Rt2					Rn					Rt			0xA9000000
118		LDP	mm7						Rt2					Rn					Rt			0xA9400000
119		STP	mm7						Rt2					Rn					Rt			0xAD000000
120		LDP	mm7						Rt2					Rn					Rt			0xAD400000
121		**Load/store register pair (pre-indexed)**	**mm7**						**Rt2**					**Rn**					**Rt**			
122		STP	mm7						Rt2					Rn					Rt			0x29800000
123		LDP	mm7						Rt2					Rn					Rt			0x29C00000
124		STP	mm7						Rt2					Rn					Rt			0x2D800000
125		LDP	mm7						Rt2					Rn					Rt			0x2DC00000
126		LDPSW	mm7						Rt2					Rn					Rt			0x69C00000
127		STP	mm7						Rt2					Rn					Rt			0x6D800000
128		LDP	mm7						Rt2					Rn					Rt			0x6DC00000
129		STP	mm7						Rt2					Rn					Rt			0xA9800000
130		LDP	mm7						Rt2					Rn					Rt			0xA9C00000
131		STP	mm7						Rt2					Rn					Rt			0xAD800000
132		LDP	mm7						Rt2					Rn					Rt			0xADC00000
133		**Load/store register (unscaled immediate)**			**imm9**					**0**	**0**			**Rn**					**Rt**			
134		STURB			imm9					0	0			Rn					Rt			0x38000000
135		LDURB			imm9					0	0			Rn					Rt			0x38400000
136		LDURSB			imm9					0	0			Rn					Rt			0x38800000
137		LDURSB			imm9					0	0			Rn					Rt			0x38C00000
138		STUR			imm9					0	0			Rn					Rt			0x3C000000
139		LDUR			imm9					0	0			Rn					Rt			0x3C400000
140		STUR			imm9					0	0			Rn					Rt			0x3C800000
141		LDUR			imm9					0	0			Rn					Rt			0x3CC00000
142		STURH			imm9					0	0			Rn					Rt			0x78000000
143		LDURH			imm9					0	0			Rn					Rt			0x78400000
144		LDURSH			imm9					0	0			Rn					Rt			0x78800000
145		LDURSH			imm9					0	0			Rn					Rt			0x78C00000
146		STUR			imm9					0	0			Rn					Rt			0x7C000000
147		LDUR			imm9					0	0			Rn					Rt			0x7C400000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
148		STUR			imm9					0	0			Rn					Rt			0xB8000000
149		LDUR			imm9					0	0			Rn					Rt			0xB8400000
150		LDURSW			imm9					0	0			Rn					Rt			0xB8800000
151		STUR			imm9					0	0			Rn					Rt			0xBC000000
152		LDUR			imm9					0	0			Rn					Rt			0xBC400000
153		STUR			imm9					0	0			Rn					Rt			0xF8000000
154		LDUR			imm9					0	0			Rn					Rt			0xF8400000
155		PRFUM			imm9					0	0			Rn					Rt			0xF8800000
156		STUR			imm9					0	0			Rn					Rt			0xFC000000
157		LDUR			imm9					0	0			Rn					Rt			0xFC400000
158		**Load/store register (immediate post-indexe**			**imm9**					**0**	**1**			**Rn**					**Rt**			
159		STRB			imm9					0	1			Rn					Rt			0x38000400
160		LDRB			imm9					0	1			Rn					Rt			0x38400400
161		LDRSB			imm9					0	1			Rn					Rt			0x38800400
162		LDRSB			imm9					0	1			Rn					Rt			0x38C00400
163		STR			imm9					0	1			Rn					Rt			0x3C000400
164		LDR			imm9					0	1			Rn					Rt			0x3C400400
165		STR			imm9					0	1			Rn					Rt			0x3C800400
166		LDR			imm9					0	1			Rn					Rt			0x3CC00400
167		STRH			imm9					0	1			Rn					Rt			0x78000400
168		LDRH			imm9					0	1			Rn					Rt			0x78400400
169		LDRSH			imm9					0	1			Rn					Rt			0x78800400
170		LDRSH			imm9					0	1			Rn					Rt			0x78C00400
171		STR			imm9					0	1			Rn					Rt			0x7C000400
172		LDR			imm9					0	1			Rn					Rt			0x7C400400
173		STR			imm9					0	1			Rn					Rt			0xB8000400
174		LDR			imm9					0	1			Rn					Rt			0xB8400400
175		LDRSW			imm9					0	1			Rn					Rt			0xB8800400
176		STR			imm9					0	1			Rn					Rt			0xBC000400
177		LDR			imm9					0	1			Rn					Rt			0xBC400400
178		STR			imm9					0	1			Rn					Rt			0xF8000400
179		LDR			imm9					0	1			Rn					Rt			0xF8400400
180		STR			imm9					0	1			Rn					Rt			0xFC000400
181		LDR			imm9					0	1			Rn					Rt			0xFC400400
182		**Load/store register (unprivileged)**			**imm9**					**1**	**0**			**Rn**					**Rt**			
183		STTRB			imm9					1	0			Rn					Rt			0x38000800
184		LDTRB			imm9					1	0			Rn					Rt			0x38400800
185		LDTRSB			imm9					1	0			Rn					Rt			0x38800800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
186		LDTRSB			imm9					1	0			Rn					Rt			0x38C00800
187		STTRH			imm9					1	0			Rn					Rt			0x78000800
188		LDTRH			imm9					1	0			Rn					Rt			0x78400800
189		LDTRSH			imm9					1	0			Rn					Rt			0x78800800
190		LDTRSH			imm9					1	0			Rn					Rt			0x78C00800
191		STTR			imm9					1	0			Rn					Rt			0xB8000800
192		LDTR			imm9					1	0			Rn					Rt			0xB8400800
193		LDTRSW			imm9					1	0			Rn					Rt			0xB8800800
194		STTR			imm9					1	0			Rn					Rt			0xF8000800
195		LDTR			imm9					1	0			Rn					Rt			0xF8400800
196		**Load/store register (immediate pre-indexec**			**imm9**					**1**	**1**			**Rn**					**Rt**			
197		STRB			imm9					1	1			Rn					Rt			0x38000C00
198		LDRB			imm9					1	1			Rn					Rt			0x38400C00
199		LDRSB			imm9					1	1			Rn					Rt			0x38800C00
200		LDRSB			imm9					1	1			Rn					Rt			0x38C00C00
201		STR			imm9					1	1			Rn					Rt			0x3C000C00
202		LDR			imm9					1	1			Rn					Rt			0x3C400C00
203		STR			imm9					1	1			Rn					Rt			0x3C800C00
204		LDR			imm9					1	1			Rn					Rt			0x3CC00C00
205		STRH			imm9					1	1			Rn					Rt			0x78000C00
206		LDRH			imm9					1	1			Rn					Rt			0x78400C00
207		LDRSH			imm9					1	1			Rn					Rt			0x78800C00
208		LDRSH			imm9					1	1			Rn					Rt			0x78C00C00
209		STR			imm9					1	1			Rn					Rt			0x7C000C00
210		LDR			imm9					1	1			Rn					Rt			0x7C400C00
211		STR			imm9					1	1			Rn					Rt			0xB8000C00
212		LDR			imm9					1	1			Rn					Rt			0xB8400C00
213		LDRSW			imm9					1	1			Rn					Rt			0xB8800C00
214		STR			imm9					1	1			Rn					Rt			0xBC000C00
215		LDR			imm9					1	1			Rn					Rt			0xBC400C00
216		STR			imm9					1	1			Rn					Rt			0xF8000C00
217		LDR			imm9					1	1			Rn					Rt			0xF8400C00
218		STR			imm9					1	1			Rn					Rt			0xFC000C00
219		LDR			imm9					1	1			Rn					Rt			0xFC400C00
220		**Load/store register (register offset)**	**Rm**				**option**		**S**	**1**	**0**			**Rn**					**Rt**			
221		STRB	Rm			-	-	-	S	1	0			Rn					Rt			0x38200800
222		LDRB	Rm			-	-	-	S	1	0			Rn					Rt			0x38600800
223		LDRSB	Rm			-	-	-	S	1	0			Rn					Rt			0x38A00800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
224		LDRSB	Rm			-	-	-	S	1	0			Rn					Rt			0x38E00800
225		STR	Rm			-	-	-	S	1	0			Rn					Rt			0x3C200800
226		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0x3C600800
227		STR	Rm			-	-	-	S	1	0			Rn					Rt			0x3CA00800
228		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0x3CE00800
229		STRH	Rm			-	-	-	S	1	0			Rn					Rt			0x78200800
230		LDRH	Rm			-	-	-	S	1	0			Rn					Rt			0x78600800
231		LDRSH	Rm			-	-	-	S	1	0			Rn					Rt			0x78A00800
232		LDRSH	Rm			-	-	-	S	1	0			Rn					Rt			0x78E00800
233		STR	Rm			-	-	-	S	1	0			Rn					Rt			0x7C200800
234		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0x7C600800
235		STR	Rm			-	-	-	S	1	0			Rn					Rt			0xB8200800
236		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0xB8600800
237		LDRSW	Rm			-	-	-	S	1	0			Rn					Rt			0xB8A00800
238		STR	Rm			-	-	-	S	1	0			Rn					Rt			0xBC200800
239		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0xBC600800
240		STR	Rm			-	-	-	S	1	0			Rn					Rt			0xF8200800
241		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0xF8600800
243		STR	Rm			-	-	-	S	1	0			Rn					Rt			0xFC200800
244		LDR	Rm			-	-	-	S	1	0			Rn					Rt			0xFC600800
242		PRFM	Rm			-	-	-	S	1	0			Rn					Rt			0xF8A00800
245		**Load/store register (unsigned immediate)**			**imm12**									**Rn**					**Rt**			
246		STRB			imm12									Rn					Rt			0x39000000
247		LDRB			imm12									Rn					Rt			0x39400000
248		LDRSB			imm12									Rn					Rt			0x39800000
249		LDRSB			imm12									Rn					Rt			0x39C00000
250		STR			imm12									Rn					Rt			0x3D000000
251		LDR			imm12									Rn					Rt			0x3D400000
252		STR			imm12									Rn					Rt			0x3D800000
253		LDR			imm12									Rn					Rt			0x3DC00000
254		STRH			imm12									Rn					Rt			0x79000000
255		LDRH			imm12									Rn					Rt			0x79400000
256		LDRSH			imm12									Rn					Rt			0x79800000
257		LDRSH			imm12									Rn					Rt			0x79C00000
258		STR			imm12									Rn					Rt			0x7D000000
259		LDR			imm12									Rn					Rt			0x7D400000
260		STR			imm12									Rn					Rt			0xB9000000
261		LDR			imm12									Rn					Rt			0xB9400000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
262		LDRSW			imm12									Rn					Rt			0xB9800000
263		STR			imm12									Rn					Rt			0xBD000000
264		LDR			imm12									Rn					Rt			0xBD400000
265		STR			imm12									Rn					Rt			0xF9000000
266		LDR			imm12									Rn					Rt			0xF9400000
268		STR			imm12									Rn					Rt			0xFD000000
269		LDR			imm12									Rn					Rt			0xFD400000
267		PRFM			imm12									Rn					Rt			0xF9800000
270		**Data processing – Immediate**																				
271		**PC-rel. addressing**				**immhi**													**Rd**			
272		ADR				immhi													Rd			0x10000000
273		ADRP				immhi													Rd			0x90000000
274		**Add/subtract (immediate)**			**imm12**									**Rn**					**Rd**			
275		ADD			imm12									Rn					Rd			0x11000000
276		ADDS			imm12									Rn					Rd			0x31000000
277		SUB			imm12									Rn					Rd			0x51000000
278		SUBS			imm12									Rn					Rd			0x71000000
279		ADD			imm12									Rn					Rd			0x91000000
280		ADDS			imm12									Rn					Rd			0xB1000000
281		SUB			imm12									Rn					Rd			0xD1000000
282		SUBS			imm12									Rn					Rd			0xF1000000
283		**Logical (immediate)**	**mr**					**imms**						**Rn**					**Rd**			
284		AND	mr					imms						Rn					Rd			0x12000000
285		ORR	mr					imms						Rn					Rd			0x32000000
286		EOR	mr					imms						Rn					Rd			0x52000000
287		ANDS	mr					imms						Rn					Rd			0x72000000
288		AND	mr					imms						Rn					Rd			0x92000000
289		ORR	mr					imms						Rn					Rd			0xB2000000
290		EOR	mr					imms						Rn					Rd			0xD2000000
291		ANDS	mr					imms						Rn					Rd			0xF2000000
292		**Move wide (immediate)**						**imm16**											**Rd**			
293		MOVN						imm16											Rd			0x12800000
294		MOVZ						imm16											Rd			0x52800000
295		MOVK						imm16											Rd			0x72800000
296		MOVN						imm16											Rd			0x92800000
297		MOVZ						imm16											Rd			0xD2800000
298		MOVK						imm16											Rd			0xF2800000
299		**Bitfield**	**mr**					**imms**						**Rn**					**Rd**			

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
300		SBFM	mr					imms						Rn					Rd			0x13000000
301		BFM	mr					imms						Rn					Rd			0x33000000
302		UBFM	mr					imms						Rn					Rd			0x53000000
303		SBFM	mr					imms						Rn					Rd			0x93400000
304		BFM	mr					imms						Rn					Rd			0xB3400000
305		UBFM	mr					imms						Rn					Rd			0xD3400000
306		**Extract**	**Rm**					**imms**						**Rn**					**Rd**			
307		EXTR	Rm			0	x	x	x	x	x			Rn					Rd			0x13800000
308		EXTR	Rm			-	-	-	-	-	-			Rn					Rd			0x93C00000
309		**Data Processing – register**																				
310		**Logical (shifted register)**	**Rm**					**imm6**						**Rn**					**Rd**			
311		AND	Rm			-	-	-	-	-	-			Rn					Rd			0x0A000000
312		BIC	Rm			-	-	-	-	-	-			Rn					Rd			0x0A200000
313		ORR	Rm			-	-	-	-	-	-			Rn					Rd			0x2A000000
314		ORN	Rm			-	-	-	-	-	-			Rn					Rd			0x2A200000
315		EOR	Rm			-	-	-	-	-	-			Rn					Rd			0x4A000000
316		EON	Rm			-	-	-	-	-	-			Rn					Rd			0x4A200000
317		ANDS	Rm			-	-	-	-	-	-			Rn					Rd			0x6A000000
318		BICS	Rm			-	-	-	-	-	-			Rn					Rd			0x6A200000
319		AND	Rm			-	-	-	-	-	-			Rn					Rd			0x8A000000
320		BIC	Rm			-	-	-	-	-	-			Rn					Rd			0x8A200000
321		ORR	Rm			-	-	-	-	-	-			Rn					Rd			0xAA000000
322		ORN	Rm			-	-	-	-	-	-			Rn					Rd			0xAA200000
323		EOR	Rm			-	-	-	-	-	-			Rn					Rd			0xCA000000
324		EON	Rm			-	-	-	-	-	-			Rn					Rd			0xCA200000
325		ANDS	Rm			-	-	-	-	-	-			Rn					Rd			0xEA000000
326		BICS	Rm			-	-	-	-	-	-			Rn					Rd			0xEA200000
327		**Add/subtract (shifted register)**	**Rm**					**imm6**						**Rn**					**Rd**			
328		ADD	Rm			-	-	-	-	-	-			Rn					Rd			0x0B000000
329		ADDS	Rm			-	-	-	-	-	-			Rn					Rd			0x2B000000
330		SUB	Rm			-	-	-	-	-	-			Rn					Rd			0x4B000000
331		SUBS	Rm			-	-	-	-	-	-			Rn					Rd			0x6B000000
332		ADD	Rm			-	-	-	-	-	-			Rn					Rd			0x8B000000
333		ADDS	Rm			-	-	-	-	-	-			Rn					Rd			0xAB000000
334		SUB	Rm			-	-	-	-	-	-			Rn					Rd			0xCB000000
335		SUBS	Rm			-	-	-	-	-	-			Rn					Rd			0xEB000000
336		**Add/subtract (extended register)**	**Rm**				**option**			**imm3**				**Rn**					**Rd**			
337		ADD	Rm				option			-	-	-		Rn					Rd			0x0B200000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
338		ADDS	Rm			option			-	-	-			Rn					Rd			0x2B200000
339		SUB	Rm			option			-	-	-			Rn					Rd			0x4B200000
340		SUBS	Rm			option			-	-	-			Rn					Rd			0x6B200000
341		ADD	Rm			option			-	-	-			Rn					Rd			0x8B200000
342		ADDS	Rm			option			-	-	-			Rn					Rd			0xAB200000
343		SUB	Rm			option			-	-	-			Rn					Rd			0xCB200000
344		SUBS	Rm			option			-	-	-			Rn					Rd			0xEB200000
345		**Add/subtract (with carry)**	Rm					**opcode2**						**Rn**					**Rd**			
346		ADC	Rm			0	0	0	0	0	0			Rn					Rd			0x1A000000
347		ADCS	Rm			0	0	0	0	0	0			Rn					Rd			0x3A000000
348		SBC	Rm			0	0	0	0	0	0			Rn					Rd			0x5A000000
349		SBCS	Rm			0	0	0	0	0	0			Rn					Rd			0x7A000000
350		ADC	Rm			0	0	0	0	0	0			Rn					Rd			0x9A000000
351		ADCS	Rm			0	0	0	0	0	0			Rn					Rd			0xBA000000
352		SBC	Rm			0	0	0	0	0	0			Rn					Rd			0xDA000000
353		SBCS	Rm			0	0	0	0	0	0			Rn					Rd			0xFA000000
354		**Conditional compare (register)**	**mm5**				**cond**			**0**	**o2**			**Rn**			**o3**		**nzcv**			
355		CCMN	mm5				cond			0	0			Rn			0		nzcv			0x3A400000
356		CCMN	mm5				cond			0	0			Rn			0		nzcv			0xBA400000
357		CCMP	mm5				cond			0	0			Rn			0		nzcv			0x7A400000
358		CCMP	mm5				cond			0	0			Rn			0		nzcv			0xFA400000
359		**Conditional compare (immediate)**	**mm5**				**cond**			**1**	**o2**			**Rn**			**o3**		**nzcv**			
360		CCMN	mm5				cond			1	0			Rn			0		nzcv			0x3A400800
361		CCMN	mm5				cond			1	0			Rn			0		nzcv			0xBA400800
362		CCMP	mm5				cond			1	0			Rn			0		nzcv			0x7A400800
363		CCMP	mm5				cond			1	0			Rn			0		nzcv			0xFA400800
364		**Conditional select**	**Rm**				**cond**			**op2**				**Rn**					**Rd**			
365		CSEL	Rm				cond			0	0			Rn					Rd			0x1A800000
366		CSINC	Rm				cond			0	1			Rn					Rd			0x1A800400
367		CSINV	Rm				cond			0	0			Rn					Rd			0x5A800000
368		CSNEG	Rm				cond			0	1			Rn					Rd			0x5A800400
369		CSEL	Rm				cond			0	0			Rn					Rd			0x9A800000
370		CSINC	Rm				cond			0	1			Rn					Rd			0x9A800400
371		CSINV	Rm				cond			0	0			Rn					Rd			0xDA800000
372		CSNEG	Rm				cond			0	1			Rn					Rd			0xDA800400
373		**Data-processing (3 source)**	**Rm**			**o0**			**Ra**					**Rn**					**Rd**			
374		MADD	Rm			0			Ra					Rn					Rd			0x1B000000
375		MADD	Rm			0			Ra					Rn					Rd			0x9B000000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
376		SMADDL	Rm			0			Ra					Rn					Rd			0x9B200000
377		UMADDL	Rm			0			Ra					Rn					Rd			0x9BA00000
378		MSUB	Rm			1			Ra					Rn					Rd			0x1B008000
379		MSUB	Rm			1			Ra					Rn					Rd			0x9B008000
380		SMSUBL	Rm			1			Ra					Rn					Rd			0x9B208000
381		UMSUBL	Rm			1			Ra					Rn					Rd			0x9BA08000
382		SMULH	Rm			0			Ra					Rn					Rd			0x9B400000
383		UMULH	Rm			0			Ra					Rn					Rd			0x9BC00000
384		**Data-processing (2 source)**	**Rm**					**opcode**						**Rn**					**Rd**			
385		CRC32X	Rm			0	1	0	0	1	1			Rn					Rd			0x9AC04C00
386		CRC32CX	Rm			0	1	0	1	1	1			Rn					Rd			0x9AC05C00
387		CRC32B	Rm			0	1	0	0	0	0			Rn					Rd			0x1AC04000
388		CRC32CB	Rm			0	1	0	1	0	0			Rn					Rd			0x1AC05000
389		CRC32H	Rm			0	1	0	0	0	1			Rn					Rd			0x1AC04400
390		CRC32CH	Rm			0	1	0	1	0	1			Rn					Rd			0x1AC05400
391		CRC32W	Rm			0	1	0	0	1	0			Rn					Rd			0x1AC04800
392		CRC32CW	Rm			0	1	0	1	1	0			Rn					Rd			0x1AC05800
393		UDIV	Rm			0	0	0	0	1	0			Rn					Rd			0x1AC00800
394		UDIV	Rm			0	0	0	0	1	0			Rn					Rd			0x9AC00800
395		SDIV	Rm			0	0	0	0	1	1			Rn					Rd			0x1AC00C00
396		SDIV	Rm			0	0	0	0	1	1			Rn					Rd			0x9AC00C00
397		LSLV	Rm			0	0	1	0	0	0			Rn					Rd			0x1AC02000
398		LSLV	Rm			0	0	1	0	0	0			Rn					Rd			0x9AC02000
399		LSRV	Rm			0	0	1	0	0	1			Rn					Rd			0x1AC02400
400		LSRV	Rm			0	0	1	0	0	1			Rn					Rd			0x9AC02400
401		ASRV	Rm			0	0	1	0	1	0			Rn					Rd			0x1AC02800
402		ASRV	Rm			0	0	1	0	1	0			Rn					Rd			0x9AC02800
403		RORV	Rm			0	0	1	0	1	1			Rn					Rd			0x1AC02C00
404		RORV	Rm			0	0	1	0	1	1			Rn					Rd			0x9AC02C00
405		**Data-processing (1 source)**	**code2**						**opcode**					**Rn**					**Rd**			
406		RBIT	0	0	0	0	0	0	0	0	0			Rn					Rd			0x5AC00000
407		RBIT	0	0	0	0	0	0	0	0	0			Rn					Rd			0xDAC00000
408		CLZ	0	0	0	0	0	0	1	0	0			Rn					Rd			0x5AC01000
409		CLZ	0	0	0	0	0	0	1	0	0			Rn					Rd			0xDAC01000
410		CLS	0	0	0	0	0	0	1	0	1			Rn					Rd			0x5AC01400
411		CLS	0	0	0	0	0	0	1	0	1			Rn					Rd			0xDAC01400
412		REV	0	0	0	0	0	0	0	1	0			Rn					Rd			0x5AC00800
413		REV	0	0	0	0	0	0	0	1	1			Rn					Rd			0xDAC00C00

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
414		REV16	0	0	0	0	0	0	0	0	1			Rn					Rd			0xDAC00400
415		REV16	0	0	0	0	0	0	0	0	1			Rn					Rd			0x5AC00400
416		REV32	0	0	0	0	0	0	0	1	0			Rn					Rd			0xDAC00800
417	//	**Data Processing – SIMD and floating p**																				
418	//	**Floating-point<->fixed-point conversions**	**opcode**					**scale**						**Rn**					**Rd**			
419	//	SCVTF	0	1	0	-	-	-	-	-	-			Rn					Rd			0x1E020000
420	//	UCVTF	0	1	1	-	-	-	-	-	-			Rn					Rd			0x1E030000
421	//	FCVTZS	0	0	0	-	-	-	-	-	-			Rn					Rd			0x1ED80000
422	//	FCVTZU	0	0	1	-	-	-	-	-	-			Rn					Rd			0x1ED90000
423	//	SCVTF	0	1	0	-	-	-	-	-	-			Rn					Rd			0x1E020000
424	//	UCVTF	0	1	1	-	-	-	-	-	-			Rn					Rd			0x1E030000
425	//	FCVTZS	0	0	0	-	-	-	-	-	-			Rn					Rd			0x1ED80000
426	//	FCVTZU	0	0	1	-	-	-	-	-	-			Rn					Rd			0x1ED90000
427	//	SCVTF	0	1	0	-	-	-	-	-	-			Rn					Rd			0x9E020000
428	//	UCVTF	0	1	1	-	-	-	-	-	-			Rn					Rd			0x9E030000
429	//	FCVTZS	0	0	0	-	-	-	-	-	-			Rn					Rd			0x9ED80000
430	//	FCVTZU	0	0	1	-	-	-	-	-	-			Rn					Rd			0x9ED90000
431	//	SCVTF	0	1	0	-	-	-	-	-	-			Rn					Rd			0x9E020000
432	//	UCVTF	0	1	1	-	-	-	-	-	-			Rn					Rd			0x9E030000
433	//	FCVTZS	0	0	0	-	-	-	-	-	-			Rn					Rd			0x9ED80000
434	//	FCVTZU	0	0	1	-	-	-	-	-	-			Rn					Rd			0x9ED90000
435	//	**Floating-point conditional compare**	**Rm**				**cond**			**0**	**1**			**Rn**			**op**		**nzcv**			
436	//	FCCMP	Rm				cond			0	1			Rn			0		nzcv			0x1E200400
437	//	FCCMPE	Rm				cond			0	1			Rn			1		nzcv			0x1E200410
438	//	FCCMP	Rm				cond			0	1			Rn			0		nzcv			0x1E600400
439	//	FCCMPE	Rm				cond			0	1			Rn			1		nzcv			0x1E600410
440	//	**Floating-point data-processing (2 source)**	**Rm**				**opcode**			**1**	**0**			**Rn**					**Rd**			
441	//	FMUL	Rm			0	0	0	0	1	0			Rn					Rd			0x1E200800
442	//	FDIV	Rm			0	0	0	1	1	0			Rn					Rd			0x1E201800
443	//	FADD	Rm			0	0	1	0	1	0			Rn					Rd			0x1E202800
444	//	FSUB	Rm			0	0	1	1	1	0			Rn					Rd			0x1E203800
445	//	FMAX	Rm			0	1	0	0	1	0			Rn					Rd			0x1E204800
446	//	FMIN	Rm			0	1	0	1	1	0			Rn					Rd			0x1E205800
447	//	FMAXNM	Rm			0	1	1	0	1	0			Rn					Rd			0x1E206800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
448	//	FMINNM	Rm			0	1	1	1	1	0			Rn					Rd			0x1E207800
449	//	FNMUL	Rm			1	0	0	0	1	0			Rn					Rd			0x1E208800
450	//	FMUL	Rm			0	0	0	0	1	0			Rn					Rd			0x1E600800
451	//	FDIV	Rm			0	0	0	1	1	0			Rn					Rd			0x1E601800
452	//	FADD	Rm			0	0	1	0	1	0			Rn					Rd			0x1E602800
453	//	FSUB	Rm			0	0	1	1	1	0			Rn					Rd			0x1E603800
454	//	FMAX	Rm			0	1	0	0	1	0			Rn					Rd			0x1E604800
455	//	FMIN	Rm			0	1	0	1	1	0			Rn					Rd			0x1E605800
456	//	FMAXNM	Rm			0	1	1	0	1	0			Rn					Rd			0x1E606800
457	//	FMINNM	Rm			0	1	1	1	1	0			Rn					Rd			0x1E607800
458	//	FNMUL	Rm			1	0	0	0	1	0			Rn					Rd			0x1E608800
459	//	**Floating-point conditional select**	**Rm**				**cond**			**1**	**1**			**Rn**					**Rd**			
460	//	FCSEL	Rm				cond			1	1			Rn					Rd			0x1E200C00
461	//	FCSEL	Rm				cond			1	1			Rn					Rd			0x1E600C00
462	//	**Floating-point immediate**		**imm8**					**1**	**0**	**0**			**imm5**					**Rd**			
463	//	FMOV		imm8					1	0	0	0	0	0	0	0			Rd			0x1E201000
464	//	FMOV		imm8					1	0	0	0	0	0	0	0			Rd			0x1E601000
465	//	**Floating-point compare**	**Rm**				**op**		**1**	**0**	**0**	**0**		**Rn**				**opcode2**				
466	//	FCMP	Rm			0	0	1	0	0	0			Rn			0	0	0	0	0	0x1E202000
467	//	FCMP	Rm			0	0	1	0	0	0			Rn			0	1	0	0	0	0x1E202008
468	//	FCMPE	Rm			0	0	1	0	0	0			Rn			1	0	0	0	0	0x1E202010
469	//	FCMPE	Rm			0	0	1	0	0	0			Rn			1	1	0	0	0	0x1E202018
470	//	FCMP	Rm			0	0	1	0	0	0			Rn			0	0	0	0	0	0x1E602000
471	//	FCMP	Rm			0	0	1	0	0	0			Rn			0	1	0	0	0	0x1E602008
472	//	FCMPE	Rm			0	0	1	0	0	0			Rn			1	0	0	0	0	0x1E602010
473	//	FCMPE	Rm			0	0	1	0	0	0			Rn			1	1	0	0	0	0x1E602018
474	//	**Floating-point data-processing (1 source)**	**opcode**				**1**	**0**	**0**	**0**	**0**			**Rn**					**Rd**			
475	//	FMOV	0	0	0	0	1	0	0	0	0			Rn					Rd			0x1E204000
476	//	FABS	0	0	0	1	1	0	0	0	0			Rn					Rd			0x1E20C000
477	//	FNEG	0	0	1	0	1	0	0	0	0			Rn					Rd			0x1E214000
478	//	FSQRT	0	0	1	1	1	0	0	0	0			Rn					Rd			0x1E21C000
479	//	FCVT	0	1	0	1	1	0	0	0	0			Rn					Rd			0x1E22C000
480	//	FCVT	0	1	1	1	1	0	0	0	0			Rn					Rd			0x1E23C000
481	//	FRINTN	1	0	0	0	1	0	0	0	0			Rn					Rd			0x1E244000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
482	//	FRINTP	1	0	0	1	1	0	0	0	0			Rn					Rd			0x1E24C000
483	//	FRINTM	1	0	1	0	1	0	0	0	0			Rn					Rd			0x1E254000
484	//	FRINTZ	1	0	1	1	1	0	0	0	0			Rn					Rd			0x1E25C000
485	//	FRINTA	1	1	0	0	1	0	0	0	0			Rn					Rd			0x1E264000
486	//	FRINTX	1	1	1	0	1	0	0	0	0			Rn					Rd			0x1E274000
487	//	FRINTI	1	1	1	1	1	0	0	0	0			Rn					Rd			0x1E27C000
488	//	FMOV	0	0	0	0	1	0	0	0	0			Rn					Rd			0x1E604000
489	//	FABS	0	0	0	1	1	0	0	0	0			Rn					Rd			0x1E60C000
490	//	FNEG	0	0	1	0	1	0	0	0	0			Rn					Rd			0x1E614000
491	//	FSQRT	0	0	1	1	1	0	0	0	0			Rn					Rd			0x1E61C000
492	//	FCVT	0	1	0	1	1	0	0	0	0			Rn					Rd			0x1E62C000
493	//	FCVT	0	1	1	1	1	0	0	0	0			Rn					Rd			0x1E63C000
494	//	FRINTN	1	0	0	0	1	0	0	0	0			Rn					Rd			0x1E644000
495	//	FRINTP	1	0	0	1	1	0	0	0	0			Rn					Rd			0x1E64C000
496	//	FRINTM	1	0	1	0	1	0	0	0	0			Rn					Rd			0x1E654000
497	//	FRINTZ	1	0	1	1	1	0	0	0	0			Rn					Rd			0x1E65C000
498	//	FRINTA	1	1	0	0	1	0	0	0	0			Rn					Rd			0x1E664000
499	//	FRINTX	1	1	1	0	1	0	0	0	0			Rn					Rd			0x1E674000
500	//	FRINTI	1	1	1	1	1	0	0	0	0			Rn					Rd			0x1E67C000
501	//	FCVT	0	1	0	0	1	0	0	0	0			Rn					Rd			0x1EE24000
502	//	FCVT	0	1	0	1	1	0	0	0	0			Rn					Rd			0x1EE2C000
503	//	**Floating-point<->integer conversions**	**opcode**			**0**	**0**	**0**	**0**	**0**	**0**			**Rn**					**Rd**			
504	//	FCVTNS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E200000
505	//	FCVTNU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E210000
506	//	SCVTF	0	1	0	0	0	0	0	0	0			Rn					Rd			0x1E220000
507	//	UCVTF	0	1	1	0	0	0	0	0	0			Rn					Rd			0x1E230000
508	//	FCVTAS	1	0	0	0	0	0	0	0	0			Rn					Rd			0x1E240000
509	//	FCVTAU	1	0	1	0	0	0	0	0	0			Rn					Rd			0x1E250000
510	//	FMOV	1	1	0	0	0	0	0	0	0			Rn					Rd			0x1E260000
511	//	FMOV	1	1	1	0	0	0	0	0	0			Rn					Rd			0x1E270000
512	//	FCVTPS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E280000
513	//	FCVTPU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E290000
514	//	FCVTMS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E300000
515	//	FCVTMU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E310000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
516	//	FCVTZS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E380000
517	//	FCVTZU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E390000
518	//	FCVTNS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E600000
519	//	FCVTNU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E610000
520	//	SCVTF	0	1	0	0	0	0	0	0	0			Rn					Rd			0x1E620000
521	//	UCVTF	0	1	1	0	0	0	0	0	0			Rn					Rd			0x1E630000
522	//	FCVTAS	1	0	0	0	0	0	0	0	0			Rn					Rd			0x1E640000
523	//	FCVTAU	1	0	1	0	0	0	0	0	0			Rn					Rd			0x1E650000
524	//	FCVTPS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E680000
525	//	FCVTPU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E690000
526	//	FCVTMS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E700000
527	//	FCVTMU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E710000
528	//	FCVTZS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x1E780000
529	//	FCVTZU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x1E790000
530	//	FCVTNS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E200000
531	//	FCVTNU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E210000
532	//	SCVTF	0	1	0	0	0	0	0	0	0			Rn					Rd			0x9E220000
533	//	UCVTF	0	1	1	0	0	0	0	0	0			Rn					Rd			0x9E230000
534	//	FCVTAS	1	0	0	0	1	0	0	0	0			Rn					Rd			0x9E244000
535	//	FCVTAU	1	0	1	0	0	0	0	0	0			Rn					Rd			0x9E250000
536	//	FCVTPS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E280000
537	//	FCVTPU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E290000
538	//	FCVTMS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E300000
539	//	FCVTMU	0	0	1	1	0	0	0	0	0			Rn					Rd			0x9E318000
540	//	FCVTZS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E380000
541	//	FCVTZU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E390000
542	//	FCVTNS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E200000
543	//	FCVTNU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E210000
544	//	SCVTF	0	1	0	0	0	0	0	0	0			Rn					Rd			0x9E620000
545	//	UCVTF	0	1	1	0	0	0	0	0	0			Rn					Rd			0x9E630000
546	//	FCVTAS	1	0	0	0	0	0	0	0	0			Rn					Rd			0x9E640000
547	//	FCVTAU	1	0	1	0	0	0	0	0	0			Rn					Rd			0x9E650000
548	//	FMOV	1	1	0	0	0	0	0	0	0			Rn					Rd			0x9E660000
549	//	FMOV	1	1	1	0	0	0	0	0	0			Rn					Rd			0x9E670000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
550	//	FCVTPS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E680000
551	//	FCVTPU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E690000
552	//	FCVTMS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E700000
553	//	FCVTMU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E710000
554	//	FCVTZS	0	0	0	0	0	0	0	0	0			Rn					Rd			0x9E780000
555	//	FCVTZU	0	0	1	0	0	0	0	0	0			Rn					Rd			0x9E790000
556	//	FMOV	1	1	0	0	0	0	0	0	0			Rn					Rd			0x9EAE0000
557	//	FMOV	1	1	1	0	0	0	0	0	0			Rn					Rd			0x9EAF0000
558	//	**Floating-point data-processing (3 source)**	**Rm**			**o0**			**Ra**					**Rn**					**Rd**			
559	//	FMADD	Rm			0			Ra					Rn					Rd			0x1F000000
560	//	FMSUB	Rm			1			Ra					Rn					Rd			0x1F008000
561	//	FNMADD	Rm			0			Ra					Rn					Rd			0x1F200000
562	//	FNMSUB	Rm			1			Ra					Rn					Rd			0x1F208000
563	//	FMADD	Rm			0			Ra					Rn					Rd			0x1F400000
564	//	FMSUB	Rm			1			Ra					Rn					Rd			0x1F408000
565	//	FNMADD	Rm			0			Ra					Rn					Rd			0x1F600000
566	//	FNMSUB	Rm			1			Ra					Rn					Rd			0x1F608000
567	//	**AdvSIMD scalar three same**	**Rm**				**opcode**				**1**			**Rn**					**Rd**			
568	//	SQADD	Rm			0	0	0	0	1	1			Rn					Rd			0x5E200C00
569	//	SQSUB	Rm			0	0	1	0	1	1			Rn					Rd			0x5E202C00
570	//	CMGT	Rm			0	0	1	1	0	1			Rn					Rd			0x5E203400
571	//	CMGE	Rm			0	0	1	1	1	1			Rn					Rd			0x5E203C00
572	//	SSHL	Rm			0	1	0	0	0	1			Rn					Rd			0x5E204400
573	//	SQSHL	Rm			0	1	0	0	1	1			Rn					Rd			0x5E204C00
574	//	SRSHL	Rm			0	1	0	1	0	1			Rn					Rd			0x5E205400
575	//	SQRSHL	Rm			0	1	0	1	1	1			Rn					Rd			0x5E205C00
576	//	ADD	Rm			1	0	0	0	0	1			Rn					Rd			0x5E208400
577	//	CMTST	Rm			1	0	0	0	1	1			Rn					Rd			0x5E208C00
578	//	SQDMULH	Rm			1	0	1	1	0	1			Rn					Rd			0x5E20B400
579	//	FMULX	Rm			1	1	0	1	1	1			Rn					Rd			0x5E20DC00
580	//	FCMEQ	Rm			1	1	1	0	0	1			Rn					Rd			0x5E20E400
581	//	FRECPS	Rm			1	1	1	1	1	1			Rn					Rd			0x5E20FC00
582	//	FRSQRTS	Rm			1	1	1	1	1	1			Rn					Rd			0x5EA0FC00
583	//	UQADD	Rm			0	0	0	0	1	1			Rn					Rd			0x7E200C00

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
584	//	UQSUB	Rm			0	0	1	0	1	1			Rn					Rd			0x7E202C00
585	//	CMHI	Rm			0	0	1	1	0	1			Rn					Rd			0x7E203400
586	//	CMHS	Rm			0	0	1	1	1	1			Rn					Rd			0x7E203C00
587	//	USHL	Rm			0	1	0	0	0	1			Rn					Rd			0x7E204400
588	//	UQSHL	Rm			0	1	0	0	1	1			Rn					Rd			0x7E204C00
589	//	URSHL	Rm			0	1	0	1	0	1			Rn					Rd			0x7E205400
590	//	UQRSHL	Rm			0	1	0	1	1	1			Rn					Rd			0x7E205C00
591	//	SUB	Rm			1	0	0	0	0	1			Rn					Rd			0x7E208400
592	//	CMEQ	Rm			1	0	0	0	1	1			Rn					Rd			0x7E208C00
593	//	SQRDMULH	Rm			1	0	1	1	0	1			Rn					Rd			0x7E20B400
594	//	FCMGE	Rm			1	1	1	0	0	1			Rn					Rd			0x7E20E400
595	//	FACGE	Rm			1	1	1	0	1	1			Rn					Rd			0x7E20EC00
596	//	FABD	Rm			1	1	0	1	0	1			Rn					Rd			0x7EA0D400
597	//	FCMGT	Rm			1	1	1	0	0	1			Rn					Rd			0x7EA0E400
598	//	FACGT	Rm			1	1	1	0	1	1			Rn					Rd			0x7EA0EC00
599	//	**AdvSIMD scalar three different**	**Rm**				**opcode**			**0**	**0**			**Rn**					**Rd**			
600	//	SQDMLAL	Rm			1	0	0	1	0	0			Rn					Rd			0x5E209000
601	//	SQDMLAL2	Rm			1	0	0	1	0	0			Rn					Rd			0x5E209000
602	//	SQDMLSL	Rm			1	0	1	1	0	0			Rn					Rd			0x5E20B000
603	//	SQDMLSL2	Rm			1	0	1	1	0	0			Rn					Rd			0x5E20B000
604	//	SQDMULL	Rm			1	1	0	1	0	0			Rn					Rd			0x5E20D000
605	//	SQDMULL2	Rm			1	1	0	1	0	0			Rn					Rd			0x5E20D000
606	//	**AdvSIMD scalar two-reg misc**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**			
607	//	SUQADD	0	0	0	0	0	1	1	1	0			Rn					Rd			0x5E203800
608	//	SQABS	0	0	0	0	1	1	1	1	0			Rn					Rd			0x5E207800
609	//	CMGT	0	0	0	1	0	0	0	1	0			Rn					Rd			0x5E208800
610	//	CMEQ	0	0	0	1	0	0	1	1	0			Rn					Rd			0x5E209800
611	//	CMLT	0	0	0	1	0	1	0	1	0			Rn					Rd			0x5E20A800
612	//	ABS	0	0	0	1	0	1	1	1	0			Rn					Rd			0x5E20B800
613	//	SQXTN	0	0	1	0	1	0	0	1	0			Rn					Rd			0x5E214800
614	//	SQXTN2	0	0	1	0	1	0	0	1	0			Rn					Rd			0x5E214800
615	//	FCVTNS	0	0	1	1	0	1	0	1	0			Rn					Rd			0x5E21A800
616	//	FCVTMS	0	0	1	1	0	1	1	1	0			Rn					Rd			0x5E21B800
617	//	FCVTAS	0	0	1	1	1	0	0	1	0			Rn					Rd			0x5E21C800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
618	//	SCVTF	0	0	1	1	1	0	1	1	0			Rn					Rd			0x5E21D800
619	//	FCMGT	0	0	0	1	1	0	0	1	0			Rn					Rd			0x5EA0C800
620	//	FCMEQ	0	0	0	1	1	0	1	1	0			Rn					Rd			0x5EA0D800
621	//	FCMLT	0	0	0	1	1	1	0	1	0			Rn					Rd			0x5EA0E800
622	//	FCVTPS	0	0	1	1	0	1	0	1	0			Rn					Rd			0x5EA1A800
623	//	FCVTZS	0	0	1	1	0	1	1	1	0			Rn					Rd			0x5EA1B800
624	//	FRECPE	0	0	1	1	1	0	1	1	0			Rn					Rd			0x5EA1D800
625	//	FRECPX	0	0	1	1	1	1	1	1	0			Rn					Rd			0x5EA1F800
626	//	USQADD	0	0	0	0	0	1	1	1	0			Rn					Rd			0x7E203800
627	//	SQNEG	0	0	0	0	1	1	1	1	0			Rn					Rd			0x7E207800
628	//	CMGE	0	0	0	1	0	0	0	1	0			Rn					Rd			0x7E208800
629	//	CMLE	0	0	0	1	0	0	1	1	0			Rn					Rd			0x7E209800
630	//	NEG	0	0	0	1	0	1	1	1	0			Rn					Rd			0x7E20B800
631	//	SQXTUN	0	0	1	0	0	1	0	1	0			Rn					Rd			0x7E212800
632	//	SQXTUN2	0	0	1	0	0	1	0	1	0			Rn					Rd			0x7E212800
633	//	UQXTN	0	0	1	0	1	0	0	1	0			Rn					Rd			0x7E214800
634	//	UQXTN2	0	0	1	0	1	0	0	1	0			Rn					Rd			0x7E214800
635	//	FCVTXN	0	0	1	0	1	1	0	1	0			Rn					Rd			0x7E216800
636	//	FCVTXN2	0	0	1	0	1	1	0	1	0			Rn					Rd			0x7E216800
637	//	FCVTNU	0	0	1	1	0	1	0	1	0			Rn					Rd			0x7E21A800
638	//	FCVTMU	0	0	1	1	0	1	1	1	0			Rn					Rd			0x7E21B800
639	//	FCVTAU	0	0	1	1	1	0	0	1	0			Rn					Rd			0x7E21C800
640	//	UCVTF	0	0	1	1	1	0	1	1	0			Rn					Rd			0x7E21D800
641	//	FCMGE	0	0	0	1	1	0	0	1	0			Rn					Rd			0x7EA0C800
642	//	FCMLE	0	0	0	1	1	0	1	1	0			Rn					Rd			0x7EA0D800
643	//	FCVTPU	0	0	1	1	0	1	0	1	0			Rn					Rd			0x7EA1A800
644	//	FCVTZU	0	0	1	1	0	1	1	1	0			Rn					Rd			0x7EA1B800
645	//	FRSQRTE	0	0	1	1	1	0	1	1	0			Rn					Rd			0x7EA1D800
646	//	**AdvSIMD scalar pairwise**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**			
647	//	ADDP	0	0	1	1	0	1	1	1	0			Rn					Rd			0x5E31B800
648	//	FMAXNMP	0	0	0	1	1	0	0	1	0			Rn					Rd			0x7E30C800
649	//	FADDP	0	0	0	1	1	0	1	1	0			Rn					Rd			0x7E30D800
650	//	FMAXP	0	0	0	1	1	1	1	1	0			Rn					Rd			0x7E30F800
651	//	FMINNMP	0	0	0	1	1	0	0	1	0			Rn					Rd			0x7EB0C800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
652	//	FMINP	0	0	0	1	1	1	1	1	0			Rn					Rd			0x7EB0F800
653	//	**AdvSIMD scalar copy**	**imm5**			**0**		**imm4**			**1**			**Rn**					**Rd**			
654	//	DUP	-	-	-	0	0	0	0	0	1			Rn					Rd			0x5E000400
655	//	**AdvSIMD scalar x indexed element**	**Rm**				**opcode**			**H**	**0**			**Rn**					**Rd**			
656	//	SQDMLAL	Rm			0	0	1	1	H	0			Rn					Rd			0x5F003000
657	//	SQDMLAL2	Rm			0	0	1	1	H	0			Rn					Rd			0x5F003000
658	//	SQDMLSL	Rm			0	1	1	1	H	0			Rn					Rd			0x5F007000
659	//	SQDMLSL2	Rm			0	1	1	1	H	0			Rn					Rd			0x5F007000
660	//	SQDMULL	Rm			1	0	1	1	H	0			Rn					Rd			0x5F00B000
661	//	SQDMULL2	Rm			1	0	1	1	H	0			Rn					Rd			0x5F00B000
662	//	SQDMULH	Rm			1	1	0	0	H	0			Rn					Rd			0x5F00C000
663	//	SQRDMULH	Rm			1	1	0	1	H	0			Rn					Rd			0x5F00D000
664	//	FMLA	Rm			0	0	0	1	H	0			Rn					Rd			0x5F801000
665	//	FMLS	Rm			0	1	0	1	H	0			Rn					Rd			0x5F805000
666	//	FMUL	Rm			1	0	0	1	H	0			Rn					Rd			0x5F809000
667	//	FMULX	Rm			1	0	0	1	H	0			Rn					Rd			0x7F809000
668	//	**AdvSIMD scalar shift by immediate**	**immb**				**opcode**				**1**			**Rn**					**Rd**			
669	//	SSHR	immb			0	0	0	0	0	1			Rn					Rd			0x5F000400
670	//	SSRA	immb			0	0	0	1	0	1			Rn					Rd			0x5F001400
671	//	SRSHR	immb			0	0	1	0	0	1			Rn					Rd			0x5F002400
672	//	SRSRA	immb			0	0	1	1	0	1			Rn					Rd			0x5F003400
673	//	SHL	immb			0	1	0	1	0	1			Rn					Rd			0x5F005400
674	//	SQSHL	immb			0	1	1	1	0	1			Rn					Rd			0x5F007400
675	//	SQSHRN	immb			1	0	0	1	0	1			Rn					Rd			0x5F009400
676	//	SQSHRN2	immb			1	0	0	1	0	1			Rn					Rd			0x5F009400
677	//	SQRSHRN	immb			1	0	0	1	1	1			Rn					Rd			0x5F009C00
678	//	SQRSHRN2	immb			1	0	0	1	1	1			Rn					Rd			0x5F009C00
679	//	SCVTF	immb			1	1	1	0	0	1			Rn					Rd			0x5F00E400
680	//	FCVTZS	immb			1	1	1	1	1	1			Rn					Rd			0x5F00FC00
681	//	USHR	immb			0	0	0	0	0	1			Rn					Rd			0x7F000400
682	//	USRA	immb			0	0	0	1	0	1			Rn					Rd			0x7F001400
683	//	URSHR	immb			0	0	1	0	0	1			Rn					Rd			0x7F002400
684	//	URSRA	immb			0	0	1	1	0	1			Rn					Rd			0x7F003400
685	//	SRI	immb			0	1	0	0	0	1			Rn					Rd			0x7F004400

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
686	//	SLI		immb		0	1	0	1	0	1			Rn					Rd			0x7F005400
687	//	SQSHLU		immb		0	1	1	0	0	1			Rn					Rd			0x7F006400
688	//	UQSHL		immb		0	1	1	1	0	1			Rn					Rd			0x7F007400
689	//	SQSHRUN		immb		1	0	0	0	0	1			Rn					Rd			0x7F008400
690	//	SQSHRUN2		immb		1	0	0	0	0	1			Rn					Rd			0x7F008400
691	//	SQRSHRUN		immb		1	0	0	0	1	1			Rn					Rd			0x7F008C00
692	//	SQRSHRUN2		immb		1	0	0	0	1	1			Rn					Rd			0x7F008C00
693	//	UQSHRN		immb		1	0	0	1	0	1			Rn					Rd			0x7F009400
694	//	UQRSHRN		immb		1	0	0	1	1	1			Rn					Rd			0x7F009C00
695	//	UQRSHRN2		immb		1	0	0	1	1	1			Rn					Rd			0x7F009C00
696	//	UCVTF		immb		1	1	1	0	0	1			Rn					Rd			0x7F00E400
697	//	FCVTZU		immb		1	1	1	1	1	1			Rn					Rd			0x7F00FC00
698	//	**Crypto three-reg SHA**	**Rm**			**0**	**opcode**			**0**	**0**			**Rn**					**Rd**			
699	//	SHA1C	Rm			0	0	0	0	0	0			Rn					Rd			0x5E000000
700	//	SHA1P	Rm			0	0	0	1	0	0			Rn					Rd			0x5E001000
701	//	SHA1M	Rm			0	0	1	0	0	0			Rn					Rd			0x5E002000
702	//	SHA1SU0	Rm			0	0	1	1	0	0			Rn					Rd			0x5E003000
703	//	SHA256H	Rm			0	1	0	0	0	0			Rn					Rd			0x5E004000
704	//	SHA256H2	Rm			0	1	0	1	0	0			Rn					Rd			0x5E005000
705	//	SHA256SU1	Rm			0	1	1	0	0	0			Rn					Rd			0x5E006000
706	//	**Crypto two-reg SHA**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**			
707	//	SHA1H	0	0	0	0	0	0	0	1	0			Rn					Rd			0x5E280800
708	//	SHA1SU1	0	0	0	0	0	0	1	1	0			Rn					Rd			0x5E281800
709	//	SHA256SU0	0	0	0	0	0	1	0	1	0			Rn					Rd			0x5E282800
710	//	**Crypto AES**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**			
711	//	AESE	0	0	0	0	1	0	0	1	0			Rn					Rd			0x4E284800
712	//	AESD	0	0	0	0	1	0	1	1	0			Rn					Rd			0x4E285800
713	//	AESMC	0	0	0	0	1	1	0	1	0			Rn					Rd			0x4E286800
714	//	AESIMC	0	0	0	0	1	1	1	1	0			Rn					Rd			0x4E287800
715	//	**AdvSIMD three same**	**Rm**				**opcode**				**1**			**Rn**					**Rd**			
716	//	SHADD	Rm			0	0	0	0	0	1			Rn					Rd			0x0E200400
717	//	SQADD	Rm			0	0	0	0	1	1			Rn					Rd			0x0E200C00
718	//	SRHADD	Rm			0	0	0	1	0	1			Rn					Rd			0x0E201400
719	//	SHSUB	Rm			0	0	1	0	0	1			Rn					Rd			0x0E202400

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
720	//	SQSUB	Rm			0	0	1	0	1	1			Rn					Rd			0x0E202C00
721	//	CMGT	Rm			0	0	1	1	0	1			Rn					Rd			0x0E203400
722	//	CMGE	Rm			0	0	1	1	1	1			Rn					Rd			0x0E203C00
723	//	SSHL Vector	Rm			0	1	0	0	0	1			Rn					Rd			0x0E204400
724	//	SQSHL	Rm			0	1	0	0	1	1			Rn					Rd			0x0E204C00
725	//	SRSHL	Rm			0	1	0	1	0	1			Rn					Rd			0x0E205400
726	//	SQRSHL	Rm			0	1	0	1	1	1			Rn					Rd			0x0E205C00
727	//	SMAX	Rm			0	1	1	0	0	1			Rn					Rd			0x0E206400
728	//	SMIN	Rm			0	1	1	0	1	1			Rn					Rd			0x0E206C00
729	//	SABD	Rm			0	1	1	1	0	1			Rn					Rd			0x0E207400
730	//	SABA	Rm			0	1	1	1	1	1			Rn					Rd			0x0E207C00
731	//	ADD	Rm			1	0	0	0	0	1			Rn					Rd			0x0E208400
732	//	CMTST	Rm			1	0	0	0	1	1			Rn					Rd			0x0E208C00
733	//	MLA	Rm			1	0	0	1	0	1			Rn					Rd			0x0E209400
734	//	MUL	Rm			1	0	0	1	1	1			Rn					Rd			0x0E209C00
735	//	SMAXP	Rm			1	0	1	0	0	1			Rn					Rd			0x0E20A400
736	//	SMINP	Rm			1	0	1	0	1	1			Rn					Rd			0x0E20AC00
737	//	SQDMULH	Rm			1	0	1	1	0	1			Rn					Rd			0x0E20B400
738	//	ADDP	Rm			1	0	1	1	1	1			Rn					Rd			0x0E20BC00
739	//	FMAXNM	Rm			1	1	0	0	0	1			Rn					Rd			0x0E20C400
740	//	FMLA	Rm			1	1	0	0	1	1			Rn					Rd			0x0E20CC00
741	//	FADD	Rm			1	1	0	1	0	1			Rn					Rd			0x0E20D400
742	//	FMULX	Rm			1	1	0	1	1	1			Rn					Rd			0x0E20DC00
743	//	FCMEQ	Rm			1	1	1	0	0	1			Rn					Rd			0x0E20E400
744	//	FMAX	Rm			1	1	1	1	0	1			Rn					Rd			0x0E20F400
745	//	FRECPS	Rm			1	1	1	1	1	1			Rn					Rd			0x0E20FC00
746	//	AND	Rm			0	0	0	1	1	1			Rn					Rd			0x0E201C00
747	//	BIC	Rm			0	0	0	1	1	1			Rn					Rd			0x0E601C00
748	//	FMINNM	Rm			1	1	0	0	0	1			Rn					Rd			0x0EA0C400
749	//	FMLS	Rm			1	1	0	0	1	1			Rn					Rd			0x0EA0CC00
750	//	FSUB	Rm			1	1	0	1	0	1			Rn					Rd			0x0EA0D400
751	//	FMIN	Rm			1	1	1	1	0	1			Rn					Rd			0x0EA0F400
752	//	FRSQRTS	Rm			1	1	1	1	1	1			Rn					Rd			0x0EA0FC00
753	//	ORR	Rm			0	0	0	1	1	1			Rn					Rd			0x0EA01C00

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
754	//	ORN	Rm			0	0	0	1	1	1			Rn					Rd			0x0EE01C00
755	//	UHADD	Rm			0	0	0	0	0	1			Rn					Rd			0x2E200400
756	//	UQADD	Rm			0	0	0	0	1	1			Rn					Rd			0x2E200C00
757	//	URHADD	Rm			0	0	0	1	0	1			Rn					Rd			0x2E201400
758	//	UHSUB	Rm			0	0	1	0	0	1			Rn					Rd			0x2E202400
759	//		Rm			0	0	1	0	1	1			Rn					Rd			0x2E202C00
760	//	CMHI	Rm			0	0	1	1	0	1			Rn					Rd			0x2E203400
761	//	CMHS	Rm			0	0	1	1	1	1			Rn					Rd			0x2E203C00
762	//	USHL	Rm			0	1	0	0	0	1			Rn					Rd			0x2E204400
763	//	UQSHL	Rm			0	1	0	0	1	1			Rn					Rd			0x2E204C00
764	//	URSHL	Rm			0	1	0	1	0	1			Rn					Rd			0x2E205400
765	//	UQRSHL	Rm			0	1	0	1	1	1			Rn					Rd			0x2E205C00
766	//	UMAX	Rm			0	1	1	0	0	1			Rn					Rd			0x2E206400
767	//	UMIN	Rm			0	1	1	0	1	1			Rn					Rd			0x2E206C00
768	//	UABD	Rm			0	1	1	1	0	1			Rn					Rd			0x2E207400
769	//	UABA	Rm			0	1	1	1	1	1			Rn					Rd			0x2E207C00
770	//	SUB	Rm			1	0	0	0	0	1			Rn					Rd			0x2E208400
771	//	CMEQ	Rm			1	0	0	0	1	1			Rn					Rd			0x2E208C00
772	//	MLS	Rm			1	0	0	1	0	1			Rn					Rd			0x2E209400
773	//	PMUL	Rm			1	0	0	1	1	1			Rn					Rd			0x2E209C00
774	//	UMAXP	Rm			1	0	1	0	0	1			Rn					Rd			0x2E20A400
775	//	UMINP	Rm			1	0	1	0	1	1			Rn					Rd			0x2E20AC00
776	//	SQRDMULH	Rm			1	0	1	1	0	1			Rn					Rd			0x2E20B400
777	//	FMAXNMP	Rm			1	0	1	1	0	1			Rn					Rd			0x2E20B400
778	//	FADDP	Rm			1	1	0	1	0	1			Rn					Rd			0x2E20D400
779	//	FMUL	Rm			1	1	0	1	1	1			Rn					Rd			0x2E20DC00
780	//	FCMGE	Rm			1	1	1	0	0	1			Rn					Rd			0x2E20E400
781	//	FACGE	Rm			1	1	1	0	1	1			Rn					Rd			0x2E20EC00
782	//	FMAXP	Rm			1	1	1	1	0	1			Rn					Rd			0x2E20F400
783	//	FDIV	Rm			1	1	1	1	1	1			Rn					Rd			0x2E20FC00
784	//	EOR	Rm			0	0	0	1	1	1			Rn					Rd			0x2E201C00
785	//	BSL	Rm			0	0	0	1	1	1			Rn					Rd			0x2E601C00
786	//	FMINNMP	Rm			1	1	0	0	0	1			Rn					Rd			0x2EA0C400
787	//	FABD	Rm			1	1	0	1	0	1			Rn					Rd			0x2EA0D400

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
788	//	FCMGT	Rm			1	1	1	0	0	1			Rn					Rd			0x2EA0E400
789	//	FACGT	Rm			1	1	1	0	1	1			Rn					Rd			0x2EA0EC00
790	//	FMINP	Rm			1	1	1	1	0	1			Rn					Rd			0x2EA0F400
791	//	BIT	Rm			0	0	0	1	1	1			Rn					Rd			0x2EA01C00
792	//	BIF	Rm			0	0	0	1	1	1			Rn					Rd			0x2EE01C00
793	//	**AdvSIMD three different**	**Rm**				**opcode**			**0**	**0**			**Rn**					**Rd**			
794	//	SADDL	Rm			0	0	0	0	0	0			Rn					Rd			0x0E200000
795	//	SADDL2	Rm			0	0	0	0	0	0			Rn					Rd			0x4E200000
796	//	SADDW	Rm			0	0	0	1	0	0			Rn					Rd			0x0E201000
797	//	SADDW2	Rm			0	0	0	1	0	0			Rn					Rd			0x4E201000
798	//	SSUBL	Rm			0	0	1	0	0	0			Rn					Rd			0x0E202000
799	//	SSUBL2	Rm			0	0	1	0	0	0			Rn					Rd			0x4E202000
800	//	SSUBW	Rm			0	0	1	1	0	0			Rn					Rd			0x0E203000
801	//	SSUBW2	Rm			0	0	1	1	0	0			Rn					Rd			0x4E203000
802	//	ADDHN	Rm			0	1	0	0	0	0			Rn					Rd			0x0E204000
803	//	ADDHN2	Rm			0	1	0	0	0	0			Rn					Rd			0x4E204000
804	//	SABAL	Rm			0	1	0	1	0	0			Rn					Rd			0x0E205000
805	//	SABAL2	Rm			0	1	0	1	0	0			Rn					Rd			0x4E205000
806	//	SUBHN	Rm			0	1	1	0	0	0			Rn					Rd			0x0E206000
807	//	SUBHN2	Rm			0	1	1	0	0	0			Rn					Rd			0x4E206000
808	//	SABDL	Rm			0	1	1	1	0	0			Rn					Rd			0x0E207000
809	//	SABDL2	Rm			0	1	1	1	0	0			Rn					Rd			0x4E207000
810	//	SMLAL	Rm			1	0	0	0	0	0			Rn					Rd			0x0E208000
811	//	SMLAL2	Rm			1	0	0	0	0	0			Rn					Rd			0x4E208000
812	//	SQDMLAL	Rm			1	0	0	1	0	0			Rn					Rd			0x0E209000
813	//	SQDMLAL2	Rm			1	0	0	1	0	0			Rn					Rd			0x4E209000
814	//	SMLSL	Rm			1	0	1	0	0	0			Rn					Rd			0x0E20A000
815	//	SMLSL2	Rm			1	0	1	0	0	0			Rn					Rd			0x4E20A000
816	//	SQDMLSL	Rm			1	0	1	1	0	0			Rn					Rd			0x0E20B000
817	//	SQDMLSL2	Rm			1	0	1	1	0	0			Rn					Rd			0x4E20B000
818	//	SMULL	Rm			1	1	0	0	0	0			Rn					Rd			0x0E20C000
819	//	SMULL2	Rm			1	1	0	0	0	0			Rn					Rd			0x4E20C000
820	//	SQDMULL	Rm			1	1	0	1	0	0			Rn					Rd			0x0E20D000
821	//	SQDMULL2	Rm			1	1	0	1	0	0			Rn					Rd			0x4E20D000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
822	//	PMULL	Rm			1	1	1	0	0	0			Rn					Rd			0x0E20E000
823	//	PMULL2	Rm			1	1	1	0	0	0			Rn					Rd			0x4E20E000
824	//	UADDL	Rm			0	0	0	0	0	0			Rn					Rd			0x2E200000
825	//	UADDL2	Rm			0	0	0	0	0	0			Rn					Rd			0x6E200000
826	//	UADDW	Rm			0	0	0	1	0	0			Rn					Rd			0x2E201000
827	//	UADDW2	Rm			0	0	0	1	0	0			Rn					Rd			0x6E201000
828	//	USUBL	Rm			0	0	1	0	0	0			Rn					Rd			0x2E202000
829	//	USUBL2	Rm			0	0	1	0	0	0			Rn					Rd			0x6E202000
830	//	USUBW	Rm			0	0	1	1	0	0			Rn					Rd			0x2E203000
831	//	USUBW2	Rm			0	0	1	1	0	0			Rn					Rd			0x6E203000
832	//	RADDHN	Rm			0	1	0	0	0	0			Rn					Rd			0x2E204000
833	//	RADDHN2	Rm			0	1	0	0	0	0			Rn					Rd			0x6E204000
834	//	UABAL	Rm			0	1	0	1	0	0			Rn					Rd			0x2E205000
835	//	UABAL2	Rm			0	1	0	1	0	0			Rn					Rd			0x6E205000
836	//	RSUBHN	Rm			0	1	1	0	0	0			Rn					Rd			0x2E206000
837	//	RSUBHN2	Rm			0	1	1	0	0	0			Rn					Rd			0x6E206000
838	//	UABDL	Rm			0	1	1	1	0	0			Rn					Rd			0x2E207000
839	//	UABDL2	Rm			0	1	1	1	0	0			Rn					Rd			0x6E207000
840	//	UMLAL	Rm			1	0	0	0	0	0			Rn					Rd			0x2E208000
841	//	UMLAL2	Rm			1	0	0	0	0	0			Rn					Rd			0x6E208000
842	//	UMLSL	Rm			1	0	1	0	0	0			Rn					Rd			0x2E20A000
843	//	UMLSL2	Rm			1	0	1	0	0	0			Rn					Rd			0x6E20A000
844	//	UMULL	Rm			1	1	0	0	0	0			Rn					Rd			0x2E20C000
845	//	UMULL2	Rm			1	1	0	0	0	0			Rn					Rd			0x6E20C000
846	//	**AdvSIMD two-reg misc**	**0**	**0**		**opcode**				**1**	**0**			**Rn**					**Rd**			
847	//	REV64	0	0	0	0	0	0	0	1	0			Rn					Rd			0x0E200800
848	//	REV16	0	0	0	0	0	0	1	1	0			Rn					Rd			0x0E201800
849	//	SADDLP	0	0	0	0	0	1	0	1	0			Rn					Rd			0x0E202800
850	//	SUQADD	0	0	0	0	0	1	1	1	0			Rn					Rd			0x0E203800
851	//	CLS	0	0	0	0	1	0	0	1	0			Rn					Rd			0x0E204800
852	//	CNT	0	0	0	0	1	0	1	1	0			Rn					Rd			0x0E205800
853	//	SADALP	0	0	0	0	1	1	0	1	0			Rn					Rd			0x0E206800
854	//	SQABS	0	0	0	0	1	1	1	1	0			Rn					Rd			0x0E207800
855	//	CMGT	0	0	0	1	0	0	0	1	0			Rn					Rd			0x0E208800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
856	//	CMEQ	0	0	0	1	0	0	1	1	0			Rn					Rd			0x0E209800
857	//	CMLT	0	0	0	1	0	1	0	1	0			Rn					Rd			0x0E20A800
858	//	ABS	0	0	0	1	0	1	1	1	0			Rn					Rd			0x0E20B800
859	//	XTN	0	0	1	0	0	1	0	1	0			Rn					Rd			0x0E212800
860	//	XTN2	0	0	1	0	0	1	0	1	0			Rn					Rd			0x0E212800
861	//	SQXTN	0	0	1	0	1	0	0	1	0			Rn					Rd			0x0E214800
862	//	SQXTN2	0	0	1	0	1	0	0	1	0			Rn					Rd			0x0E214800
863	//	FCVTN	0	0	1	0	1	1	0	1	0			Rn					Rd			0x0E216800
864	//	FCVTN2	0	0	1	0	1	1	0	1	0			Rn					Rd			0x0E216800
865	//	FCVTL	0	0	1	0	1	1	1	1	0			Rn					Rd			0x0E217800
866	//	FCVTL2	0	0	1	0	1	1	1	1	0			Rn					Rd			0x0E217800
867	//	FRINTN	0	0	1	1	0	0	0	1	0			Rn					Rd			0x0E218800
868	//	FRINTM	0	0	1	1	0	0	1	1	0			Rn					Rd			0x0E219800
869	//	FCVTNS	0	0	1	1	0	1	0	1	0			Rn					Rd			0x0E21A800
870	//	FCVTMS	0	0	1	1	0	1	1	1	0			Rn					Rd			0x0E21B800
871	//	FCVTAS	0	0	1	1	1	0	0	1	0			Rn					Rd			0x0E21C800
872	//	SCVTF	0	0	1	1	1	0	1	1	0			Rn					Rd			0x0E21D800
873	//	FCMGT	0	0	0	1	1	0	0	1	0			Rn					Rd			0x0EA0C800
874	//	FCMEQ	0	0	0	1	1	0	1	1	0			Rn					Rd			0x0EA0D800
875	//	FCMLT	0	0	0	1	1	1	0	1	0			Rn					Rd			0x0EA0E800
876	//	FABS	0	0	0	1	1	1	1	1	0			Rn					Rd			0x0EA0F800
877	//	FRINTP	0	0	1	1	0	0	0	1	0			Rn					Rd			0x0EA18800
878	//	FRINTZ	0	0	1	1	0	0	1	1	0			Rn					Rd			0x0EA19800
879	//	FCVTPS	0	0	1	1	0	1	0	1	0			Rn					Rd			0x0EA1A800
880	//	FCVTZS	0	0	1	1	0	1	1	1	0			Rn					Rd			0x0EA1B800
881	//	URECPE	0	0	1	1	1	0	0	1	0			Rn					Rd			0x0EA1C800
882	//	FRECPE	0	0	1	1	1	0	1	1	0			Rn					Rd			0x0EA1D800
883	//	REV32	0	0	0	0	0	0	0	1	0			Rn					Rd			0x2E200800
884	//	UADDLP	0	0	0	0	0	1	0	1	0			Rn					Rd			0x2E202800
885	//	USQADD	0	0	0	0	0	1	1	1	0			Rn					Rd			0x2E203800
886	//	CLZ	0	0	0	0	1	0	0	1	0			Rn					Rd			0x2E204800
887	//	UADALP	0	0	0	0	1	1	0	1	0			Rn					Rd			0x2E206800
888	//	SQNEG	0	0	0	0	1	1	1	1	0			Rn					Rd			0x2E207800
889	//	CMGE	0	0	0	1	0	0	0	1	0			Rn					Rd			0x2E208800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
890	//	CMLE	0	0	0	1	0	0	1	1	0			Rn					Rd			0x2E209800
891	//	NEG	0	0	0	1	0	1	1	1	0			Rn					Rd			0x2E20B800
892	//	SQXTUN	0	0	1	0	0	1	0	1	0			Rn					Rd			0x2E212800
893	//	SQXTUN2	0	0	1	0	0	1	0	1	0			Rn					Rd			0x2E212800
894	//	SHLL	0	0	1	0	0	1	1	1	0			Rn					Rd			0x2E213800
895	//	SHLL2	0	0	1	0	0	1	1	1	0			Rn					Rd			0x2E213800
896	//	UQXTN	0	0	1	0	1	0	0	1	0			Rn					Rd			0x2E214800
897	//	UQXTN2	0	0	1	0	1	0	0	1	0			Rn					Rd			0x2E214800
898	//	FCVTXN	0	0	1	0	1	1	0	1	0			Rn					Rd			0x2E216800
899	//	FCVTXN2	0	0	1	0	1	1	0	1	0			Rn					Rd			0x2E216800
900	//	FRINTA	0	0	1	1	0	0	0	1	0			Rn					Rd			0x2E218800
901	//	FRINTX	0	0	1	1	0	0	1	1	0			Rn					Rd			0x2E219800
902	//	FCVTNU	0	0	1	1	0	1	0	1	0			Rn					Rd			0x2E21A800
903	//	FCVTMU	0	0	1	1	0	1	1	1	0			Rn					Rd			0x2E21B800
904	//	FCVTAU	0	0	1	1	1	0	0	1	0			Rn					Rd			0x2E21C800
905	//	UCVTF	0	0	1	1	1	0	1	1	0			Rn					Rd			0x2E21D800
906	//	NOT	0	0	0	0	1	0	1	1	0			Rn					Rd			0x2E205800
907	//	RBIT	0	0	0	0	1	0	1	1	0			Rn					Rd			0x2E605800
908	//	FCMGE	0	0	0	1	1	0	0	1	0			Rn					Rd			0x2EA0C800
909	//	FCMLE	0	0	0	1	1	0	1	1	0			Rn					Rd			0x2EA0D800
910	//	FNEG	0	0	0	1	1	1	1	1	0			Rn					Rd			0x2EA0F800
911	//	FRINTI	0	0	1	1	0	0	1	1	0			Rn					Rd			0x2EA19800
912	//	FCVTPU	0	0	1	1	0	1	0	1	0			Rn					Rd			0x2EA1A800
913	//	FCVTZU	0	0	1	1	0	1	1	1	0			Rn					Rd			0x2EA1B800
914	//	URSQRTE	0	0	1	1	1	0	0	1	0			Rn					Rd			0x2EA1C800
915	//	FRSQRTE	0	0	1	1	1	0	1	1	0			Rn					Rd			0x2EA1D800
916	//	FSQRT	0	0	1	1	1	1	1	1	0			Rn					Rd			0x2EA1F800
917	//	**AdvSIMD across lanes**	**0**	**0**			**opcode**			**1**	**0**			**Rn**					**Rd**			
918	//	SADDLV	0	0	0	0	0	1	1	1	0			Rn					Rd			0x0E303800
919	//	SMAXV	0	0	0	1	0	1	0	1	0			Rn					Rd			0x0E30A800
920	//	SMINV	0	0	1	1	0	1	0	1	0			Rn					Rd			0x0E31A800
921	//	ADDV	0	0	1	1	0	1	1	1	0			Rn					Rd			0x0E31B800
922	//	UADDLV	0	0	0	0	0	1	1	1	0			Rn					Rd			0x2E303800
923	//	UMAXV	0	0	0	1	0	1	0	1	0			Rn					Rd			0x2E30A800

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
924	//	UMINV	0	0	1	1	0	1	0	1	0			Rn					Rd			0x2E31A800
925	//	FMAXNMV	0	0	0	1	1	0	0	1	0			Rn					Rd			0x2E30C800
926	//	FMAXV	0	0	0	1	1	1	1	1	0			Rn					Rd			0x2E30F800
927	//	FMINNMV	0	0	0	1	1	0	0	1	0			Rn					Rd			0x2EB0C800
928	//	FMINV	0	0	0	1	1	1	1	1	0			Rn					Rd			0x2EB0F800
929	//	**AdvSIMD copy**	**imm5**			**0**		**imm4**			**1**			**Rn**					**Rd**			
930	//	DUP	-	-	-	0	0	0	0	0	1			Rn					Rd			0x0E000400
931	//	DUP	-	-	-	0	0	0	0	1	1			Rn					Rd			0x0E000C00
932	//	SMOV	-	-	-	0	0	1	0	1	1			Rn					Rd			0x0E002C00
933	//	UMOV	-	-	-	0	0	1	1	1	1			Rn					Rd			0x0E003C00
934	//	INS	-	-	-	0	0	0	1	1	1			Rn					Rd			0x4E001C00
935	//	SMOV	-	-	-	0	0	1	0	1	1			Rn					Rd			0x4E002C00
936	//	UMOV	-	-	-	0	0	1	1	1	1			Rn					Rd			0x4E003C00
937	//	INS	-	-	-	0	-	-	-	-	1			Rn					Rd			0x6E000400
938	//	**AdvSIMD vector x indexed element**	**Rm**				**opcode**			**H**	**0**			**Rn**					**Rd**			
939	//	SMLAL	Rm			0	0	1	0	H	0			Rn					Rd			0x0F002000
940	//	SMLAL2	Rm			0	0	1	0	H	0			Rn					Rd			0x0F002000
941	//	SQDMLAL	Rm			0	0	1	1	H	0			Rn					Rd			0x0F003000
942	//	SQDMLAL2	Rm			0	0	1	1	H	0			Rn					Rd			0x0F003000
943	//	SMLSL	Rm			0	1	1	0	H	0			Rn					Rd			0x0F006000
944	//	SMLSL2	Rm			0	1	1	0	H	0			Rn					Rd			0x0F006000
945	//	SQDMLSL	Rm			0	1	1	1	H	0			Rn					Rd			0x0F007000
946	//	SQDMLSL2	Rm			0	1	1	1	H	0			Rn					Rd			0x0F007000
947	//	MUL	Rm			1	0	0	0	H	0			Rn					Rd			0x0F008000
948	//	SMULL	Rm			1	0	1	0	H	0			Rn					Rd			0x0F00A000
949	//	SMULL2	Rm			1	0	1	0	H	0			Rn					Rd			0x0F00A000
950	//	SQDMULL	Rm			1	0	1	1	H	0			Rn					Rd			0x0F00B000
951	//	SQDMULL2	Rm			1	0	1	1	H	0			Rn					Rd			0x0F00B000
952	//	SQDMULH	Rm			1	1	0	0	H	0			Rn					Rd			0x0F00C000
953	//	SQRDMULH	Rm			1	1	0	1	H	0			Rn					Rd			0x0F00D000
954	//	FMLA	Rm			0	0	0	1	H	0			Rn					Rd			0x0F801000
955	//	FMLS	Rm			0	1	0	1	H	0			Rn					Rd			0x0F805000
956	//	FMUL	Rm			1	0	0	1	H	0			Rn					Rd			0x0F809000
957	//	MLA	Rm			0	0	0	0	H	0			Rn					Rd			0x2F000000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
958	//	UMLAL	Rm			0	0	1	0	H	0			Rn					Rd			0x2F002000
959	//	UMLAL2	Rm			0	0	1	0	H	0			Rn					Rd			0x2F002000
960	//	MLS	Rm			0	1	0	0	H	0			Rn					Rd			0x2F004000
961	//	UMLSL	Rm			0	1	1	0	H	0			Rn					Rd			0x2F006000
962	//	UMLSL2	Rm			0	1	1	0	H	0			Rn					Rd			0x2F006000
963	//	UMULL	Rm			1	0	1	0	H	0			Rn					Rd			0x2F00A000
964	//	UMULL2	Rm			1	0	1	0	H	0			Rn					Rd			0x2F00A000
965	//	FMULX	Rm			1	0	0	1	H	0			Rn					Rd			0x2F809000
966	//	**AdvSIMD modified immediate**	**a**	**b**	**c**		**cmode**			**o2**	**1**	**d**	**e**	**f**	**g**	**h**			**Rd**			
967	//	MOVI	a	b	c	0	x	x	0	0	1	d	e	f	g	h			Rd			0x0F000400
968	//	ORR	a	b	c	0	x	x	1	0	1	d	e	f	g	h			Rd			0x0F001400
969	//	MOVI	a	b	c	1	0	x	0	0	1	d	e	f	g	h			Rd			0x0F008400
970	//	ORR	a	b	c	1	0	x	1	0	1	d	e	f	g	h			Rd			0x0F009400
971	//	MOVI	a	b	c	1	1	0	x	0	1	d	e	f	g	h			Rd			0x0F00C400
972	//	MOVI	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd			0x0F00E400
973	//	FMOV	a	b	c	1	1	1	1	0	1	d	e	f	g	h			Rd			0x0F00F400
974	//	MVNI	a	b	c	0	x	x	0	0	1	d	e	f	g	h			Rd			0x2F000400
975	//	BIC	a	b	c	0	x	x	1	0	1	d	e	f	g	h			Rd			0x2F001400
976	//	MVNI	a	b	c	1	0	x	0	0	1	d	e	f	g	h			Rd			0x2F008400
977	//	BIC	a	b	c	1	0	x	1	0	1	d	e	f	g	h			Rd			0x2F009400
978	//	MVNI	a	b	c	1	1	0	x	0	1	d	e	f	g	h			Rd			0x2F00C400
979	//	MOVI	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd			0x2F00E400
980	//	MOVI	a	b	c	1	1	1	0	0	1	d	e	f	g	h			Rd			0x6F00E400
981	//	FMOV	a	b	c	1	1	1	1	0	1	d	e	f	g	h			Rd			0x6F00F400
982	//	**AdvSIMD shift by immediate**	**immb**				**opcode**				**1**			**Rn**					**Rd**			
983	//	SSHR	immb			0	0	0	0	0	1			Rn					Rd			0x0F000400
984	//	SSRA	immb			0	0	0	1	0	1			Rn					Rd			0x0F001400
985	//	SRSHR	immb			0	0	1	0	0	1			Rn					Rd			0x0F002400
986	//	SRSRA	immb			0	0	1	1	0	1			Rn					Rd			0x0F003400
987	//	SHL	immb			0	1	0	1	0	1			Rn					Rd			0x0F005400
988	//	SQSHL	immb			0	1	1	1	0	1			Rn					Rd			0x0F007400
989	//	SHRN	immb			1	0	0	0	0	1			Rn					Rd			0x0F008400
990	//	SHRN2	immb			1	0	0	0	0	1			Rn					Rd			0x0F008400
991	//	RSHRN	immb			1	0	0	0	1	1			Rn					Rd			0x0F008C00

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
992	//	RSHRN2		immb		1	0	0	0	1	1			Rn					Rd			0x0F008C00
993	//	SQSHRN		immb		1	0	0	1	0	1			Rn					Rd			0x0F009400
994	//	SQSHRN2		immb		1	0	0	1	0	1			Rn					Rd			0x0F009400
995	//	SQRSHRN		immb		1	0	0	1	1	1			Rn					Rd			0x0F009C00
996	//	SQRSHRN2		immb		1	0	0	1	1	1			Rn					Rd			0x0F009C00
997	//	SSHLL		immb		1	0	1	0	0	1			Rn					Rd			0x0F00A400
998	//	SSHLL2		immb		1	0	1	0	0	1			Rn					Rd			0x0F00A400
999	//	SCVTF		immb		1	1	1	0	0	1			Rn					Rd			0x0F00E400
100(	//	FCVTZS		immb		1	1	1	1	1	1			Rn					Rd			0x0F00FC00
1001	//	USHR		immb		0	0	0	0	0	1			Rn					Rd			0x2F000400
1002	//	USRA		immb		0	0	0	1	0	1			Rn					Rd			0x2F001400
1003	//	URSHR		immb		0	0	1	0	0	1			Rn					Rd			0x2F002400
1004	//	URSRA		immb		0	0	1	1	0	1			Rn					Rd			0x2F003400
1005	//	SRI		immb		0	1	0	0	0	1			Rn					Rd			0x2F004400
1006	//	SLI		immb		0	1	0	1	0	1			Rn					Rd			0x2F005400
1007	//	SQSHLU		immb		0	1	1	0	0	1			Rn					Rd			0x2F006400
1008	//	UQSHL		immb		0	1	1	1	0	1			Rn					Rd			0x2F007400
1009	//	SQSHRUN		immb		1	0	0	0	0	1			Rn					Rd			0x2F008400
101(	//	SQSHRUN2		immb		1	0	0	0	0	1			Rn					Rd			0x2F008400
1011	//	SQRSHRUN		immb		1	0	0	0	1	1			Rn					Rd			0x2F008C00
1012	//	SQRSHRUN2		immb		1	0	0	0	1	1			Rn					Rd			0x2F008C00
1013	//	UQSHRN		immb		1	0	0	1	0	1			Rn					Rd			0x2F009400
1014	//	UQRSHRN		immb		1	0	0	1	1	1			Rn					Rd			0x2F009C00
1015	//	UQRSHRN2		immb		1	0	0	1	1	1			Rn					Rd			0x2F009C00
1016	//	USHLL		immb		1	0	1	0	0	1			Rn					Rd			0x2F00A400
1017	//	USHLL2		immb		1	0	1	0	0	1			Rn					Rd			0x2F00A400
1018	//	UCVTF		immb		1	1	1	0	0	1			Rn					Rd			0x2F00E400
1019	//	FCVTZU		immb		1	1	1	1	1	1			Rn					Rd			0x2F00FC00
102(	//	**AdvSIMD TBL/TBX**	**Rm**			**0**		**len**		**op**	**0**	**0**		**Rn**					**Rd**			
1021	//	TBL	Rm			0	0	0	0	0	0			Rn					Rd			0x0E000000
1022	//	TBX	Rm			0	0	0	1	0	0			Rn					Rd			0x0E001000
1023	//	TBL	Rm			0	0	1	0	0	0			Rn					Rd			0x0E002000
1024	//	TBX	Rm			0	0	1	1	0	0			Rn					Rd			0x0E003000
1025	//	TBL	Rm			0	1	0	0	0	0			Rn					Rd			0x0E004000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
102	//	TBX	Rm			0	1	0	1	0	0			Rn					Rd			0x0E005000
102	//	TBL	Rm			0	1	1	0	0	0			Rn					Rd			0x0E006000
102	//	TBX	Rm			0	1	1	1	0	0			Rn					Rd			0x0E007000
102	//	**AdvSIMD ZIP/UZP/TRN**	**Rm**			**0**	**opcode**			**1**	**0**			**Rn**					**Rd**			
103	//	UZP1	Rm			0	0	0	1	1	0			Rn					Rd			0x0E001800
103	//	TRN1	Rm			0	0	1	0	1	0			Rn					Rd			0x0E002800
103	//	ZIP1	Rm			0	0	1	1	1	0			Rn					Rd			0x0E003800
103	//	UZP2	Rm			0	1	0	1	1	0			Rn					Rd			0x0E005800
103	//	TRN2	Rm			0	1	1	0	1	0			Rn					Rd			0x0E006800
103	//	ZIP2	Rm			0	1	1	1	1	0			Rn					Rd			0x0E007800
103	//	**AdvSIMD EXT**	**Rm**			**0**		**imm4**			**0**			**Rn**					**Rd**			
103	//	EXT	Rm			0		imm4			0			Rn					Rd			0x2E000000
103	//	**Loads and stores**																				
103	//	**AdvSIMD load/store multiple structures**	**0**	**0**	**0**		**opcode**			**size**				**Rn**					**Rt**			
104	//	ST4	0	0	0	0	0	0	0	size				Rn					Rt			0x0C000000
104	//	ST1	0	0	0	0	0	1	0	size				Rn					Rt			0x0C002000
104	//	ST3	0	0	0	0	1	0	0	size				Rn					Rt			0x0C004000
104	//	ST1	0	0	0	0	1	1	0	size				Rn					Rt			0x0C006000
104	//	ST1	0	0	0	0	1	1	1	size				Rn					Rt			0x0C007000
104	//	ST2	0	0	0	1	0	0	0	size				Rn					Rt			0x0C008000
104	//	ST1	0	0	0	1	0	1	0	size				Rn					Rt			0x0C00A000
104	//	LD4	0	0	0	0	0	0	0	size				Rn					Rt			0x0C400000
104	//	LD1	0	0	0	0	0	1	0	size				Rn					Rt			0x0C402000
104	//	LD3	0	0	0	0	1	0	0	size				Rn					Rt			0x0C404000
105	//	LD1	0	0	0	0	1	1	0	size				Rn					Rt			0x0C406000
105	//	LD1	0	0	0	0	1	1	1	size				Rn					Rt			0x0C407000
105	//	LD2	0	0	0	1	0	0	0	size				Rn					Rt			0x0C408000
105	//	LD1	0	0	0	1	0	1	0	size				Rn					Rt			0x0C40A000
105	//	**AdvSIMD load/store multiple structures (pc**	**Rm**				**opcode**			**size**				**Rn**					**Rt**			
105	//	ST4	Rm			0	0	0	0	size				Rn					Rt			0x0C800000
105	//	ST1	Rm			0	0	1	0	size				Rn					Rt			0x0C802000
105	//	ST3	Rm			0	1	0	0	size				Rn					Rt			0x0C804000
105	//	ST1	Rm			0	1	1	0	size				Rn					Rt			0x0C806000
105	//	ST1	Rm			0	1	1	1	size				Rn					Rt			0x0C807000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
106(	//	ST2	Rm			1	0	0	0	size				Rn					Rt			0x0C808000
1061	//	ST1	Rm			1	0	1	0	size				Rn					Rt			0x0C80A000
1062	//	ST4	1	1	1	0	0	0	0	size				Rn					Rt			0x0C9F0000
1063	//	ST1	1	1	1	0	0	1	0	size				Rn					Rt			0x0C9F2000
1064	//	ST3	1	1	1	0	1	0	0	size				Rn					Rt			0x0C9F4000
1065	//	ST1	1	1	1	0	1	1	0	size				Rn					Rt			0x0C9F6000
1066	//	ST1	1	1	1	0	1	1	1	size				Rn					Rt			0x0C9F7000
1067	//	ST2	1	1	1	1	0	0	0	size				Rn					Rt			0x0C9F8000
1068	//	ST1	1	1	1	1	0	1	0	size				Rn					Rt			0x0C9FA000
1069	//	LD4	Rm			0	0	0	0	size				Rn					Rt			0x0CC00000
107(	//	LD1	Rm			0	0	1	0	size				Rn					Rt			0x0CC02000
1071	//	LD3	Rm			0	1	0	0	size				Rn					Rt			0x0CC04000
1072	//	LD1	Rm			0	1	1	0	size				Rn					Rt			0x0CC06000
1073	//	LD1	Rm			0	1	1	1	size				Rn					Rt			0x0CC07000
1074	//	LD2	Rm			1	0	0	0	size				Rn					Rt			0x0CC08000
1075	//	LD1	Rm			1	0	1	0	size				Rn					Rt			0x0CC0A000
1076	//	LD4	1	1	1	0	0	0	0	size				Rn					Rt			0x0CDF0000
1077	//	LD1	1	1	1	0	0	1	0	size				Rn					Rt			0x0CDF2000
1078	//	LD3	1	1	1	0	1	0	0	size				Rn					Rt			0x0CDF4000
1079	//	LD1	1	1	1	0	1	1	0	size				Rn					Rt			0x0CDF6000
108(	//	LD1	1	1	1	0	1	1	1	size				Rn					Rt			0x0CDF7000
1081	//	LD2	1	1	1	1	0	0	0	size				Rn					Rt			0x0CDF8000
1082	//	LD1	1	1	1	1	0	1	0	size				Rn					Rt			0x0CDFA000
1083	//	**AdvSIMD load/store single structure**	**0**	**0**	**0**	**opcode**			**S**	**size**				**Rn**					**Rt**			
1084	//	ST1	0	0	0	0	0	0	-	-	-			Rn					Rt			0x0D000000
1085	//	ST3	0	0	0	0	0	1	-	-	-			Rn					Rt			0x0D002000
1086	//	ST1	0	0	0	0	1	0	-	x	0			Rn					Rt			0x0D004000
1087	//	ST3	0	0	0	0	1	1	-	x	0			Rn					Rt			0x0D006000
1088	//	ST1	0	0	0	1	0	0	-	0	0			Rn					Rt			0x0D008000
1089	//	ST1	0	0	0	1	0	0	0	0	1			Rn					Rt			0x0D008400
109(	//	ST3	0	0	0	1	0	1	-	0	0			Rn					Rt			0x0D00A000
1091	//	ST3	0	0	0	1	0	1	0	0	1			Rn					Rt			0x0D00A400
1092	//	ST2	0	0	0	0	0	0	-	-	-			Rn					Rt			0x0D200000
1093	//	ST4	0	0	0	0	0	1	-	-	-			Rn					Rt			0x0D202000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
1094	//	ST2	0	0	0	0	1	0	-	x	0			Rn					Rt			0x0D204000
1095	//	ST4	0	0	0	0	1	1	-	x	0			Rn					Rt			0x0D206000
1096	//	ST2	0	0	0	1	0	0	-	0	0			Rn					Rt			0x0D208000
1097	//	ST2	0	0	0	1	0	0	0	0	1			Rn					Rt			0x0D208400
1098	//	ST4	0	0	0	1	0	1	-	0	0			Rn					Rt			0x0D20A000
1099	//	ST4	0	0	0	1	0	1	0	0	1			Rn					Rt			0x0D20A400
1100	//	LD1	0	0	0	0	0	0	-	-	-			Rn					Rt			0x0D400000
1101	//	LD3	0	0	0	0	0	1	-	-	-			Rn					Rt			0x0D402000
1102	//	LD1	0	0	0	0	1	0	-	x	0			Rn					Rt			0x0D404000
1103	//	LD3	0	0	0	0	1	1	-	x	0			Rn					Rt			0x0D406000
1104	//	LD1	0	0	0	1	0	0	-	0	0			Rn					Rt			0x0D408000
1105	//	LD1	0	0	0	1	0	0	0	0	1			Rn					Rt			0x0D408400
1106	//	LD3	0	0	0	1	0	1	-	0	0			Rn					Rt			0x0D40A000
1107	//	LD3	0	0	0	1	0	1	0	0	1			Rn					Rt			0x0D40A400
1108	//	LD1R	0	0	0	1	1	0	0	-	-			Rn					Rt			0x0D40C000
1109	//	LD3R	0	0	0	1	1	1	0	-	-			Rn					Rt			0x0D40E000
1110	//	LD2	0	0	0	0	0	0	-	-	-			Rn					Rt			0x0D600000
1111	//	LD4	0	0	0	0	0	1	-	-	-			Rn					Rt			0x0D602000
1112	//	LD2	0	0	0	0	1	0	-	x	0			Rn					Rt			0x0D604000
1113	//	LD4	0	0	0	0	1	1	-	x	0			Rn					Rt			0x0D606000
1114	//	LD2	0	0	0	1	0	0	-	0	0			Rn					Rt			0x0D608000
1115	//	LD2	0	0	0	1	0	0	0	0	1			Rn					Rt			0x0D608400
1116	//	LD4	0	0	0	1	0	1	-	0	0			Rn					Rt			0x0D60A000
1117	//	LD4	0	0	0	1	0	1	0	0	1			Rn					Rt			0x0D60A400
1118	//	LD2R	0	0	0	1	1	0	0	-	-			Rn					Rt			0x0D60C000
1119	//	LD4R	0	0	0	1	1	1	0	-	-			Rn					Rt			0x0D60E000
1120	//	**AdvSIMD load/store single structure (post-**	Rm			**opcode**			**S**	**size**				**Rn**					**Rt**			
1121	//	ST1	Rm			0	0	0	-	-	-			Rn					Rt			0x0D800000
1122	//	ST3	Rm			0	0	1	-	-	-			Rn					Rt			0x0D802000
1123	//	ST1	Rm			0	1	0	-	x	0			Rn					Rt			0x0D804000
1124	//	ST3	Rm			0	1	1	-	x	0			Rn					Rt			0x0D806000
1125	//	ST1	Rm			1	0	0	-	0	0			Rn					Rt			0x0D808000
1126	//	ST1	Rm			1	0	0	0	0	1			Rn					Rt			0x0D808400
1127	//	ST3	Rm			1	0	1	-	0	0			Rn					Rt			0x0D80A000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
1128	//	ST3	Rm			1	0	1	0	0	1			Rn					Rt			0x0D80A400
1129	//	ST1	1	1	1	0	0	0	-	-	-			Rn					Rt			0x0D9F0000
1130	//	ST3	1	1	1	0	0	1	-	-	-			Rn					Rt			0x0D9F2000
1131	//	ST1	1	1	1	0	1	0	-	x	0			Rn					Rt			0x0D9F4000
1132	//	ST3	1	1	1	0	1	1	-	x	0			Rn					Rt			0x0D9F6000
1133	//	ST1	1	1	1	1	0	0	-	0	0			Rn					Rt			0x0D9F8000
1134	//	ST1	1	1	1	1	0	0	0	0	1			Rn					Rt			0x0D9F8400
1135	//	ST3	1	1	1	1	0	1	-	0	0			Rn					Rt			0x0D9FA000
1136	//	ST3	1	1	1	1	0	1	0	0	1			Rn					Rt			0x0D9FA400
1137	//	ST2	Rm			0	0	0	-	-	-			Rn					Rt			0x0DA00000
1138	//	ST4	Rm			0	0	1	-	-	-			Rn					Rt			0x0DA02000
1139	//	ST2	Rm			0	1	0	-	x	0			Rn					Rt			0x0DA04000
1140	//	ST4	Rm			0	1	1	-	x	0			Rn					Rt			0x0DA06000
1141	//	ST2	Rm			1	0	0	-	0	0			Rn					Rt			0x0DA08000
1142	//	ST2	Rm			1	0	0	0	0	1			Rn					Rt			0x0DA08400
1143	//	ST4	Rm			1	0	1	-	0	0			Rn					Rt			0x0DA0A000
1144	//	ST4	Rm			1	0	1	0	0	1			Rn					Rt			0x0DA0A400
1145	//	ST2	1	1	1	0	0	0	-	-	-			Rn					Rt			0x0DBF0000
1146	//	ST4	1	1	1	0	0	1	-	-	-			Rn					Rt			0x0DBF2000
1147	//	ST2	1	1	1	0	1	0	-	x	0			Rn					Rt			0x0DBF4000
1148	//	ST4	1	1	1	0	1	1	-	x	0			Rn					Rt			0x0DBF6000
1149	//	ST2	1	1	1	1	0	0	-	0	0			Rn					Rt			0x0DBF8000
1150	//	ST2	1	1	1	1	0	0	0	0	1			Rn					Rt			0x0DBF8400
1151	//	ST4	1	1	1	1	0	1	-	0	0			Rn					Rt			0x0DBFA000
1152	//	ST4	1	1	1	1	0	1	0	0	1			Rn					Rt			0x0DBFA400
1153	//	LD1	Rm			0	0	0	-	-	-			Rn					Rt			0x0DC00000
1154	//	LD3	Rm			0	0	1	-	-	-			Rn					Rt			0x0DC02000
1155	//	LD1	Rm			0	1	0	-	x	0			Rn					Rt			0x0DC04000
1156	//	LD3	Rm			0	1	1	-	x	0			Rn					Rt			0x0DC06000
1157	//	LD1	Rm			1	0	0	-	0	0			Rn					Rt			0x0DC08000
1158	//	LD1	Rm			1	0	0	0	0	1			Rn					Rt			0x0DC08400
1159	//	LD3	Rm			1	0	1	-	0	0			Rn					Rt			0x0DC0A000
1160	//	LD3	Rm			1	0	1	0	0	1			Rn					Rt			0x0DC0A400
1161	//	LD1R	Rm			1	1	0	0	-	-			Rn					Rt			0x0DC0C000

1	in_use	Opcode	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Binary
1162	//	LD3R	Rm			1	1	1	0	-	-			Rn					Rt			0x0DC0E000
1163	//	LD1	1	1	1	0	0	0	-	-	-			Rn					Rt			0x0DDF0000
1164	//	LD3	1	1	1	0	0	1	-	-	-			Rn					Rt			0x0DDF2000
1165	//	LD1	1	1	1	0	1	0	-	x	0			Rn					Rt			0x0DDF4000
1166	//	LD3	1	1	1	0	1	1	-	x	0			Rn					Rt			0x0DDF6000
1167	//	LD1	1	1	1	1	0	0	-	0	0			Rn					Rt			0x0DDF8000
1168	//	LD1	1	1	1	1	0	0	0	0	1			Rn					Rt			0x0DDF8400
1169	//	LD3	1	1	1	1	0	1	-	0	0			Rn					Rt			0x0DDFA000
1170	//	LD3	1	1	1	1	0	1	0	0	1			Rn					Rt			0x0DDFA400
1171	//	LD1R	1	1	1	1	1	0	0	-	-			Rn					Rt			0x0DDFC000
1172	//	LD3R	1	1	1	1	1	1	0	-	-			Rn					Rt			0x0DDFE000
1173	//	LD2	Rm			0	0	0	-	-	-			Rn					Rt			0x0DE00000
1174	//	LD4	Rm			0	0	1	-	-	-			Rn					Rt			0x0DE02000
1175	//	LD2	Rm			0	1	0	-	x	0			Rn					Rt			0x0DE04000
1176	//	LD4	Rm			0	1	1	-	x	0			Rn					Rt			0x0DE06000
1177	//	LD2	Rm			1	0	0	-	0	0			Rn					Rt			0x0DE08000
1178	//	LD2	Rm			1	0	0	0	0	1			Rn					Rt			0x0DE08400
1179	//	LD4	Rm			1	0	1	-	0	0			Rn					Rt			0x0DE0A000
1180	//	LD4	Rm			1	0	1	0	0	1			Rn					Rt			0x0DE0A400
1181	//	LD2R	Rm			1	1	0	0	-	-			Rn					Rt			0x0DE0C000
1182	//	LD4R	Rm			1	1	1	0	-	-			Rn					Rt			0x0DE0E000
1183	//	LD2	1	1	1	0	0	0	-	-	-			Rn					Rt			0x0DFF0000
1184	//	LD4	1	1	1	0	0	1	-	-	-			Rn					Rt			0x0DFF2000
1185	//	LD2	1	1	1	0	1	0	-	x	0			Rn					Rt			0x0DFF4000
1186	//	LD4	1	1	1	0	1	1	-	x	0			Rn					Rt			0x0DFF6000
1187	//	LD2	1	1	1	1	0	0	-	0	0			Rn					Rt			0x0DFF8000
1188	//	LD2	1	1	1	1	0	0	0	0	1			Rn					Rt			0x0DFF8400
1189	//	LD4	1	1	1	1	0	1	-	0	0			Rn					Rt			0x0DFFA000
1190	//	LD4	1	1	1	1	0	1	0	0	1			Rn					Rt			0x0DFFA400
1191	//	LD2R	1	1	1	1	1	0	0	-	-			Rn					Rt			0x0DFFC000
1192	//	LD4R	1	1	1	1	1	1	0	-	-			Rn					Rt			0x0DFFE000

1		**in_use Opcode**	[redacted]	**NAME**
2		**UNALLOCATED**		
3		BAD		bad,
4		**Branch,exception generation and syst**		
5		**Compare _ Branch (immediate)**		
6		CBZ		cbz_W,
7		CBNZ		cbnz_W,
8		CBZ		cbz_X,
9		CBNZ		cbnz_X,
10		**Test bit & branch (immediate)**		
11		TBZ		tbz,
12		TBNZ		tbnz,
13		**Conditional branch (immediate)**		
14		B_cond		b_cond,
15		**Exception generation**		
16	//	SVC		svc,
17	//	HVC		hvc,
18	//	SMC		smc,
19		BRK		brk,
20	//	HLT		hlt,
21	//	DCPS1		dcps1,
22	//	DCPS2		dcps2,
23	//	DCPS3		dcps3,
24	//	**System**		
25	//	MSR		msr_imm,
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 (register)**		
36		BR		br,
37		BLR		blr,

	in_use	Opcode	[redacted]	NAME
38		RET		ret,
39	//	ERET		eret,
40	//	DRPS		drps,
41	//	**Unconditional branch (immediate)**		
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		stxr_W,
59		STLXR		stlxr_W,
60		STXP		stxp_W,
61		STLXP		stlxp_W,
62		LDXR		ldxr_W,
63		LDAXR		ldaxr_W,
64		LDXP		ldxp_W,
65		LDAXP		ldaxp_W,
66		STLR		stlr_W,
67		LDAR		ldar_W,
68		STXR		stxr_X,
69		STLXR		stlxr_X,
70		STXP		stxp_X,
71		STLXP		stlxp_X,
72		LDXR		ldxr_X,
73		LDAXR		ldaxr_X,
74		LDXP		ldxp_X,

1	in_use	Opcode		NAME
75		LDAXP		ldaxp_X,
76		STLR		stlr_X,
77		LDAR		ldar_X,
78		**Load register (literal)**		
79		LDR		ldr_W,
80		LDR		ldr_S,
81		LDR		ldr_X,
82		LDR		ldr_D,
83		LDRSW		ldrsw,
84		LDR		ldr_Q,
85		PRFM		prfm,
86		**Load/store no-allocate pair (offset)**		
87		STNP		stnp_W,
88		LDNP		ldnp_W,
89		STNP		stnp_S,
90		LDNP		ldnp_S,
91		STNP		stnp_D,
92		LDNP		ldnp_D,
93		STNP		stnp_X,
94		LDNP		ldnp_X,
95		STNP		stnp_Q,
96		LDNP		ldnp_Q,
97		**Load/store register pair (post-indexed)**		
98		STP		stp_post_W,
99		LDP		ldp_post_W,
100		STP		stp_post_S,
101		LDP		ldp_post_S,
102		LDPSW		ldpsw_post,
103		STP		stp_post_D,
104		LDP		ldp_post_D,
105		STP		stp_post_X,
106		LDP		ldp_post_X,
107		STP		stp_post_Q,
108		LDP		ldp_post_Q,
109		**Load/store register pair (offset)**		

1	in_use	Opcode		NAME
110		STP		stp_off_W,
111		LDP		ldp_off_W,
112		STP		stp_off_S,
113		LDP		ldp_off_S,
114		LDPSW		ldpsw_off,
115		STP		stp_off_D,
116		LDP		ldp_off_D,
117		STP		stp_off_X,
118		LDP		ldp_off_X,
119		STP		stp_off_Q,
120		LDP		ldp_off_Q,
121		**Load/store register pair (pre-indexed)**		
122		STP		stp_pre_W,
123		LDP		ldp_pre_W,
124		STP		stp_pre_S,
125		LDP		ldp_pre_S,
126		LDPSW		ldpsw_pre,
127		STP		stp_pre_D,
128		LDP		ldp_pre_D,
129		STP		stp_pre_X,
130		LDP		ldp_pre_X,
131		STP		stp_pre_Q,
132		LDP		ldp_pre_Q,
133		**Load/store register (unscaled immediate)**		
134		STURB		sturb,
135		LDURB		ldurb,
136		LDURSB		ldursb_X,
137		LDURSB		ldursb_W,
138		STUR		stur_B,
139		LDUR		ldur_B,
140		STUR		stur_Q,
141		LDUR		ldur_Q,
142		STURH		sturh,
143		LDURH		ldurh,
144		LDURSH		ldursh_X,
145		LDURSH		ldursh_W,
146		STUR		stur_H,
147		LDUR		ldur_H,

1	in_use	Opcode	[redacted]	NAME
148		STUR		stur_W,
149		LDUR		ldur_W,
150		LDURSW		ldursw,
151		STUR		stur_S,
152		LDUR		ldur_S,
153		STUR		stur_X,
154		LDUR		ldur_X,
155		PRFUM		prfum,
156		STUR		stur_D,
157		LDUR		ldur_D,
158		**Load/store register (immediate post-indexe**		
159		STRB		strb_post,
160		LDRB		ldrb_post,
161		LDRSB		ldrsb_post_X,
162		LDRSB		ldrsb_post_W,
163		STR		str_post_B,
164		LDR		ldr_post_B,
165		STR		str_post_Q,
166		LDR		ldr_post_Q,
167		STRH		strh_post,
168		LDRH		ldrh_post,
169		LDRSH		ldrsh_post_X,
170		LDRSH		ldrsh_post_W,
171		STR		str_post_H,
172		LDR		ldr_post_H,
173		STR		str_post_W,
174		LDR		ldr_post_W,
175		LDRSW		ldrsw_post,
176		STR		str_post_S,
177		LDR		ldr_post_S,
178		STR		str_post_X,
179		LDR		ldr_post_X,
180		STR		str_post_D,
181		LDR		ldr_post_D,
182		**Load/store register (unprivileged)**		
183		STTRB		sttrb,
184		LDTRB		ldtrb,
185		LDTRSB		ldtrsb_X,

1	in_use	Opcode		NAME
186		LDTRSB		ldtrsb_W,
187		STTRH		sttrh,
188		LDTRH		ldtrh,
189		LDTRSH		ldtrsh_X,
190		LDTRSH		ldtrsh_W,
191		STTR		sttr_W,
192		LDTR		ldtr_W,
193		LDTRSW		ldtrsw,
194		STTR		sttr_X,
195		LDTR		ldtr_X,
196		**Load/store register (immediate pre-indexed**		
197		STRB		strb_pre,
198		LDRB		ldrb_pre,
199		LDRSB		ldrsb_pre_X,
200		LDRSB		ldrsb_pre_W,
201		STR		str_pre_B,
202		LDR		ldr_pre_B,
203		STR		str_pre_Q,
204		LDR		ldr_pre_Q,
205		STRH		strh_pre,
206		LDRH		ldrh_pre,
207		LDRSH		ldrsh_pre_X,
208		LDRSH		ldrsh_pre_W,
209		STR		str_pre_H,
210		LDR		ldr_pre_H,
211		STR		str_pre_W,
212		LDR		ldr_pre_W,
213		LDRSW		ldrsw_pre,
214		STR		str_pre_S,
215		LDR		ldr_pre_S,
216		STR		str_pre_X,
217		LDR		ldr_pre_X,
218		STR		str_pre_D,
219		LDR		ldr_pre_D,
220		**Load/store register (register offset)**		
221		STRB		strb_off,
222		LDRB		ldrb_off,
223		LDRSB		ldrsb_off_X,

1	in_use	Opcode		NAME
224		LDRSB		ldrsb_off_W,
225		STR		str_off_B,
226		LDR		ldr_off_B,
227		STR		str_off_Q,
228		LDR		ldr_off_Q,
229		STRH		strh_off,
230		LDRH		ldrh_off,
231		LDRSH		ldrsh_off_X,
232		LDRSH		ldrsh_off_W,
233		STR		str_off_H,
234		LDR		ldr_off_H,
235		STR		str_off_W,
236		LDR		ldr_off_W,
237		LDRSW		ldrsw_off,
238		STR		str_off_S,
239		LDR		ldr_off_S,
240		STR		str_off_D,
241		LDR		ldr_off_D,
243		STR		str_off_D,
244		LDR		ldr_off_D,
242		PRFM		prfm_off,
245		**Load/store register (unsigned immediate)**		
246		STRB		strb_imm,
247		LDRB		ldrb_imm,
248		LDRSB		ldrsb_imm_X,
249		LDRSB		ldrsb_imm_W,
250		STR		str_imm_B,
251		LDR		ldr_imm_B,
252		STR		str_imm_Q,
253		LDR		ldr_imm_Q,
254		STRH		strh_imm,
255		LDRH		ldrh_imm,
256		LDRSH		ldrsh_imm_X,
257		LDRSH		ldrsh_imm_W,
258		STR		str_imm_H,
259		LDR		ldr_imm_H,
260		STR		str_imm_W,
261		LDR		ldr_imm_W,

1	in_use	Opcode		NAME
262		LDRSW		ldrsw_imm,
263		STR		str_imm_S,
264		LDR		ldr_imm_S,
265		STR		str_imm_X,
266		LDR		ldr_imm_X,
268		STR		str_imm_D,
269		LDR		ldr_imm_D,
267		PRFM		prfm_imm,
270	**Data processing – Immediate**			
271		**PC-rel. addressing**		
272		ADR		adr,
273		ADRP		adrp,
274		**Add/subtract (immediate)**		
275		ADD		add_imm_W,
276		ADDS		adds_imm_W,
277		SUB		sub_imm_W,
278		SUBS		subs_imm_W,
279		ADD		add_imm_X,
280		ADDS		adds_imm_X,
281		SUB		sub_imm_X,
282		SUBS		subs_imm_X,
283		**Logical (immediate)**		
284		AND		and_imm_W,
285		ORR		orr_imm_W,
286		EOR		eor_imm_W,
287		ANDS		ands_imm_W,
288		AND		and_imm_X,
289		ORR		orr_imm_X,
290		EOR		eor_imm_X,
291		ANDS		ands_imm_X,
292		**Move wide (immediate)**		
293		MOVN		movn_W,
294		MOVZ		movz_W,
295		MOVK		movk_W,
296		MOVN		movn_X,
297		MOVZ		movz_X,
298		MOVK		movk_X,
299		**Bitfield**		

1	in_use	Opcode	[redacted]	NAME
300		SBFM		sbfm_W,
301		BFM		bfm_W,
302		UBFM		ubfm_W,
303		SBFM		sbfm_X,
304		BFM		bfm_X,
305		UBFM		ubfm_X,
306		**Extract**		
307		EXTR		extr_W,
308		EXTR		extr_X,
309		**Data Processing – register**		
310		**Logical (shifted register)**		
311		AND		and_W,
312		BIC		bic_W,
313		ORR		orr_W,
314		ORN		orn_W,
315		EOR		eor_W,
316		EON		eon_W,
317		ANDS		ands_W,
318		BICS		bics_W,
319		AND		and_X,
320		BIC		bic_X,
321		ORR		orr_X,
322		ORN		orn_X,
323		EOR		eor_X,
324		EON		eon_X,
325		ANDS		ands_X,
326		BICS		bics_X,
327		**Add/subtract (shifted register)**		
328		ADD		add_W,
329		ADDS		adds_W,
330		SUB		sub_W,
331		SUBS		subs_W,
332		ADD		add_X,
333		ADDS		adds_X,
334		SUB		sub_X,
335		SUBS		subs_X,
336		**Add/subtract (extended register)**		
337		ADD		add_ext_W,

1	in_use	Opcode		NAME
338		ADDS		adds_ext_W,
339		SUB		sub_ext_W,
340		SUBS		subs_ext_W,
341		ADD		add_ext_X,
342		ADDS		adds_ext_X,
343		SUB		sub_ext_X,
344		SUBS		subs_ext_X,
345		**Add/subtract (with carry)**		
346		ADC		adc_W,
347		ADCS		adcs_W,
348		SBC		sbc_W,
349		SBCS		sbcs_W,
350		ADC		adc_X,
351		ADCS		adcs_X,
352		SBC		sbc_X,
353		SBCS		sbcs_X,
354		**Conditional compare (register)**		
355		CCMN		ccmn_W,
356		CCMN		ccmn_X,
357		CCMP		ccmp_W,
358		CCMP		ccmp_X,
359		**Conditional compare (immediate)**		
360		CCMN		ccmn_imm_W,
361		CCMN		ccmn_imm_X,
362		CCMP		ccmp_imm_W,
363		CCMP		ccmp_imm_X,
364		**Conditional select**		
365		CSEL		csel_W,
366		CSINC		csinc_W,
367		CSINV		csinv_W,
368		CSNEG		csneg_W,
369		CSEL		csel_X,
370		CSINC		csinc_X,
371		CSINV		csinv_X,
372		CSNEG		csneg_X,
373		**Data-processing (3 source)**		
374		MADD		madd_W,
375		MADD		madd_X,

1	in_use	Opcode		NAME
376		SMADDL		smaddl,
377		UMADDL		umaddl,
378		MSUB		msub_W,
379		MSUB		msub_X,
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		udiv_W,
394		UDIV		udiv_X,
395		SDIV		sdiv_W,
396		SDIV		sdiv_X,
397		LSLV		lslv_W,
398		LSLV		lslv_X,
399		LSRV		lsrv_W,
400		LSRV		lsrv_X,
401		ASRV		asrv_W,
402		ASRV		asrv_X,
403		RORV		rorv_W,
404		RORV		rorv_X,
405		**Data-processing (1 source)**		
406		RBIT		rbit_W,
407		RBIT		rbit_X,
408		CLZ		clz_W,
409		CLZ		clz_X,
410		CLS		cls_W,
411		CLS		cls_X,
412		REV		rev_W,
413		REV		rev_X,

1	in_use	Opcode	NAME
414		REV16	rev16_W,
415		REV16	rev16_X,
416		REV32	rev32,
417	//	**Data Processing – SIMD and floating p**	
418	//	**Floating-point<->fixed-point conversions**	
419	//	SCVTF	ARM64Op_scvtf_scalar_fixed_point_32_bit_to_single_precision
420	//	UCVTF	ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_single_precision
421	//	FCVTZS	ARM64Op_fcvtzs_scalar_fixed_point_Single_precision_to_32_bit
422	//	FCVTZU	ARM64Op_fcvtzu_scalar_fixed_point_Single_precision_to_32_bit
423	//	SCVTF	ARM64Op_scvtf_scalar_fixed_point_32_bit_to_double_precision
424	//	UCVTF	ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_double_precision
425	//	FCVTZS	ARM64Op_fcvtzs_scalar_fixed_point_Double_precision_to_32_bit
426	//	FCVTZU	ARM64Op_fcvtzu_scalar_fixed_point_Double_precision_to_32_bit
427	//	SCVTF	ARM64Op_scvtf_scalar_fixed_point_64_bit_to_single_precision
428	//	UCVTF	ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_single_precision
429	//	FCVTZS	ARM64Op_fcvtzs_scalar_fixed_point_Single_precision_to_64_bit
430	//	FCVTZU	ARM64Op_fcvtzu_scalar_fixed_point_Single_precision_to_64_bit
431	//	SCVTF	ARM64Op_scvtf_scalar_fixed_point_64_bit_to_double_precision
432	//	UCVTF	ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_double_precision
433	//	FCVTZS	ARM64Op_fcvtzs_scalar_fixed_point_Double_precision_to_64_bit
434	//	FCVTZU	ARM64Op_fcvtzu_scalar_fixed_point_Double_precision_to_64_bit
435	//	**Floating-point conditional compare**	
436	//	FCCMP	ARM64Op_fccmp_Single_precision
437	//	FCCMPE	ARM64Op_fccmpe_Single_precision
438	//	FCCMP	ARM64Op_fccmp_Double_precision
439	//	FCCMPE	ARM64Op_fccmpe_Double_precision
440	//	**Floating-point data-processing (2 source)**	
441	//	FMUL	ARM64Op_fmul_scalar_Single_precision
442	//	FDIV	ARM64Op_fdiv_scalar_Single_precision
443	//	FADD	ARM64Op_fadd_scalar_Single_precision
444	//	FSUB	ARM64Op_fsub_scalar_Single_precision
445	//	FMAX	ARM64Op_fmax_scalar_Single_precision
446	//	FMIN	ARM64Op_fmin_scalar_Single_precision
447	//	FMAXNM	ARM64Op_fmaxnm_scalar_Single_precision

1	in_use	Opcode	NAME
448	//	FMINNM	ARM64Op_fminnm_scalar_Single_precision
449	//	FNMUL	ARM64Op_fnmul_Single_precision
450	//	FMUL	ARM64Op_fmul_scalar_Double_precision
451	//	FDIV	ARM64Op_fdiv_scalar_Double_precision
452	//	FADD	ARM64Op_fadd_scalar_Double_precision
453	//	FSUB	ARM64Op_fsub_scalar_Double_precision
454	//	FMAX	ARM64Op_fmax_scalar_Double_precision
455	//	FMIN	ARM64Op_fmin_scalar_Double_precision
456	//	FMAXNM	ARM64Op_fmaxnm_scalar_Double_precision
457	//	FMINNM	ARM64Op_fminnm_scalar_Double_precision
458	//	FNMUL	ARM64Op_fnmul_Double_precision
459	//	**Floating-point conditional select**	
460	//	FCSEL	ARM64Op_fcsel_Single_precision
461	//	FCSEL	ARM64Op_fcsel_Double_precision
462	//	**Floating-point immediate**	
463	//	FMOV	ARM64Op_fmov_scalar_immediate_Single_precision
464	//	FMOV	ARM64Op_fmov_scalar_immediate_Double_precision
465	//	**Floating-point compare**	
466	//	FCMP	ARM64Op_fcmp_Single_precision
467	//	FCMP	ARM64Op_fcmp_Single_precision_zero
468	//	FCMPE	ARM64Op_fcmpe_Single_precision
469	//	FCMPE	ARM64Op_fcmpe_Single_precision_zero
470	//	FCMP	ARM64Op_fcmp_Double_precision
471	//	FCMP	ARM64Op_fcmp_Double_precision_zero
472	//	FCMPE	ARM64Op_fcmpe_Double_precision
473	//	FCMPE	ARM64Op_fcmpe_Double_precision_zero
474	//	**Floating-point data-processing (1 source)**	
475	//	FMOV	ARM64Op_fmov_register_Single_precision
476	//	FABS	ARM64Op_fabs_scalar_Single_precision
477	//	FNEG	ARM64Op_fneg_scalar_Single_precision
478	//	FSQRT	ARM64Op_fsqrt_scalar_Single_precision
479	//	FCVT	ARM64Op_fcvt_Single_precision_to_double_precision
480	//	FCVT	ARM64Op_fcvt_Single_precision_to_half_precision
481	//	FRINTN	ARM64Op_frintn_scalar_Single_precision

1	in_use	Opcode		NAME
482	//	FRINTP		ARM64Op_frintp_scalar_Single_precision
483	//	FRINTM		ARM64Op_frintm_scalar_Single_precision
484	//	FRINTZ		ARM64Op_frintz_scalar_Single_precision
485	//	FRINTA		ARM64Op_frinta_scalar_Single_precision
486	//	FRINTX		ARM64Op_frintx_scalar_Single_precision
487	//	FRINTI		ARM64Op_frinti_scalar_Single_precision
488	//	FMOV		ARM64Op_fmov_register_Double_precision
489	//	FABS		ARM64Op_fabs_scalar_Double_precision
490	//	FNEG		ARM64Op_fneg_scalar_Double_precision
491	//	FSQRT		ARM64Op_fsqrt_scalar_Double_precision
492	//	FCVT		ARM64Op_fcvt_Double_precision_to_single_precision
493	//	FCVT		ARM64Op_fcvt_Double_precision_to_half_precision
494	//	FRINTN		ARM64Op_frintn_scalar_Double_precision
495	//	FRINTP		ARM64Op_frintp_scalar_Double_precision
496	//	FRINTM		ARM64Op_frintm_scalar_Double_precision
497	//	FRINTZ		ARM64Op_frintz_scalar_Double_precision
498	//	FRINTA		ARM64Op_frinta_scalar_Double_precision
499	//	FRINTX		ARM64Op_frintx_scalar_Double_precision
500	//	FRINTI		ARM64Op_frinti_scalar_Double_precision
501	//	FCVT		ARM64Op_fcvt_Half_precision_to_single_precision
502	//	FCVT		ARM64Op_fcvt_Half_precision_to_double_precision
503	//	**Floating-point<->integer conversions**		
504	//	FCVTNS		ARM64Op_fcvtns_scalar_Single_precision_to_32_bit
505	//	FCVTNU		ARM64Op_fcvtnu_scalar_Single_precision_to_32_bit
506	//	SCVTF		ARM64Op_scvtf_scalar_integer_32_bit_to_single_precision
507	//	UCVTF		ARM64Op_ucvtf_scalar_integer_32_bit_to_single_precision
508	//	FCVTAS		ARM64Op_fcvtas_scalar_Single_precision_to_32_bit
509	//	FCVTAU		ARM64Op_fcvtau_scalar_Single_precision_to_32_bit
510	//	FMOV		ARM64Op_fmov_general_Single_precision_to_32_bit
511	//	FMOV		ARM64Op_fmov_general_32_bit_to_single_precision
512	//	FCVTPS		ARM64Op_fcvtps_scalar_Single_precision_to_32_bit
513	//	FCVTPU		ARM64Op_fcvtpu_scalar_Single_precision_to_32_bit
514	//	FCVTMS		ARM64Op_fcvtms_scalar_Single_precision_to_32_bit
515	//	FCVTMU		ARM64Op_fcvtmu_scalar_Single_precision_to_32_bit

1	in_use	Opcode		NAME
516	//	FCVTZS		ARM64Op_fcvtzs_scalar_integer_Single_precision_to_32_bit
517	//	FCVTZU		ARM64Op_fcvtzu_scalar_integer_Single_precision_to_32_bit
518	//	FCVTNS		ARM64Op_fcvtns_scalar_Double_precision_to_32_bit
519	//	FCVTNU		ARM64Op_fcvtnu_scalar_Double_precision_to_32_bit
520	//	SCVTF		ARM64Op_scvtf_scalar_integer_32_bit_to_double_precision
521	//	UCVTF		ARM64Op_ucvtf_scalar_integer_32_bit_to_double_precision
522	//	FCVTAS		ARM64Op_fcvtas_scalar_Double_precision_to_32_bit
523	//	FCVTAU		ARM64Op_fcvtau_scalar_Double_precision_to_32_bit
524	//	FCVTPS		ARM64Op_fcvtps_scalar_Double_precision_to_32_bit
525	//	FCVTPU		ARM64Op_fcvtpu_scalar_Double_precision_to_32_bit
526	//	FCVTMS		ARM64Op_fcvtms_scalar_Double_precision_to_32_bit
527	//	FCVTMU		ARM64Op_fcvtmu_scalar_Double_precision_to_32_bit
528	//	FCVTZS		ARM64Op_fcvtzs_scalar_integer_Double_precision_to_32_bit
529	//	FCVTZU		ARM64Op_fcvtzu_scalar_integer_Double_precision_to_32_bit
530	//	FCVTNS		ARM64Op_fcvtns_scalar_Single_precision_to_64_bit
531	//	FCVTNU		ARM64Op_fcvtnu_scalar_Single_precision_to_64_bit
532	//	SCVTF		ARM64Op_scvtf_scalar_integer_64_bit_to_single_precision
533	//	UCVTF		ARM64Op_ucvtf_scalar_integer_64_bit_to_single_precision
534	//	FCVTAS		ARM64Op_fcvtas_scalar_Single_precision_to_64_bit
535	//	FCVTAU		ARM64Op_fcvtau_scalar_Single_precision_to_64_bit
536	//	FCVTPS		ARM64Op_fcvtps_scalar_Single_precision_to_64_bit
537	//	FCVTPU		ARM64Op_fcvtpu_scalar_Single_precision_to_64_bit
538	//	FCVTMS		ARM64Op_fcvtms_scalar_Single_precision_to_64_bit
539	//	FCVTMU		ARM64Op_fcvtmu_scalar_Single_precision_to_64_bit
540	//	FCVTZS		ARM64Op_fcvtzs_scalar_integer_Single_precision_to_64_bit
541	//	FCVTZU		ARM64Op_fcvtzu_scalar_integer_Single_precision_to_64_bit
542	//	FCVTNS		ARM64Op_fcvtns_scalar_Double_precision_to_64_bit
543	//	FCVTNU		ARM64Op_fcvtnu_scalar_Double_precision_to_64_bit
544	//	SCVTF		ARM64Op_scvtf_scalar_integer_64_bit_to_double_precision
545	//	UCVTF		ARM64Op_ucvtf_scalar_integer_64_bit_to_double_precision
546	//	FCVTAS		ARM64Op_fcvtas_scalar_Double_precision_to_64_bit
547	//	FCVTAU		ARM64Op_fcvtau_scalar_Double_precision_to_64_bit
548	//	FMOV		ARM64Op_fmov_general_Double_precision_to_64_bit
549	//	FMOV		ARM64Op_fmov_general_64_bit_to_double_precision

1	in_use	Opcode	NAME
550	//	FCVTPS	ARM64Op_fcvtps_scalar_Double_precision_to_64_bit
551	//	FCVTPU	ARM64Op_fcvtpu_scalar_Double_precision_to_64_bit
552	//	FCVTMS	ARM64Op_fcvtms_scalar_Double_precision_to_64_bit
553	//	FCVTMU	ARM64Op_fcvtmu_scalar_Double_precision_to_64_bit
554	//	FCVTZS	ARM64Op_fcvtzs_scalar_integer_Double_precision_to_64_bit
555	//	FCVTZU	ARM64Op_fcvtzu_scalar_integer_Double_precision_to_64_bit
556	//	FMOV	ARM64Op_fmov_general_Top_half_of_128_bit_to_64_bit
557	//	FMOV	ARM64Op_fmov_general_64_bit_to_top_half_of_128_bit
558	//	**Floating-point data-processing (3 source)**	
559	//	FMADD	ARM64Op_fmadd_Single_precision
560	//	FMSUB	ARM64Op_fmsub_Single_precision
561	//	FNMADD	ARM64Op_fnmadd_Single_precision
562	//	FNMSUB	ARM64Op_fnmsub_Single_precision
563	//	FMADD	ARM64Op_fmadd_Double_precision
564	//	FMSUB	ARM64Op_fmsub_Double_precision
565	//	FNMADD	ARM64Op_fnmadd_Double_precision
566	//	FNMSUB	ARM64Op_fnmsub_Double_precision
567	//	**AdvSIMD scalar three same**	
568	//	SQADD	ARM64Op_sqadd_Scalar
569	//	SQSUB	ARM64Op_sqsub_Scalar
570	//	CMGT	ARM64Op_cmgt_register_Scalar
571	//	CMGE	ARM64Op_cmge_register_Scalar
572	//	SSHL	ARM64Op_sshl_Scalar
573	//	SQSHL	ARM64Op_sqshl_register_Scalar
574	//	SRSHL	ARM64Op_srshl_Scalar
575	//	SQRSHL	ARM64Op_sqrshl_Scalar
576	//	ADD	ARM64Op_add_vector_Scalar
577	//	CMTST	ARM64Op_cmtst_Scalar
578	//	SQDMULH	ARM64Op_sqdmulh_vector_Scalar
579	//	FMULX	ARM64Op_fmulx_Scalar
580	//	FCMEQ	ARM64Op_fcmeq_register_Scalar
581	//	FRECPS	ARM64Op_frecps_Scalar
582	//	FRSQRTS	ARM64Op_frsqrts_Scalar
583	//	UQADD	ARM64Op_uqadd_Scalar

1	in_use	Opcode	NAME
584	//	UQSUB	ARM64Op_uqsub_Scalar
585	//	CMHI	ARM64Op_cmhi_register_Scalar
586	//	CMHS	ARM64Op_cmhs_register_Scalar
587	//	USHL	ARM64Op_ushl_Scalar
588	//	UQSHL	ARM64Op_uqshl_register_Scalar
589	//	URSHL	ARM64Op_urshl_Scalar
590	//	UQRSHL	ARM64Op_uqrshl_Scalar
591	//	SUB	ARM64Op_sub_vector_Scalar
592	//	CMEQ	ARM64Op_cmeq_register_Scalar
593	//	SQRDMULH	ARM64Op_sqrdmulh_vector_Scalar
594	//	FCMGE	ARM64Op_fcmge_register_Scalar
595	//	FACGE	ARM64Op_facge_Scalar
596	//	FABD	ARM64Op_fabd_Scalar
597	//	FCMGT	ARM64Op_fcmgt_register_Scalar
598	//	FACGT	ARM64Op_facgt_Scalar
599	//	**AdvSIMD scalar three different**	
600	//	SQDMLAL	ARM64Op_sqdmlal_vector_Scalar
601	//	SQDMLAL2	ARM64Op_sqdmlal2_vector_Scalar
602	//	SQDMLSL	ARM64Op_sqdmlsl_vector_Scalar
603	//	SQDMLSL2	ARM64Op_sqdmlsl2_vector_Scalar
604	//	SQDMULL	ARM64Op_sqdmull_vector_Scalar
605	//	SQDMULL2	ARM64Op_sqdmull2_vector_Scalar
606	//	**AdvSIMD scalar two-reg misc**	
607	//	SUQADD	ARM64Op_suqadd_Scalar
608	//	SQABS	ARM64Op_sqabs_Scalar
609	//	CMGT	ARM64Op_cmgt_zero_Scalar
610	//	CMEQ	ARM64Op_cmeq_zero_Scalar
611	//	CMLT	ARM64Op_cmlt_zero_Scalar
612	//	ABS	ARM64Op_abs_Scalar
613	//	SQXTN	ARM64Op_sqxtn_Scalar
614	//	SQXTN2	ARM64Op_sqxtn2_Scalar
615	//	FCVTNS	ARM64Op_fcvtns_vector_Scalar
616	//	FCVTMS	ARM64Op_fcvtms_vector_Scalar
617	//	FCVTAS	ARM64Op_fcvtas_vector_Scalar

1	in_use	Opcode	NAME
618	//	SCVTF	ARM64Op_scvtf_vector_integer_Scalar
619	//	FCMGT	ARM64Op_fcmgt_zero_Scalar
620	//	FCMEQ	ARM64Op_fcmeq_zero_Scalar
621	//	FCMLT	ARM64Op_fcmlt_zero_Scalar
622	//	FCVTPS	ARM64Op_fcvtps_vector_Scalar
623	//	FCVTZS	ARM64Op_fcvtzs_vector_integer_Scalar
624	//	FRECPE	ARM64Op_frecpe_Scalar
625	//	FRECPX	ARM64Op_frecpx
626	//	USQADD	ARM64Op_usqadd_Scalar
627	//	SQNEG	ARM64Op_sqneg_Scalar
628	//	CMGE	ARM64Op_cmge_zero_Scalar
629	//	CMLE	ARM64Op_cmle_zero_Scalar
630	//	NEG	ARM64Op_neg_vector_Scalar
631	//	SQXTUN	ARM64Op_sqxtun_Scalar
632	//	SQXTUN2	ARM64Op_sqxtun2_Scalar
633	//	UQXTN	ARM64Op_uqxtn_Scalar
634	//	UQXTN2	ARM64Op_uqxtn2_Scalar
635	//	FCVTXN	ARM64Op_fcvtxn_Scalar
636	//	FCVTXN2	ARM64Op_fcvtxn2_Scalar
637	//	FCVTNU	ARM64Op_fcvtnu_vector_Scalar
638	//	FCVTMU	ARM64Op_fcvtmu_vector_Scalar
639	//	FCVTAU	ARM64Op_fcvtau_vector_Scalar
640	//	UCVTF	ARM64Op_ucvtf_vector_integer_Scalar
641	//	FCMGE	ARM64Op_fcmge_zero_Scalar
642	//	FCMLE	ARM64Op_fcmle_zero_Scalar
643	//	FCVTPU	ARM64Op_fcvtpu_vector_Scalar
644	//	FCVTZU	ARM64Op_fcvtzu_vector_integer_Scalar
645	//	FRSQRTE	ARM64Op_frsqrte_Scalar
646	//	**AdvSIMD scalar pairwise**	
647	//	ADDP	ARM64Op_addp_scalar
648	//	FMAXNMP	ARM64Op_fmaxnmp_scalar
649	//	FADDP	ARM64Op_faddp_scalar
650	//	FMAXP	ARM64Op_fmaxp_scalar
651	//	FMINNMP	ARM64Op_fminnmp_scalar

1	in_use	Opcode	NAME
652	//	FMINP	ARM64Op_fminp_scalar
653	//	**AdvSIMD scalar copy**	
654	//	DUP	ARM64Op_dup_element_Scalar
655	//	**AdvSIMD scalar x indexed element**	
656	//	SQDMLAL	ARM64Op_sqdmlal_by_element_Scalar
657	//	SQDMLAL2	ARM64Op_sqdmlal2_by_element_Scalar
658	//	SQDMLSL	ARM64Op_sqdmlsl_by_element_Scalar
659	//	SQDMLSL2	ARM64Op_sqdmlsl2_by_element_Scalar
660	//	SQDMULL	ARM64Op_sqdmull_by_element_Scalar
661	//	SQDMULL2	ARM64Op_sqdmull2_by_element_Scalar
662	//	SQDMULH	ARM64Op_sqdmulh_by_element_Scalar
663	//	SQRDMULH	ARM64Op_sqrdmulh_by_element_Scalar
664	//	FMLA	ARM64Op_fmla_by_element_Scalar
665	//	FMLS	ARM64Op_fmls_by_element_Scalar
666	//	FMUL	ARM64Op_fmul_by_element_Scalar
667	//	FMULX	ARM64Op_fmulx_by_element_Scalar
668	//	**AdvSIMD scalar shift by immediate**	
669	//	SSHR	ARM64Op_sshr_Scalar
670	//	SSRA	ARM64Op_ssra_Scalar
671	//	SRSHR	ARM64Op_srshr_Scalar
672	//	SRSRA	ARM64Op_srsra_Scalar
673	//	SHL	ARM64Op_shl_Scalar
674	//	SQSHL	ARM64Op_sqshl_immediate_Scalar
675	//	SQSHRN	ARM64Op_sqshrn_Scalar
676	//	SQSHRN2	ARM64Op_sqshrn2_Scalar
677	//	SQRSHRN	ARM64Op_sqrshrn_Scalar
678	//	SQRSHRN2	ARM64Op_sqrshrn2_Scalar
679	//	SCVTF	ARM64Op_scvtf_vector_fixed_point_Scalar
680	//	FCVTZS	ARM64Op_fcvtzs_vector_fixed_point_Scalar
681	//	USHR	ARM64Op_ushr_Scalar
682	//	USRA	ARM64Op_usra_Scalar
683	//	URSHR	ARM64Op_urshr_Scalar
684	//	URSRA	ARM64Op_ursra_Scalar
685	//	SRI	ARM64Op_sri_Scalar

1	in_use	Opcode	NAME
686	//	SLI	ARM64Op_sli_Scalar
687	//	SQSHLU	ARM64Op_sqshlu_Scalar
688	//	UQSHL	ARM64Op_uqshl_immediate_Scalar
689	//	SQSHRUN	ARM64Op_sqshrun_Scalar
690	//	SQSHRUN2	ARM64Op_sqshrun2_Scalar
691	//	SQRSHRUN	ARM64Op_sqrshrun_Scalar
692	//	SQRSHRUN2	ARM64Op_sqrshrun2_Scalar
693	//	UQSHRN	ARM64Op_uqshrn_Scalar
694	//	UQRSHRN	ARM64Op_uqrshrn_Scalar
695	//	UQRSHRN2	ARM64Op_uqrshrn2_Scalar
696	//	UCVTF	ARM64Op_ucvtf_vector_fixed_point_Scalar
697	//	FCVTZU	ARM64Op_fcvtzu_vector_fixed_point_Scalar
698	//	**Crypto three-reg SHA**	
699	//	SHA1C	ARM64Op_sha1c
700	//	SHA1P	ARM64Op_sha1p
701	//	SHA1M	ARM64Op_sha1m
702	//	SHA1SU0	ARM64Op_sha1su0
703	//	SHA256H	ARM64Op_sha256h
704	//	SHA256H2	ARM64Op_sha256h2
705	//	SHA256SU1	ARM64Op_sha256su1
706	//	**Crypto two-reg SHA**	
707	//	SHA1H	ARM64Op_sha1h
708	//	SHA1SU1	ARM64Op_sha1su1
709	//	SHA256SU0	ARM64Op_sha256su0
710	//	**Crypto AES**	
711	//	AESE	ARM64Op_aese
712	//	AESD	ARM64Op_aesd
713	//	AESMC	ARM64Op_aesmc
714	//	AESIMC	ARM64Op_aesimc
715	//	**AdvSIMD three same**	
716	//	SHADD	ARM64Op_shadd
717	//	SQADD	ARM64Op_sqadd_Vector
718	//	SRHADD	ARM64Op_srhadd
719	//	SHSUB	ARM64Op_shsub

1	in_use	Opcode		NAME
720	//	SQSUB		ARM64Op_sqsub_Vector
721	//	CMGT		ARM64Op_cmgt_register_Vector
722	//	CMGE		ARM64Op_cmge_register_Vector
723	//	SSHL Vector		ARM64Op_sshl vector
724	//	SQSHL		ARM64Op_sqshl_register_Vector
725	//	SRSHL		ARM64Op_srshl_Vector
726	//	SQRSHL		ARM64Op_sqrshl_Vector
727	//	SMAX		ARM64Op_smax
728	//	SMIN		ARM64Op_smin
729	//	SABD		ARM64Op_sabd
730	//	SABA		ARM64Op_saba
731	//	ADD		ARM64Op_add_vector_Vector
732	//	CMTST		ARM64Op_cmtst_Vector
733	//	MLA		ARM64Op_mla_vector
734	//	MUL		ARM64Op_mul_vector
735	//	SMAXP		ARM64Op_smaxp
736	//	SMINP		ARM64Op_sminp
737	//	SQDMULH		ARM64Op_sqdmulh_vector_Vector
738	//	ADDP		ARM64Op_addp_vector
739	//	FMAXNM		ARM64Op_fmaxnm_vector
740	//	FMLA		ARM64Op_fmla_vector
741	//	FADD		ARM64Op_fadd_vector
742	//	FMULX		ARM64Op_fmulx_Vector
743	//	FCMEQ		ARM64Op_fcmeq_register_Vector
744	//	FMAX		ARM64Op_fmax_vector
745	//	FRECPS		ARM64Op_frecps_Vector
746	//	AND		ARM64Op_and_vector
747	//	BIC		ARM64Op_bic_vector_register
748	//	FMINNM		ARM64Op_fminnm_vector
749	//	FMLS		ARM64Op_fmls_vector
750	//	FSUB		ARM64Op_fsub_vector
751	//	FMIN		ARM64Op_fmin_vector
752	//	FRSQRTS		ARM64Op_frsqrts_Vector
753	//	ORR		ARM64Op_orr_vector_register

1	in_use	Opcode		NAME
754	//	ORN		ARM64Op_orn_vector
755	//	UHADD		ARM64Op_uhadd
756	//	UQADD		ARM64Op_uqadd_Vector
757	//	URHADD		ARM64Op_urhadd
758	//	UHSUB		ARM64Op_uhsub
759	//			ARM64Op_ _Vector
760	//	CMHI		ARM64Op_cmhi_register_Vector
761	//	CMHS		ARM64Op_cmhs_register_Vector
762	//	USHL		ARM64Op_ushl_Vector
763	//	UQSHL		ARM64Op_uqshl_register_Vector
764	//	URSHL		ARM64Op_urshl_Vector
765	//	UQRSHL		ARM64Op_uqrshl_Vector
766	//	UMAX		ARM64Op_umax
767	//	UMIN		ARM64Op_umin
768	//	UABD		ARM64Op_uabd
769	//	UABA		ARM64Op_uaba
770	//	SUB		ARM64Op_sub_vector_Vector
771	//	CMEQ		ARM64Op_cmeq_register_Vector
772	//	MLS		ARM64Op_mls_vector
773	//	PMUL		ARM64Op_pmul
774	//	UMAXP		ARM64Op_umaxp
775	//	UMINP		ARM64Op_uminp
776	//	SQRDMULH		ARM64Op_sqrdmulh_vector_Vector
777	//	FMAXNMP		ARM64Op_fmaxnmp_vector
778	//	FADDP		ARM64Op_faddp_vector
779	//	FMUL		ARM64Op_fmul_vector
780	//	FCMGE		ARM64Op_fcmge_register_Vector
781	//	FACGE		ARM64Op_facge_Vector
782	//	FMAXP		ARM64Op_fmaxp_vector
783	//	FDIV		ARM64Op_fdiv_vector
784	//	EOR		ARM64Op_eor_vector
785	//	BSL		ARM64Op_bsl
786	//	FMINNMP		ARM64Op_fminnmp_vector
787	//	FABD		ARM64Op_fabd_Vector

1	in_use	Opcode	NAME
788	//	FCMGT	ARM64Op_fcmgt_register_Vector
789	//	FACGT	ARM64Op_facgt_Vector
790	//	FMINP	ARM64Op_fminp_vector
791	//	BIT	ARM64Op_bit
792	//	BIF	ARM64Op_bif
793	//	**AdvSIMD three different**	
794	//	SADDL	ARM64Op_saddl
795	//	SADDL2	ARM64Op_saddl2
796	//	SADDW	ARM64Op_saddw
797	//	SADDW2	ARM64Op_saddw2
798	//	SSUBL	ARM64Op_ssubl
799	//	SSUBL2	ARM64Op_ssubl2
800	//	SSUBW	ARM64Op_ssubw
801	//	SSUBW2	ARM64Op_ssubw2
802	//	ADDHN	ARM64Op_addhn
803	//	ADDHN2	ARM64Op_addhn2
804	//	SABAL	ARM64Op_sabal
805	//	SABAL2	ARM64Op_sabal2
806	//	SUBHN	ARM64Op_subhn
807	//	SUBHN2	ARM64Op_subhn2
808	//	SABDL	ARM64Op_sabdl
809	//	SABDL2	ARM64Op_sabdl2
810	//	SMLAL	ARM64Op_smlal_vector
811	//	SMLAL2	ARM64Op_smlal2_vector
812	//	SQDMLAL	ARM64Op_sqdmlal_vector_Vector
813	//	SQDMLAL2	ARM64Op_sqdmlal2_vector_Vector
814	//	SMLSL	ARM64Op_smlsl_vector
815	//	SMLSL2	ARM64Op_smlsl2_vector
816	//	SQDMLSL	ARM64Op_sqdmlsl_vector_Vector
817	//	SQDMLSL2	ARM64Op_sqdmlsl2_vector_Vector
818	//	SMULL	ARM64Op_smull_vector
819	//	SMULL2	ARM64Op_smull2_vector
820	//	SQDMULL	ARM64Op_sqdmull_vector_Vector
821	//	SQDMULL2	ARM64Op_sqdmull2_vector_Vector

1	in_use	Opcode		NAME
822	//	PMULL		ARM64Op_pmull
823	//	PMULL2		ARM64Op_pmull2
824	//	UADDL		ARM64Op_uaddl
825	//	UADDL2		ARM64Op_uaddl2
826	//	UADDW		ARM64Op_uaddw
827	//	UADDW2		ARM64Op_uaddw2
828	//	USUBL		ARM64Op_usubl
829	//	USUBL2		ARM64Op_usubl2
830	//	USUBW		ARM64Op_usubw
831	//	USUBW2		ARM64Op_usubw2
832	//	RADDHN		ARM64Op_raddhn
833	//	RADDHN2		ARM64Op_raddhn2
834	//	UABAL		ARM64Op_uabal
835	//	UABAL2		ARM64Op_uabal2
836	//	RSUBHN		ARM64Op_rsubhn
837	//	RSUBHN2		ARM64Op_rsubhn2
838	//	UABDL		ARM64Op_uabdl
839	//	UABDL2		ARM64Op_uabdl2
840	//	UMLAL		ARM64Op_umlal_vector
841	//	UMLAL2		ARM64Op_umlal2_vector
842	//	UMLSL		ARM64Op_umlsl_vector
843	//	UMLSL2		ARM64Op_umlsl2_vector
844	//	UMULL		ARM64Op_umull_vector
845	//	UMULL2		ARM64Op_umull2_vector
846	//	**AdvSIMD two-reg misc**		
847	//	REV64		ARM64Op_rev64
848	//	REV16		ARM64Op_rev16_vector
849	//	SADDLP		ARM64Op_saddlp
850	//	SUQADD		ARM64Op_suqadd_Vector
851	//	CLS		ARM64Op_cls_vector
852	//	CNT		ARM64Op_cnt
853	//	SADALP		ARM64Op_sadalp
854	//	SQABS		ARM64Op_sqabs_Vector
855	//	CMGT		ARM64Op_cmgt_zero_Vector

1	in_use	Opcode	[redacted]	NAME
856	//	CMEQ		ARM64Op_cmeq_zero_Vector
857	//	CMLT		ARM64Op_cmlt_zero_Vector
858	//	ABS		ARM64Op_abs_Vector
859	//	XTN		ARM64Op_xtn
860	//	XTN2		ARM64Op_xtn2
861	//	SQXTN		ARM64Op_sqxtn_Vector
862	//	SQXTN2		ARM64Op_sqxtn2_Vector
863	//	FCVTN		ARM64Op_fcvtn
864	//	FCVTN2		ARM64Op_fcvtn2
865	//	FCVTL		ARM64Op_fcvtl
866	//	FCVTL2		ARM64Op_fcvtl2
867	//	FRINTN		ARM64Op_frintn_vector
868	//	FRINTM		ARM64Op_frintm_vector
869	//	FCVTNS		ARM64Op_fcvtns_vector_Vector
870	//	FCVTMS		ARM64Op_fcvtms_vector_Vector
871	//	FCVTAS		ARM64Op_fcvtas_vector_Vector
872	//	SCVTF		ARM64Op_scvtf_vector_integer_Vector
873	//	FCMGT		ARM64Op_fcmgt_zero_Vector
874	//	FCMEQ		ARM64Op_fcmeq_zero_Vector
875	//	FCMLT		ARM64Op_fcmlt_zero_Vector
876	//	FABS		ARM64Op_fabs_vector
877	//	FRINTP		ARM64Op_frintp_vector
878	//	FRINTZ		ARM64Op_frintz_vector
879	//	FCVTPS		ARM64Op_fcvtps_vector_Vector
880	//	FCVTZS		ARM64Op_fcvtzs_vector_integer_Vector
881	//	URECPE		ARM64Op_urecpe
882	//	FRECPE		ARM64Op_frecpe_Vector
883	//	REV32		ARM64Op_rev32_vector
884	//	UADDLP		ARM64Op_uaddlp
885	//	USQADD		ARM64Op_usqadd_Vector
886	//	CLZ		ARM64Op_clz_vector
887	//	UADALP		ARM64Op_uadalp
888	//	SQNEG		ARM64Op_sqneg_Vector
889	//	CMGE		ARM64Op_cmge_zero_Vector

1	in_use	Opcode		NAME
890	//	CMLE		ARM64Op_cmle_zero_Vector
891	//	NEG		ARM64Op_neg_vector_Vector
892	//	SQXTUN		ARM64Op_sqxtun_Vector
893	//	SQXTUN2		ARM64Op_sqxtun2_Vector
894	//	SHLL		ARM64Op_shll
895	//	SHLL2		ARM64Op_shll2
896	//	UQXTN		ARM64Op_uqxtn_Vector
897	//	UQXTN2		ARM64Op_uqxtn2_Vector
898	//	FCVTXN		ARM64Op_fcvtxn_Vector
899	//	FCVTXN2		ARM64Op_fcvtxn2_Vector
900	//	FRINTA		ARM64Op_frinta_vector
901	//	FRINTX		ARM64Op_frintx_vector
902	//	FCVTNU		ARM64Op_fcvtnu_vector_Vector
903	//	FCVTMU		ARM64Op_fcvtmu_vector_Vector
904	//	FCVTAU		ARM64Op_fcvtau_vector_Vector
905	//	UCVTF		ARM64Op_ucvtf_vector_integer_Vector
906	//	NOT		ARM64Op_not
907	//	RBIT		ARM64Op_rbit_vector
908	//	FCMGE		ARM64Op_fcmge_zero_Vector
909	//	FCMLE		ARM64Op_fcmle_zero_Vector
910	//	FNEG		ARM64Op_fneg_vector
911	//	FRINTI		ARM64Op_frinti_vector
912	//	FCVTPU		ARM64Op_fcvtpu_vector_Vector
913	//	FCVTZU		ARM64Op_fcvtzu_vector_integer_Vector
914	//	URSQRTE		ARM64Op_ursqrte
915	//	FRSQRTE		ARM64Op_frsqrte_Vector
916	//	FSQRT		ARM64Op_fsqrt_vector
917	//	**AdvSIMD across lanes**		
918	//	SADDLV		ARM64Op_saddlv
919	//	SMAXV		ARM64Op_smaxv
920	//	SMINV		ARM64Op_sminv
921	//	ADDV		ARM64Op_addv
922	//	UADDLV		ARM64Op_uaddlv
923	//	UMAXV		ARM64Op_umaxv

1	in_use	Opcode	NAME
924	//	UMINV	ARM64Op_uminv
925	//	FMAXNMV	ARM64Op_fmaxnmv
926	//	FMAXV	ARM64Op_fmaxv
927	//	FMINNMV	ARM64Op_fminnmv
928	//	FMINV	ARM64Op_fminv
929	//	**AdvSIMD copy**	
930	//	DUP	ARM64Op_dup_element_Vector
931	//	DUP	ARM64Op_dup_general
932	//	SMOV	ARM64Op_smov_32_bit
933	//	UMOV	ARM64Op_umov_32_bit
934	//	INS	ARM64Op_ins_general
935	//	SMOV	ARM64Op_smov_64_bit
936	//	UMOV	ARM64Op_umov_64_bit
937	//	INS	ARM64Op_ins_element
938	//	**AdvSIMD vector x indexed element**	
939	//	SMLAL	ARM64Op_smlal_by_element
940	//	SMLAL2	ARM64Op_smlal2_by_element
941	//	SQDMLAL	ARM64Op_sqdmlal_by_element_Vector
942	//	SQDMLAL2	ARM64Op_sqdmlal2_by_element_Vector
943	//	SMLSL	ARM64Op_smlsl_by_element
944	//	SMLSL2	ARM64Op_smlsl2_by_element
945	//	SQDMLSL	ARM64Op_sqdmlsl_by_element_Vector
946	//	SQDMLSL2	ARM64Op_sqdmlsl2_by_element_Vector
947	//	MUL	ARM64Op_mul_by_element
948	//	SMULL	ARM64Op_smull_by_element
949	//	SMULL2	ARM64Op_smull2_by_element
950	//	SQDMULL	ARM64Op_sqdmull_by_element_Vector
951	//	SQDMULL2	ARM64Op_sqdmull2_by_element_Vector
952	//	SQDMULH	ARM64Op_sqdmulh_by_element_Vector
953	//	SQRDMULH	ARM64Op_sqrdmulh_by_element_Vector
954	//	FMLA	ARM64Op_fmla_by_element_Vector
955	//	FMLS	ARM64Op_fmls_by_element_Vector
956	//	FMUL	ARM64Op_fmul_by_element_Vector
957	//	MLA	ARM64Op_mla_by_element

1	in_use	Opcode	NAME
958	//	UMLAL	ARM64Op_umlal_by_element
959	//	UMLAL2	ARM64Op_umlal2_by_element
960	//	MLS	ARM64Op_mls_by_element
961	//	UMLSL	ARM64Op_umlsl_by_element
962	//	UMLSL2	ARM64Op_umlsl2_by_element
963	//	UMULL	ARM64Op_umull_by_element
964	//	UMULL2	ARM64Op_umull2_by_element
965	//	FMULX	ARM64Op_fmulx_by_element_Vector
966	//	**AdvSIMD modified immediate**	
967	//	MOVI	ARM64Op_movi_32_bit_shifted_immediate
968	//	ORR	ARM64Op_orr_vector_immediate_32_bit
969	//	MOVI	ARM64Op_movi_16_bit_shifted_immediate
970	//	ORR	ARM64Op_orr_vector_immediate_16_bit
971	//	MOVI	ARM64Op_movi_32_bit_shifting_ones
972	//	MOVI	ARM64Op_movi_8_bit
973	//	FMOV	ARM64Op_fmov_vector_immediate_Single_precision
974	//	MVNI	ARM64Op_mvni_32_bit_shifted_immediate
975	//	BIC	ARM64Op_bic_vector_immediate_32_bit
976	//	MVNI	ARM64Op_mvni_16_bit_shifted_immediate
977	//	BIC	ARM64Op_bic_vector_immediate_16_bit
978	//	MVNI	ARM64Op_mvni_32_bit_shifting_ones
979	//	MOVI	ARM64Op_movi_64_bit_scalar
980	//	MOVI	ARM64Op_movi_64_bit_vector
981	//	FMOV	ARM64Op_fmov_vector_immediate_Double_precision
982	//	**AdvSIMD shift by immediate**	
983	//	SSHR	ARM64Op_sshr_Vector
984	//	SSRA	ARM64Op_ssra_Vector
985	//	SRSHR	ARM64Op_srshr_Vector
986	//	SRSRA	ARM64Op_srsra_Vector
987	//	SHL	ARM64Op_shl_Vector
988	//	SQSHL	ARM64Op_sqshl_immediate_Vector
989	//	SHRN	ARM64Op_shrn
990	//	SHRN2	ARM64Op_shrn2
991	//	RSHRN	ARM64Op_rshrn

1	in_use	Opcode		NAME
992	//	RSHRN2		ARM64Op_rshrn2
993	//	SQSHRN		ARM64Op_sqshrn_Vector
994	//	SQSHRN2		ARM64Op_sqshrn2_Vector
995	//	SQRSHRN		ARM64Op_sqrshrn_Vector
996	//	SQRSHRN2		ARM64Op_sqrshrn2_Vector
997	//	SSHLL		ARM64Op_sshll
998	//	SSHLL2		ARM64Op_sshll2
999	//	SCVTF		ARM64Op_scvtf_vector_fixed_point_Vector
1000	//	FCVTZS		ARM64Op_fcvtzs_vector_fixed_point_Vector
1001	//	USHR		ARM64Op_ushr_Vector
1002	//	USRA		ARM64Op_usra_Vector
1003	//	URSHR		ARM64Op_urshr_Vector
1004	//	URSRA		ARM64Op_ursra_Vector
1005	//	SRI		ARM64Op_sri_Vector
1006	//	SLI		ARM64Op_sli_Vector
1007	//	SQSHLU		ARM64Op_sqshlu_Vector
1008	//	UQSHL		ARM64Op_uqshl_immediate_Vector
1009	//	SQSHRUN		ARM64Op_sqshrun_Vector
1010	//	SQSHRUN2		ARM64Op_sqshrun2_Vector
1011	//	SQRSHRUN		ARM64Op_sqrshrun_Vector
1012	//	SQRSHRUN2		ARM64Op_sqrshrun2_Vector
1013	//	UQSHRN		ARM64Op_uqshrn_Vector
1014	//	UQRSHRN		ARM64Op_uqrshrn_Vector
1015	//	UQRSHRN2		ARM64Op_uqrshrn2_Vector
1016	//	USHLL		ARM64Op_ushll
1017	//	USHLL2		ARM64Op_ushll2
1018	//	UCVTF		ARM64Op_ucvtf_vector_fixed_point_Vector
1019	//	FCVTZU		ARM64Op_fcvtzu_vector_fixed_point_Vector
1020	//	**AdvSIMD TBL/TBX**		
1021	//	TBL		ARM64Op_tbl_Single_register_table
1022	//	TBX		ARM64Op_tbx_Single_register_table
1023	//	TBL		ARM64Op_tbl_Two_register_table
1024	//	TBX		ARM64Op_tbx_Two_register_table
1025	//	TBL		ARM64Op_tbl_Three_register_table

	in_use	Opcode	NAME
1026	//	TBX	ARM64Op_tbx_Three_register_table
1027	//	TBL	ARM64Op_tbl_Four_register_table
1028	//	TBX	ARM64Op_tbx_Four_register_table
1029	//	**AdvSIMD ZIP/UZP/TRN**	
1030	//	UZP1	ARM64Op_uzp1
1031	//	TRN1	ARM64Op_trn1
1032	//	ZIP1	ARM64Op_zip1
1033	//	UZP2	ARM64Op_uzp2
1034	//	TRN2	ARM64Op_trn2
1035	//	ZIP2	ARM64Op_zip2
1036	//	**AdvSIMD EXT**	
1037	//	EXT	ARM64Op_ext
1038	//	**Loads and stores**	
1039	//	**AdvSIMD load/store multiple structures**	
1040	//	ST4	ARM64Op_st4_multiple_structures_No_offset
1041	//	ST1	ARM64Op_st1_multiple_structures_Four_registers
1042	//	ST3	ARM64Op_st3_multiple_structures_No_offset
1043	//	ST1	ARM64Op_st1_multiple_structures_Three_registers
1044	//	ST1	ARM64Op_st1_multiple_structures_One_register
1045	//	ST2	ARM64Op_st2_multiple_structures_No_offset
1046	//	ST1	ARM64Op_st1_multiple_structures_Two_registers
1047	//	LD4	ARM64Op_ld4_multiple_structures_No_offset
1048	//	LD1	ARM64Op_ld1_multiple_structures_Four_registers
1049	//	LD3	ARM64Op_ld3_multiple_structures_No_offset
1050	//	LD1	ARM64Op_ld1_multiple_structures_Three_registers
1051	//	LD1	ARM64Op_ld1_multiple_structures_One_register
1052	//	LD2	ARM64Op_ld2_multiple_structures_No_offset
1053	//	LD1	ARM64Op_ld1_multiple_structures_Two_registers
1054	//	**AdvSIMD load/store multiple structures (po**	
1055	//	ST4	ARM64Op_st4_multiple_structures_Register_offset
1056	//	ST1	ARM64Op_st1_multiple_structures_Four_registers_register_offset
1057	//	ST3	ARM64Op_st3_multiple_structures_Register_offset
1058	//	ST1	ARM64Op_st1_multiple_structures_Three_registers_register_offset
1059	//	ST1	ARM64Op_st1_multiple_structures_One_register_register_offset

1	in_use	Opcode		NAME
1060	//	ST2		ARM64Op_st2_multiple_structures_Register_offset
1061	//	ST1		ARM64Op_st1_multiple_structures_Two_registers_register_offset
1062	//	ST4		ARM64Op_st4_multiple_structures_Immediate_offset
1063	//	ST1		ARM64Op_st1_multiple_structures_Four_registers_immediate_offset
1064	//	ST3		ARM64Op_st3_multiple_structures_Immediate_offset
1065	//	ST1		ARM64Op_st1_multiple_structures_Three_registers_immediate_offset
1066	//	ST1		ARM64Op_st1_multiple_structures_One_register_immediate_offset
1067	//	ST2		ARM64Op_st2_multiple_structures_Immediate_offset
1068	//	ST1		ARM64Op_st1_multiple_structures_Two_registers_immediate_offset
1069	//	LD4		ARM64Op_ld4_multiple_structures_Register_offset
1070	//	LD1		ARM64Op_ld1_multiple_structures_Four_registers_register_offset
1071	//	LD3		ARM64Op_ld3_multiple_structures_Register_offset
1072	//	LD1		ARM64Op_ld1_multiple_structures_Three_registers_register_offset
1073	//	LD1		ARM64Op_ld1_multiple_structures_One_register_register_offset
1074	//	LD2		ARM64Op_ld2_multiple_structures_Register_offset
1075	//	LD1		ARM64Op_ld1_multiple_structures_Two_registers_register_offset
1076	//	LD4		ARM64Op_ld4_multiple_structures_Immediate_offset
1077	//	LD1		ARM64Op_ld1_multiple_structures_Four_registers_immediate_offset
1078	//	LD3		ARM64Op_ld3_multiple_structures_Immediate_offset
1079	//	LD1		ARM64Op_ld1_multiple_structures_Three_registers_immediate_offset
1080	//	LD1		ARM64Op_ld1_multiple_structures_One_register_immediate_offset
1081	//	LD2		ARM64Op_ld2_multiple_structures_Immediate_offset
1082	//	LD1		ARM64Op_ld1_multiple_structures_Two_registers_immediate_offset
1083	//	**AdvSIMD load/store single structure**		
1084	//	ST1		ARM64Op_st1_single_structure_8_bit
1085	//	ST3		ARM64Op_st3_single_structure_8_bit
1086	//	ST1		ARM64Op_st1_single_structure_16_bit
1087	//	ST3		ARM64Op_st3_single_structure_16_bit
1088	//	ST1		ARM64Op_st1_single_structure_32_bit
1089	//	ST1		ARM64Op_st1_single_structure_64_bit
1090	//	ST3		ARM64Op_st3_single_structure_32_bit
1091	//	ST3		ARM64Op_st3_single_structure_64_bit
1092	//	ST2		ARM64Op_st2_single_structure_8_bit
1093	//	ST4		ARM64Op_st4_single_structure_8_bit

1	in_use	Opcode		NAME
1094	//	ST2		ARM64Op_st2_single_structure_16_bit
1095	//	ST4		ARM64Op_st4_single_structure_16_bit
1096	//	ST2		ARM64Op_st2_single_structure_32_bit
1097	//	ST2		ARM64Op_st2_single_structure_64_bit
1098	//	ST4		ARM64Op_st4_single_structure_32_bit
1099	//	ST4		ARM64Op_st4_single_structure_64_bit
1100	//	LD1		ARM64Op_ld1_single_structure_8_bit
1101	//	LD3		ARM64Op_ld3_single_structure_8_bit
1102	//	LD1		ARM64Op_ld1_single_structure_16_bit
1103	//	LD3		ARM64Op_ld3_single_structure_16_bit
1104	//	LD1		ARM64Op_ld1_single_structure_32_bit
1105	//	LD1		ARM64Op_ld1_single_structure_64_bit
1106	//	LD3		ARM64Op_ld3_single_structure_32_bit
1107	//	LD3		ARM64Op_ld3_single_structure_64_bit
1108	//	LD1R		ARM64Op_ld1r_No_offset
1109	//	LD3R		ARM64Op_ld3r_No_offset
1110	//	LD2		ARM64Op_ld2_single_structure_8_bit
1111	//	LD4		ARM64Op_ld4_single_structure_8_bit
1112	//	LD2		ARM64Op_ld2_single_structure_16_bit
1113	//	LD4		ARM64Op_ld4_single_structure_16_bit
1114	//	LD2		ARM64Op_ld2_single_structure_32_bit
1115	//	LD2		ARM64Op_ld2_single_structure_64_bit
1116	//	LD4		ARM64Op_ld4_single_structure_32_bit
1117	//	LD4		ARM64Op_ld4_single_structure_64_bit
1118	//	LD2R		ARM64Op_ld2r_No_offset
1119	//	LD4R		ARM64Op_ld4r_No_offset
1120	//	**AdvSIMD load/store single structure (post-**		
1121	//	ST1		ARM64Op_st1_single_structure_8_bit_register_offset
1122	//	ST3		ARM64Op_st3_single_structure_8_bit_register_offset
1123	//	ST1		ARM64Op_st1_single_structure_16_bit_register_offset
1124	//	ST3		ARM64Op_st3_single_structure_16_bit_register_offset
1125	//	ST1		ARM64Op_st1_single_structure_32_bit_register_offset
1126	//	ST1		ARM64Op_st1_single_structure_64_bit_register_offset
1127	//	ST3		ARM64Op_st3_single_structure_32_bit_register_offset

1	in_use	Opcode		NAME
1128	//	ST3		ARM64Op_st3_single_structure_64_bit_register_offset
1129	//	ST1		ARM64Op_st1_single_structure_8_bit_immediate_offset
1130	//	ST3		ARM64Op_st3_single_structure_8_bit_immediate_offset
1131	//	ST1		ARM64Op_st1_single_structure_16_bit_immediate_offset
1132	//	ST3		ARM64Op_st3_single_structure_16_bit_immediate_offset
1133	//	ST1		ARM64Op_st1_single_structure_32_bit_immediate_offset
1134	//	ST1		ARM64Op_st1_single_structure_64_bit_immediate_offset
1135	//	ST3		ARM64Op_st3_single_structure_32_bit_immediate_offset
1136	//	ST3		ARM64Op_st3_single_structure_64_bit_immediate_offset
1137	//	ST2		ARM64Op_st2_single_structure_8_bit_register_offset
1138	//	ST4		ARM64Op_st4_single_structure_8_bit_register_offset
1139	//	ST2		ARM64Op_st2_single_structure_16_bit_register_offset
1140	//	ST4		ARM64Op_st4_single_structure_16_bit_register_offset
1141	//	ST2		ARM64Op_st2_single_structure_32_bit_register_offset
1142	//	ST2		ARM64Op_st2_single_structure_64_bit_register_offset
1143	//	ST4		ARM64Op_st4_single_structure_32_bit_register_offset
1144	//	ST4		ARM64Op_st4_single_structure_64_bit_register_offset
1145	//	ST2		ARM64Op_st2_single_structure_8_bit_immediate_offset
1146	//	ST4		ARM64Op_st4_single_structure_8_bit_immediate_offset
1147	//	ST2		ARM64Op_st2_single_structure_16_bit_immediate_offset
1148	//	ST4		ARM64Op_st4_single_structure_16_bit_immediate_offset
1149	//	ST2		ARM64Op_st2_single_structure_32_bit_immediate_offset
1150	//	ST2		ARM64Op_st2_single_structure_64_bit_immediate_offset
1151	//	ST4		ARM64Op_st4_single_structure_32_bit_immediate_offset
1152	//	ST4		ARM64Op_st4_single_structure_64_bit_immediate_offset
1153	//	LD1		ARM64Op_ld1_single_structure_8_bit_register_offset
1154	//	LD3		ARM64Op_ld3_single_structure_8_bit_register_offset
1155	//	LD1		ARM64Op_ld1_single_structure_16_bit_register_offset
1156	//	LD3		ARM64Op_ld3_single_structure_16_bit_register_offset
1157	//	LD1		ARM64Op_ld1_single_structure_32_bit_register_offset
1158	//	LD1		ARM64Op_ld1_single_structure_64_bit_register_offset
1159	//	LD3		ARM64Op_ld3_single_structure_32_bit_register_offset
1160	//	LD3		ARM64Op_ld3_single_structure_64_bit_register_offset
1161	//	LD1R		ARM64Op_ld1r_Register_offset

1	in_use	Opcode		NAME
1162	//	LD3R		ARM64Op_ld3r_Register_offset
1163	//	LD1		ARM64Op_ld1_single_structure_8_bit_immediate_offset
1164	//	LD3		ARM64Op_ld3_single_structure_8_bit_immediate_offset
1165	//	LD1		ARM64Op_ld1_single_structure_16_bit_immediate_offset
1166	//	LD3		ARM64Op_ld3_single_structure_16_bit_immediate_offset
1167	//	LD1		ARM64Op_ld1_single_structure_32_bit_immediate_offset
1168	//	LD1		ARM64Op_ld1_single_structure_64_bit_immediate_offset
1169	//	LD3		ARM64Op_ld3_single_structure_32_bit_immediate_offset
1170	//	LD3		ARM64Op_ld3_single_structure_64_bit_immediate_offset
1171	//	LD1R		ARM64Op_ld1r_Immediate_offset
1172	//	LD3R		ARM64Op_ld3r_Immediate_offset
1173	//	LD2		ARM64Op_ld2_single_structure_8_bit_register_offset
1174	//	LD4		ARM64Op_ld4_single_structure_8_bit_register_offset
1175	//	LD2		ARM64Op_ld2_single_structure_16_bit_register_offset
1176	//	LD4		ARM64Op_ld4_single_structure_16_bit_register_offset
1177	//	LD2		ARM64Op_ld2_single_structure_32_bit_register_offset
1178	//	LD2		ARM64Op_ld2_single_structure_64_bit_register_offset
1179	//	LD4		ARM64Op_ld4_single_structure_32_bit_register_offset
1180	//	LD4		ARM64Op_ld4_single_structure_64_bit_register_offset
1181	//	LD2R		ARM64Op_ld2r_Register_offset
1182	//	LD4R		ARM64Op_ld4r_Register_offset
1183	//	LD2		ARM64Op_ld2_single_structure_8_bit_immediate_offset
1184	//	LD4		ARM64Op_ld4_single_structure_8_bit_immediate_offset
1185	//	LD2		ARM64Op_ld2_single_structure_16_bit_immediate_offset
1186	//	LD4		ARM64Op_ld4_single_structure_16_bit_immediate_offset
1187	//	LD2		ARM64Op_ld2_single_structure_32_bit_immediate_offset
1188	//	LD2		ARM64Op_ld2_single_structure_64_bit_immediate_offset
1189	//	LD4		ARM64Op_ld4_single_structure_32_bit_immediate_offset
1190	//	LD4		ARM64Op_ld4_single_structure_64_bit_immediate_offset
1191	//	LD2R		ARM64Op_ld2r_Immediate_offset
1192	//	LD4R		ARM64Op_ld4r_Immediate_offset

1	**in_use**	**Opcode**	**//Opcode**	**BINARY OI**
2		**UNALLOCATED**	/* UNALLOCATED */	
3		BAD	bad,	/* 0x00000000BAD invali
4		**Branch,exception generation and syst**	/* Branch,exception generation and system Instruction */	
5		**Compare _ Branch (immediate)**	/* Compare _ Branch (immediate) */	
6		CBZ	cbz_W,	/* 0x34000000CBZ */
7		CBNZ	cbnz_W,	/* 0x35000000CBNZ */
8		CBZ	cbz_X,	/* 0xB4000000CBZ */
9		CBNZ	cbnz_X,	/* 0xB5000000CBNZ */
10		**Test bit & branch (immediate)**	/* Test bit & branch (immediate) */	
11		TBZ	tbz,	/* 0x36000000TBZ */
12		TBNZ	tbnz,	/* 0x37000000TBNZ */
13		**Conditional branch (immediate)**	/* Conditional branch (immediate) */	
14		B_cond	b_cond,	/* 0x54000000B_cond */
15		**Exception generation**	/* Exception generation */	
16	//	SVC	//svc,	/* 0xD4000001SVC */
17	//	HVC	//hvc,	/* 0xD4000002HVC */
18	//	SMC	//smc,	/* 0xD4000003SMC */
19		BRK	brk,	/* 0xD4200000BRK */
20	//	HLT	//hlt,	/* 0xD4400000HLT */
21	//	DCPS1	//dcps1,	/* 0xD4A00001DCPS1 *
22	//	DCPS2	//dcps2,	/* 0xD4A00002DCPS2 *
23	//	DCPS3	//dcps3,	/* 0xD4A00003DCPS3 *
24	//	**System**	/* System */	
25	//	MSR	//msr_imm,	/* 0xD500401FMSR
26	//	HINT	//hint,	/* 0xD503201FHINT */
27	//	CLREX	//clrex,	/* 0xD503305FCLREX */
28	//	DSB	//dsb,	/* 0xD503309FDSB */
29	//	DMB	//dmb,	/* 0xD50330BFDMB */
30	//	ISB	//isb,	/* 0xD50330DFISB */
31	//	SYS	//sys,	/* 0xD5080000SYS */
32	//	MSR	//msr,	/* 0xD5100000MSR */
33	//	SYSL	//sysl,	/* 0xD5280000SYSL */
34	//	MRS	//mrs,	/* 0xD5300000MRS */
35		**Unconditional branch (register)**	/* Unconditional branch (register) */	
36		BR	br,	/* 0xD61F0000BR */
37		BLR	blr,	/* 0xD63F0000BLR */

	in_use	Opcode	//Opcode	BINARY	Ol
38		RET	ret,	/* 0xD65F0000RET */	
39	//	ERET	//eret,	/* 0xD69F03E0ERET */	
40	//	DRPS	//drps,	/* 0xD6BF03E0DRPS */	
41	//	**Unconditional branch (immediate)**	/* 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	stxr_W,	/* 0x88000000STXR */	
59		STLXR	stlxr_W,	/* 0x88008000STLXR */	
60		STXP	stxp_W,	/* 0x88200000STXP */	
61		STLXP	stlxp_W,	/* 0x88208000STLXP */	
62		LDXR	ldxr_W,	/* 0x88400000LDXR */	
63		LDAXR	ldaxr_W,	/* 0x88408000LDAXR */	
64		LDXP	ldxp_W,	/* 0x88600000LDXP */	
65		LDAXP	ldaxp_W,	/* 0x88608000LDAXP */	
66		STLR	stlr_W,	/* 0x88808000STLR */	
67		LDAR	ldar_W,	/* 0x88C08000LDAR */	
68		STXR	stxr_X,	/* 0xC8000000STXR */	
69		STLXR	stlxr_X,	/* 0xC8008000STLXR */	
70		STXP	stxp_X,	/* 0xC8200000STXP */	
71		STLXP	stlxp_X,	/* 0xC8208000STLXP */	
72		LDXR	ldxr_X,	/* 0xC8400000LDXR */	
73		LDAXR	ldaxr_X,	/* 0xC8408000LDAXR */	
74		LDXP	ldxp_X,	/* 0xC8600000LDXP */	

1	in_use	Opcode	//Opcode	BINARY	Ol
75		LDAXP	ldaxp_X,	/* 0xC8608000LDAXP	*/
76		STLR	stlr_X,	/* 0xC8808000STLR	*/
77		LDAR	ldar_X,	/* 0xC8C08000LDAR	*/
78		**Load register (literal)**	/* Load register (literal) */		
79		LDR	ldr_W,	/* 0x18000000LDR	*/
80		LDR	ldr_S,	/* 0x1C000000LDR	*/
81		LDR	ldr_X,	/* 0x58000000LDR	*/
82		LDR	ldr_D,	/* 0x5C000000LDR	*/
83		LDRSW	ldrsw,	/* 0x98000000LDRSW	*/
84		LDR	ldr_Q,	/* 0x9C000000LDR	*/
85		PRFM	prfm,	/* 0xD8000000PRFM	*/
86		**Load/store no-allocate pair (offset)**	/* Load/store no-allocate pair (offset) */		
87		STNP	stnp_W,	/* 0x28000000STNP	*/
88		LDNP	ldnp_W,	/* 0x28400000LDNP	*/
89		STNP	stnp_S,	/* 0x2C000000STNP	*/
90		LDNP	ldnp_S,	/* 0x2C400000LDNP	*/
91		STNP	stnp_D,	/* 0x6C000000STNP	*/
92		LDNP	ldnp_D,	/* 0x6C400000LDNP	*/
93		STNP	stnp_X,	/* 0xA8000000STNP	*/
94		LDNP	ldnp_X,	/* 0xA8400000LDNP	*/
95		STNP	stnp_Q,	/* 0xAC000000STNP	*/
96		LDNP	ldnp_Q,	/* 0xAC400000LDNP	*/
97		**Load/store register pair (post-indexed)**	/* Load/store register pair (post-indexed) */		
98		STP	stp_post_W,	/* 0x28800000STP	*/
99		LDP	ldp_post_W,	/* 0x28C00000LDP	*/
100		STP	stp_post_S,	/* 0x2C800000STP	*/
101		LDP	ldp_post_S,	/* 0x2CC00000LDP	*/
102		LDPSW	ldpsw_post,	/* 0x68C00000LDPSW	
103		STP	stp_post_D,	/* 0x6C800000STP	*/
104		LDP	ldp_post_D,	/* 0x6CC00000LDP	*/
105		STP	stp_post_X,	/* 0xA8800000STP	*/
106		LDP	ldp_post_X,	/* 0xA8C00000LDP	*/
107		STP	stp_post_Q,	/* 0xAC800000STP	*/
108		LDP	ldp_post_Q,	/* 0xACC00000LDP	*
109		**Load/store register pair (offset)**	/* Load/store register pair (offset) */		

1	in_use	Opcode	//Opcode	BINARY	Ol
110		STP	stp_off_W,	/* 0x29000000STP	*/
111		LDP	ldp_off_W,	/* 0x29400000LDP	*/
112		STP	stp_off_S,	/* 0x2D000000STP	*/
113		LDP	ldp_off_S,	/* 0x2D400000LDP	*/
114		LDPSW	ldpsw_off,	/* 0x69400000LDPSW	*
115		STP	stp_off_D,	/* 0x6D000000STP	*/
116		LDP	ldp_off_D,	/* 0x6D400000LDP	*/
117		STP	stp_off_X,	/* 0xA9000000STP	*/
118		LDP	ldp_off_X,	/* 0xA9400000LDP	*/
119		STP	stp_off_Q,	/* 0xAD000000STP	*/
120		LDP	ldp_off_Q,	/* 0xAD400000LDP	*/
121		**Load/store register pair (pre-indexed)**	/* Load/store register pair (pre-indexed) */		
122		STP	stp_pre_W,	/* 0x29800000STP	*/
123		LDP	ldp_pre_W,	/* 0x29C00000LDP	*/
124		STP	stp_pre_S,	/* 0x2D800000STP	*/
125		LDP	ldp_pre_S,	/* 0x2DC00000LDP	*/
126		LDPSW	ldpsw_pre,	/* 0x69C00000LDPSW	
127		STP	stp_pre_D,	/* 0x6D800000STP	*/
128		LDP	ldp_pre_D,	/* 0x6DC00000LDP	*/
129		STP	stp_pre_X,	/* 0xA9800000STP	*/
130		LDP	ldp_pre_X,	/* 0xA9C00000LDP	*/
131		STP	stp_pre_Q,	/* 0xAD800000STP	*/
132		LDP	ldp_pre_Q,	/* 0xADC00000LDP	*/
133		**Load/store register (unscaled immediate)**	/* Load/store register (unscaled immediate) */		
134		STURB	sturb,	/* 0x38000000STURB	*/
135		LDURB	ldurb,	/* 0x38400000LDURB	*/
136		LDURSB	ldursb_X,	/* 0x38800000LDURSB	
137		LDURSB	ldursb_W,	/* 0x38C00000LDURSB	
138		STUR	stur_B,	/* 0x3C000000STUR	*/
139		LDUR	ldur_B,	/* 0x3C400000LDUR	*/
140		STUR	stur_Q,	/* 0x3C800000STUR	*/
141		LDUR	ldur_Q,	/* 0x3CC00000LDUR	*/
142		STURH	sturh,	/* 0x78000000STURH	*/
143		LDURH	ldurh,	/* 0x78400000LDURH	*/
144		LDURSH	ldursh_X,	/* 0x78800000LDURSH	
145		LDURSH	ldursh_W,	/* 0x78C00000LDURSH	
146		STUR	stur_H,	/* 0x7C000000STUR	*/
147		LDUR	ldur_H,	/* 0x7C400000LDUR	*/

1	in_use	Opcode	//Opcode	BINARY	Ol
148		STUR	stur_W,	/* 0xB8000000STUR	*/
149		LDUR	ldur_W,	/* 0xB8400000LDUR	*/
150		LDURSW	ldursw,	/* 0xB8800000LDURSW	*
151		STUR	stur_S,	/* 0xBC000000STUR	*/
152		LDUR	ldur_S,	/* 0xBC400000LDUR	*/
153		STUR	stur_X,	/* 0xF8000000STUR	*/
154		LDUR	ldur_X,	/* 0xF8400000LDUR	*/
155		PRFUM	prfum,	/* 0xF8800000PRFUM	*/
156		STUR	stur_D,	/* 0xFC000000STUR	*/
157		LDUR	ldur_D,	/* 0xFC400000LDUR	*/
158		**Load/store register (immediate post-indexe**	/* Load/store register (immediate post-indexed) */		
159		STRB	strb_post,	/* 0x38000400STRB	*/
160		LDRB	ldrb_post,	/* 0x38400400LDRB	*/
161		LDRSB	ldrsb_post_X,	/* 0x38800400LDRSB	
162		LDRSB	ldrsb_post_W,	/* 0x38C00400LDRSB	
163		STR	str_post_B,	/* 0x3C000400STR	*/
164		LDR	ldr_post_B,	/* 0x3C400400LDR	*/
165		STR	str_post_Q,	/* 0x3C800400STR	*/
166		LDR	ldr_post_Q,	/* 0x3CC00400LDR	*/
167		STRH	strh_post,	/* 0x78000400STRH	*/
168		LDRH	ldrh_post,	/* 0x78400400LDRH	*/
169		LDRSH	ldrsh_post_X,	/* 0x78800400LDRSH	
170		LDRSH	ldrsh_post_W,	/* 0x78C00400LDRSH	
171		STR	str_post_H,	/* 0x7C000400STR	*/
172		LDR	ldr_post_H,	/* 0x7C400400LDR	*/
173		STR	str_post_W,	/* 0xB8000400STR	*/
174		LDR	ldr_post_W,	/* 0xB8400400LDR	*/
175		LDRSW	ldrsw_post,	/* 0xB8800400LDRSW	
176		STR	str_post_S,	/* 0xBC000400STR	*/
177		LDR	ldr_post_S,	/* 0xBC400400LDR	*/
178		STR	str_post_X,	/* 0xF8000400STR	*/
179		LDR	ldr_post_X,	/* 0xF8400400LDR	*/
180		STR	str_post_D,	/* 0xFC000400STR	*/
181		LDR	ldr_post_D,	/* 0xFC400400LDR	*/
182		**Load/store register (unprivileged)**	/* Load/store register (unprivileged) */		
183		STTRB	sttrb,	/* 0x38000800STTRB	*/
184		LDTRB	ldtrb,	/* 0x38400800LDTRB	*/
185		LDTRSB	ldtrsb_X,	/* 0x38800800LDTRSB	*/

1	in_use	Opcode	//Opcode	BINARY	Ol
186		LDTRSB	ldtrsb_W,	/* 0x38C00800LDTRSB	
187		STTRH	sttrh,	/* 0x78000800STTRH */	
188		LDTRH	ldtrh,	/* 0x78400800LDTRH */	
189		LDTRSH	ldtrsh_X,	/* 0x78800800LDTRSH	*
190		LDTRSH	ldtrsh_W,	/* 0x78C00800LDTRSH	
191		STTR	sttr_W,	/* 0xB8000800STTR */	
192		LDTR	ldtr_W,	/* 0xB8400800LDTR */	
193		LDTRSW	ldtrsw,	/* 0xB8800800LDTRSW	*/
194		STTR	sttr_X,	/* 0xF8000800STTR */	
195		LDTR	ldtr_X,	/* 0xF8400800LDTR */	
196		**Load/store register (immediate pre-indexed**	/* Load/store register (immediate pre-indexed) */		
197		STRB	strb_pre,	/* 0x38000C00STRB */	
198		LDRB	ldrb_pre,	/* 0x38400C00LDRB */	
199		LDRSB	ldrsb_pre_X,	/* 0x38800C00LDRSB	
200		LDRSB	ldrsb_pre_W,	/* 0x38C00C00LDRSB	
201		STR	str_pre_B,	/* 0x3C000C00STR */	
202		LDR	ldr_pre_B,	/* 0x3C400C00LDR */	
203		STR	str_pre_Q,	/* 0x3C800C00STR */	
204		LDR	ldr_pre_Q,	/* 0x3CC00C00LDR	*/
205		STRH	strh_pre,	/* 0x78000C00STRH */	
206		LDRH	ldrh_pre,	/* 0x78400C00LDRH */	
207		LDRSH	ldrsh_pre_X,	/* 0x78800C00LDRSH	
208		LDRSH	ldrsh_pre_W,	/* 0x78C00C00LDRSH	
209		STR	str_pre_H,	/* 0x7C000C00STR */	
210		LDR	ldr_pre_H,	/* 0x7C400C00LDR */	
211		STR	str_pre_W,	/* 0xB8000C00STR */	
212		LDR	ldr_pre_W,	/* 0xB8400C00LDR */	
213		LDRSW	ldrsw_pre,	/* 0xB8800C00LDRSW	
214		STR	str_pre_S,	/* 0xBC000C00STR */	
215		LDR	ldr_pre_S,	/* 0xBC400C00LDR */	
216		STR	str_pre_X,	/* 0xF8000C00STR */	
217		LDR	ldr_pre_X,	/* 0xF8400C00LDR */	
218		STR	str_pre_D,	/* 0xFC000C00STR */	
219		LDR	ldr_pre_D,	/* 0xFC400C00LDR */	
220		**Load/store register (register offset)**	/* Load/store register (register offset) */		
221		STRB	strb_off,	/* 0x38200800STRB */	
222		LDRB	ldrb_off,	/* 0x38600800LDRB */	
223		LDRSB	ldrsb_off_X,	/* 0x38A00800LDRSB	

1	in_use	Opcode	//Opcode	BINARY	Ol
224		LDRSB	ldrsb_off_W,	/* 0x38E00800LDRSB	
225		STR	str_off_B,	/* 0x3C200800STR	*/
226		LDR	ldr_off_B,	/* 0x3C600800LDR	*/
227		STR	str_off_Q,	/* 0x3CA00800STR	*/
228		LDR	ldr_off_Q,	/* 0x3CE00800LDR	*/
229		STRH	strh_off,	/* 0x78200800STRH	*/
230		LDRH	ldrh_off,	/* 0x78600800LDRH	*/
231		LDRSH	ldrsh_off_X,	/* 0x78A00800LDRSH	
232		LDRSH	ldrsh_off_W,	/* 0x78E00800LDRSH	
233		STR	str_off_H,	/* 0x7C200800STR	*/
234		LDR	ldr_off_H,	/* 0x7C600800LDR	*/
235		STR	str_off_W,	/* 0xB8200800STR	*/
236		LDR	ldr_off_W,	/* 0xB8600800LDR	*/
237		LDRSW	ldrsw_off,	/* 0xB8A00800LDRSW	*
238		STR	str_off_S,	/* 0xBC200800STR	*/
239		LDR	ldr_off_S,	/* 0xBC600800LDR	*/
240		STR	str_off_D,	/* 0xF8200800STR	*/
241		LDR	ldr_off_D,	/* 0xF8600800LDR	*/
243		STR	str_off_D,	/* 0xFC200800STR	*/
244		LDR	ldr_off_D,	/* 0xFC600800LDR	*/
242		PRFM	prfm_off,	/* 0xF8A00800PRFM	*/
245		**Load/store register (unsigned immediate)**	/* Load/store register (unsigned immediate) */		
246		STRB	strb_imm,	/* 0x39000000STRB	*/
247		LDRB	ldrb_imm,	/* 0x39400000LDRB	*/
248		LDRSB	ldrsb_imm_X,	/* 0x39800000LDRSB	
249		LDRSB	ldrsb_imm_W,	/* 0x39C00000LDRSB	
250		STR	str_imm_B,	/* 0x3D000000STR	*/
251		LDR	ldr_imm_B,	/* 0x3D400000LDR	*/
252		STR	str_imm_Q,	/* 0x3D800000STR	*/
253		LDR	ldr_imm_Q,	/* 0x3DC00000LDR	*
254		STRH	strh_imm,	/* 0x79000000STRH	*/
255		LDRH	ldrh_imm,	/* 0x79400000LDRH	*/
256		LDRSH	ldrsh_imm_X,	/* 0x79800000LDRSH	
257		LDRSH	ldrsh_imm_W,	/* 0x79C00000LDRSH	
258		STR	str_imm_H,	/* 0x7D000000STR	*/
259		LDR	ldr_imm_H,	/* 0x7D400000LDR	*/
260		STR	str_imm_W,	/* 0xB9000000STR	*
261		LDR	ldr_imm_W,	/* 0xB9400000LDR	*

1	in_use	Opcode	//Opcode	BINARY	Ol
262		LDRSW	ldrsw_imm,	/* 0xB9800000LDRSW	
263		STR	str_imm_S,	/* 0xBD000000STR	*/
264		LDR	ldr_imm_S,	/* 0xBD400000LDR	*/
265		STR	str_imm_X,	/* 0xF9000000STR	*/
266		LDR	ldr_imm_X,	/* 0xF9400000LDR	*/
268		STR	str_imm_D,	/* 0xFD000000STR	*/
269		LDR	ldr_imm_D,	/* 0xFD400000LDR	*/
267		PRFM	prfm_imm,	/* 0xF9800000PRFM	*
270	**Data processing – Immediate**		/* 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	add_imm_W,	/* 0x11000000ADD	
276		ADDS	adds_imm_W,	/* 0x31000000ADDS	
277		SUB	sub_imm_W,	/* 0x51000000SUB	
278		SUBS	subs_imm_W,	/* 0x71000000SUBS	
279		ADD	add_imm_X,	/* 0x91000000ADD	
280		ADDS	adds_imm_X,	/* 0xB1000000ADDS	
281		SUB	sub_imm_X,	/* 0xD1000000SUB	
282		SUBS	subs_imm_X,	/* 0xF1000000SUBS	
283	**Logical (immediate)**		/* Logical (immediate) */		
284		AND	and_imm_W,	/* 0x12000000AND	
285		ORR	orr_imm_W,	/* 0x32000000ORR	*
286		EOR	eor_imm_W,	/* 0x52000000EOR	
287		ANDS	ands_imm_W,	/* 0x72000000ANDS	
288		AND	and_imm_X,	/* 0x92000000AND	
289		ORR	orr_imm_X,	/* 0xB2000000ORR	*
290		EOR	eor_imm_X,	/* 0xD2000000EOR	
291		ANDS	ands_imm_X,	/* 0xF2000000ANDS	
292	**Move wide (immediate)**		/* Move wide (immediate) */		
293		MOVN	movn_W,	/* 0x12800000MOVN	
294		MOVZ	movz_W,	/* 0x52800000MOVZ	*
295		MOVK	movk_W,	/* 0x72800000MOVK	*
296		MOVN	movn_X,	/* 0x92800000MOVN	*
297		MOVZ	movz_X,	/* 0xD2800000MOVZ	*
298		MOVK	movk_X,	/* 0xF2800000MOVK	*
299	**Bitfield**		/* Bitfield */		

1	in_use	Opcode	//Opcode	BINARY	Ol
300		SBFM	sbfm_W,	/* 0x13000000SBFM	*/
301		BFM	bfm_W,	/* 0x33000000BFM	*/
302		UBFM	ubfm_W,	/* 0x53000000UBFM	*/
303		SBFM	sbfm_X,	/* 0x93400000SBFM	*/
304		BFM	bfm_X,	/* 0xB3400000BFM	*/
305		UBFM	ubfm_X,	/* 0xD3400000UBFM	*/
306		**Extract**	/* Extract */		
307		EXTR	extr_W,	/* 0x13800000EXTR	*/
308		EXTR	extr_X,	/* 0x93C00000EXTR	*/
309		**Data Processing – register**	/* Data Processing – register */		
310		**Logical (shifted register)**	/* Logical (shifted register) */		
311		AND	and_W,	/* 0x0A000000AND	*/
312		BIC	bic_W,	/* 0x0A200000BIC	*/
313		ORR	orr_W,	/* 0x2A000000ORR	*/
314		ORN	orn_W,	/* 0x2A200000ORN	*/
315		EOR	eor_W,	/* 0x4A000000EOR	*/
316		EON	eon_W,	/* 0x4A200000EON	*/
317		ANDS	ands_W,	/* 0x6A000000ANDS	*/
318		BICS	bics_W,	/* 0x6A200000BICS	*/
319		AND	and_X,	/* 0x8A000000AND	*/
320		BIC	bic_X,	/* 0x8A200000BIC	*/
321		ORR	orr_X,	/* 0xAA000000ORR	*/
322		ORN	orn_X,	/* 0xAA200000ORN	*/
323		EOR	eor_X,	/* 0xCA000000EOR	*/
324		EON	eon_X,	/* 0xCA200000EON	*/
325		ANDS	ands_X,	/* 0xEA000000ANDS	*/
326		BICS	bics_X,	/* 0xEA200000BICS	*/
327		**Add/subtract (shifted register)**	/* Add/subtract (shifted register) */		
328		ADD	add_W,	/* 0x0B000000ADD	*/
329		ADDS	adds_W,	/* 0x2B000000ADDS	*/
330		SUB	sub_W,	/* 0x4B000000SUB	*/
331		SUBS	subs_W,	/* 0x6B000000SUBS	*/
332		ADD	add_X,	/* 0x8B000000ADD	*/
333		ADDS	adds_X,	/* 0xAB000000ADDS	*/
334		SUB	sub_X,	/* 0xCB000000SUB	*/
335		SUBS	subs_X,	/* 0xEB000000SUBS	*/
336		**Add/subtract (extended register)**	/* Add/subtract (extended register) */		
337		ADD	add_ext_W,	/* 0x0B200000ADD	*

1	in_use	Opcode	//Opcode	BINARY	Ol
338		ADDS	adds_ext_W,	/* 0x2B200000ADDS	
339		SUB	sub_ext_W,	/* 0x4B200000SUB	*
340		SUBS	subs_ext_W,	/* 0x6B200000SUBS	
341		ADD	add_ext_X,	/* 0x8B200000ADD	*/
342		ADDS	adds_ext_X,	/* 0xAB200000ADDS	
343		SUB	sub_ext_X,	/* 0xCB200000SUB	*/
344		SUBS	subs_ext_X,	/* 0xEB200000SUBS	
345		**Add/subtract (with carry)**	/* Add/subtract (with carry) */		
346		ADC	adc_W,	/* 0x1A000000ADC	*/
347		ADCS	adcs_W,	/* 0x3A000000ADCS	*/
348		SBC	sbc_W,	/* 0x5A000000SBC	*/
349		SBCS	sbcs_W,	/* 0x7A000000SBCS	*/
350		ADC	adc_X,	/* 0x9A000000ADC	*/
351		ADCS	adcs_X,	/* 0xBA000000ADCS	*/
352		SBC	sbc_X,	/* 0xDA000000SBC	*/
353		SBCS	sbcs_X,	/* 0xFA000000SBCS	*/
354		**Conditional compare (register)**	/* Conditional compare (register) */		
355		CCMN	ccmn_W,	/* 0x3A400000CCMN	
356		CCMN	ccmn_X,	/* 0xBA400000CCMN	*
357		CCMP	ccmp_W,	/* 0x7A400000CCMP	
358		CCMP	ccmp_X,	/* 0xFA400000CCMP	*
359		**Conditional compare (immediate)**	/* Conditional compare (immediate) */		
360		CCMN	ccmn_imm_W,	/* 0x3A400800CCMN	
361		CCMN	ccmn_imm_X,	/* 0xBA400800CCMN	
362		CCMP	ccmp_imm_W,	/* 0x7A400800CCMP	
363		CCMP	ccmp_imm_X,	/* 0xFA400800CCMP	
364		**Conditional select**	/* Conditional select */		
365		CSEL	csel_W,	/* 0x1A800000CSEL	*/
366		CSINC	csinc_W,	/* 0x1A800400CSINC	*/
367		CSINV	csinv_W,	/* 0x5A800000CSINV	*/
368		CSNEG	csneg_W,	/* 0x5A800400CSNEG	
369		CSEL	csel_X,	/* 0x9A800000CSEL	*/
370		CSINC	csinc_X,	/* 0x9A800400CSINC	*/
371		CSINV	csinv_X,	/* 0xDA800000CSINV	*/
372		CSNEG	csneg_X,	/* 0xDA800400CSNEG	
373		**Data-processing (3 source)**	/* Data-processing (3 source) */		
374		MADD	madd_W,	/* 0x1B000000MADD	
375		MADD	madd_X,	/* 0x9B000000MADD	*

1	in_use	Opcode	//Opcode	BINARY	Ol
376		SMADDL	smaddl,	/* 0x9B200000SMADDL	
377		UMADDL	umaddl,	/* 0x9BA00000UMADDL	
378		MSUB	msub_W,	/* 0x1B008000MSUB	
379		MSUB	msub_X,	/* 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	udiv_W,	/* 0x1AC00800UDIV	*/
394		UDIV	udiv_X,	/* 0x9AC00800UDIV	*/
395		SDIV	sdiv_W,	/* 0x1AC00C00SDIV	*/
396		SDIV	sdiv_X,	/* 0x9AC00C00SDIV	*/
397		LSLV	lslv_W,	/* 0x1AC02000LSLV	*/
398		LSLV	lslv_X,	/* 0x9AC02000LSLV	*/
399		LSRV	lsrv_W,	/* 0x1AC02400LSRV	*/
400		LSRV	lsrv_X,	/* 0x9AC02400LSRV	*/
401		ASRV	asrv_W,	/* 0x1AC02800ASRV	*/
402		ASRV	asrv_X,	/* 0x9AC02800ASRV	*/
403		RORV	rorv_W,	/* 0x1AC02C00RORV	*/
404		RORV	rorv_X,	/* 0x9AC02C00RORV	*/
405		**Data-processing (1 source)**	/* Data-processing (1 source) */		
406		RBIT	rbit_W,	/* 0x5AC00000RBIT	*/
407		RBIT	rbit_X,	/* 0xDAC00000RBIT	*/
408		CLZ	clz_W,	/* 0x5AC01000CLZ	*/
409		CLZ	clz_X,	/* 0xDAC01000CLZ	*/
410		CLS	cls_W,	/* 0x5AC01400CLS	*/
411		CLS	cls_X,	/* 0xDAC01400CLS	*/
412		REV	rev_W,	/* 0x5AC00800REV	*/
413		REV	rev_X,	/* 0xDAC00C00REV	*/

1	in_use	Opcode	//Opcode	BINARY	OI
414		REV16	rev16_W,	/* 0xDAC00400REV16	
415		REV16	rev16_X,	/* 0x5AC00400REV16	*
416		REV32	rev32,	/* 0xDAC00800REV32	*/
417	//	**Data Processing – SIMD and floating p**	/* Data Processing – SIMD and floating point */		
418	//	**Floating-point<->fixed-point conversions**	/* Floating-point<->fixed-point conversions */		
419	//	SCVTF	//ARM64Op_scvtf_scalar_fixed_point_32_bit_to_single_precision,	/* (	
420	//	UCVTF	//ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_single_precision,	/* (	
421	//	FCVTZS	//ARM64Op_fcvtzs_scalar_fixed_point_Single_precision_to_32_bit,	/*	
422	//	FCVTZU	//ARM64Op_fcvtzu_scalar_fixed_point_Single_precision_to_32_bit,	/*	
423	//	SCVTF	//ARM64Op_scvtf_scalar_fixed_point_32_bit_to_double_precision,	/*	
424	//	UCVTF	//ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_double_precision,	/*	
425	//	FCVTZS	//ARM64Op_fcvtzs_scalar_fixed_point_Double_precision_to_32_bit,	/*	
426	//	FCVTZU	//ARM64Op_fcvtzu_scalar_fixed_point_Double_precision_to_32_bit,	/*	
427	//	SCVTF	//ARM64Op_scvtf_scalar_fixed_point_64_bit_to_single_precision,	/* (	
428	//	UCVTF	//ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_single_precision,	/* (	
429	//	FCVTZS	//ARM64Op_fcvtzs_scalar_fixed_point_Single_precision_to_64_bit,	/*	
430	//	FCVTZU	//ARM64Op_fcvtzu_scalar_fixed_point_Single_precision_to_64_bit,	/*	
431	//	SCVTF	//ARM64Op_scvtf_scalar_fixed_point_64_bit_to_double_precision,	/*	
432	//	UCVTF	//ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_double_precision,	/*	
433	//	FCVTZS	//ARM64Op_fcvtzs_scalar_fixed_point_Double_precision_to_64_bit,	/*	
434	//	FCVTZU	//ARM64Op_fcvtzu_scalar_fixed_point_Double_precision_to_64_bit,	/*	
435	//	**Floating-point conditional compare**	/* Floating-point conditional compare */		
436	//	FCCMP	//ARM64Op_fccmp_Single_precision,	/* 0x1E2004(	
437	//	FCCMPE	//ARM64Op_fccmpe_Single_precision,	/* 0x1E2004	
438	//	FCCMP	//ARM64Op_fccmp_Double_precision,	/* 0x1E6004	
439	//	FCCMPE	//ARM64Op_fccmpe_Double_precision,	/* 0x1E600-	
440	//	**Floating-point data-processing (2 source)**	/* Floating-point data-processing (2 source) */		
441	//	FMUL	//ARM64Op_fmul_scalar_Single_precision,	/* 0x1E200	
442	//	FDIV	//ARM64Op_fdiv_scalar_Single_precision,	/* 0x1E201(	
443	//	FADD	//ARM64Op_fadd_scalar_Single_precision,	/* 0x1E202	
444	//	FSUB	//ARM64Op_fsub_scalar_Single_precision,	/* 0x1E203	
445	//	FMAX	//ARM64Op_fmax_scalar_Single_precision,	/* 0x1E20-	
446	//	FMIN	//ARM64Op_fmin_scalar_Single_precision,	/* 0x1E205	
447	//	FMAXNM	//ARM64Op_fmaxnm_scalar_Single_precision,	/* 0x1E2	

1	in_use	Opcode	//Opcode	BINARY OI
448	//	FMINNM	//ARM64Op_fminnm_scalar_Single_precision,	/* 0x1E2(
449	//	FNMUL	//ARM64Op_fnmul_Single_precision,	/* 0x1E20880
450	//	FMUL	//ARM64Op_fmul_scalar_Double_precision,	/* 0x1E60(
451	//	FDIV	//ARM64Op_fdiv_scalar_Double_precision,	/* 0x1E601
452	//	FADD	//ARM64Op_fadd_scalar_Double_precision,	/* 0x1E60:
453	//	FSUB	//ARM64Op_fsub_scalar_Double_precision,	/* 0x1E603
454	//	FMAX	//ARM64Op_fmax_scalar_Double_precision,	/* 0x1E60
455	//	FMIN	//ARM64Op_fmin_scalar_Double_precision,	/* 0x1E605
456	//	FMAXNM	//ARM64Op_fmaxnm_scalar_Double_precision,	/* 0x1E6
457	//	FMINNM	//ARM64Op_fminnm_scalar_Double_precision,	/* 0x1E6
458	//	FNMUL	//ARM64Op_fnmul_Double_precision,	/* 0x1E6088(
459	//	**Floating-point conditional select**	/* Floating-point conditional select */	
460	//	FCSEL	//ARM64Op_fcsel_Single_precision,	/* 0x1E200C0(
461	//	FCSEL	//ARM64Op_fcsel_Double_precision,	/* 0x1E600C0
462	//	**Floating-point immediate**	/* Floating-point immediate */	
463	//	FMOV	//ARM64Op_fmov_scalar_immediate_Single_precision,	/* 0x'
464	//	FMOV	//ARM64Op_fmov_scalar_immediate_Double_precision,	/* 0x
465	//	**Floating-point compare**	/* Floating-point compare */	
466	//	FCMP	//ARM64Op_fcmp_Single_precision,	/* 0x1E20200
467	//	FCMP	//ARM64Op_fcmp_Single_precision_zero,	/* 0x1E202
468	//	FCMPE	//ARM64Op_fcmpe_Single_precision,	/* 0x1E2020'
469	//	FCMPE	//ARM64Op_fcmpe_Single_precision_zero,	/* 0x1E20:
470	//	FCMP	//ARM64Op_fcmp_Double_precision,	/* 0x1E6020(
471	//	FCMP	//ARM64Op_fcmp_Double_precision_zero,	/* 0x1E60:
472	//	FCMPE	//ARM64Op_fcmpe_Double_precision,	/* 0x1E6020
473	//	FCMPE	//ARM64Op_fcmpe_Double_precision_zero,	/* 0x1E60
474	//	**Floating-point data-processing (1 source)**	/* Floating-point data-processing (1 source) */	
475	//	FMOV	//ARM64Op_fmov_register_Single_precision,	/* 0x1E20-
476	//	FABS	//ARM64Op_fabs_scalar_Single_precision,	/* 0x1E20C
477	//	FNEG	//ARM64Op_fneg_scalar_Single_precision,	/* 0x1E214
478	//	FSQRT	//ARM64Op_fsqrt_scalar_Single_precision,	/* 0x1E21C(
479	//	FCVT	//ARM64Op_fcvt_Single_precision_to_double_precision,	/* 0x1
480	//	FCVT	//ARM64Op_fcvt_Single_precision_to_half_precision,	/* 0x1E2
481	//	FRINTN	//ARM64Op_frintn_scalar_Single_precision,	/* 0x1E244(

1	in_use	Opcode	//Opcode	BINARY	OI
482	//	FRINTP	//ARM64Op_frintp_scalar_Single_precision,	/* 0x1E24C(	
483	//	FRINTM	//ARM64Op_frintm_scalar_Single_precision,	/* 0x1E254	
484	//	FRINTZ	//ARM64Op_frintz_scalar_Single_precision,	/* 0x1E25C(	
485	//	FRINTA	//ARM64Op_frinta_scalar_Single_precision,	/* 0x1E264(	
486	//	FRINTX	//ARM64Op_frintx_scalar_Single_precision,	/* 0x1E274(	
487	//	FRINTI	//ARM64Op_frinti_scalar_Single_precision,	/* 0x1E27C0	
488	//	FMOV	//ARM64Op_fmov_register_Double_precision,	/* 0x1E60	
489	//	FABS	//ARM64Op_fabs_scalar_Double_precision,	/* 0x1E60(	
490	//	FNEG	//ARM64Op_fneg_scalar_Double_precision,	/* 0x1E61⁴	
491	//	FSQRT	//ARM64Op_fsqrt_scalar_Double_precision,	/* 0x1E61C	
492	//	FCVT	//ARM64Op_fcvt_Double_precision_to_single_precision,	/* 0x1	
493	//	FCVT	//ARM64Op_fcvt_Double_precision_to_half_precision,	/* 0x1E	
494	//	FRINTN	//ARM64Op_frintn_scalar_Double_precision,	/* 0x1E644	
495	//	FRINTP	//ARM64Op_frintp_scalar_Double_precision,	/* 0x1E64C	
496	//	FRINTM	//ARM64Op_frintm_scalar_Double_precision,	/* 0x1E65⁴	
497	//	FRINTZ	//ARM64Op_frintz_scalar_Double_precision,	/* 0x1E65C	
498	//	FRINTA	//ARM64Op_frinta_scalar_Double_precision,	/* 0x1E664	
499	//	FRINTX	//ARM64Op_frintx_scalar_Double_precision,	/* 0x1E674	
500	//	FRINTI	//ARM64Op_frinti_scalar_Double_precision,	/* 0x1E67C	
501	//	FCVT	//ARM64Op_fcvt_Half_precision_to_single_precision,	/* 0x1EI	
502	//	FCVT	//ARM64Op_fcvt_Half_precision_to_double_precision,	/* 0x1E	
503	//	**Floating-point<->integer conversions**	/* Floating-point<->integer conversions */		
504	//	FCVTNS	//ARM64Op_fcvtns_scalar_Single_precision_to_32_bit,	/* 0x1E	
505	//	FCVTNU	//ARM64Op_fcvtnu_scalar_Single_precision_to_32_bit,	/* 0x1E	
506	//	SCVTF	//ARM64Op_scvtf_scalar_integer_32_bit_to_single_precision,	/* 0x	
507	//	UCVTF	//ARM64Op_ucvtf_scalar_integer_32_bit_to_single_precision,	/* 0x	
508	//	FCVTAS	//ARM64Op_fcvtas_scalar_Single_precision_to_32_bit,	/* 0x1E	
509	//	FCVTAU	//ARM64Op_fcvtau_scalar_Single_precision_to_32_bit,	/* 0x1E	
510	//	FMOV	//ARM64Op_fmov_general_Single_precision_to_32_bit,	/* 0x1	
511	//	FMOV	//ARM64Op_fmov_general_32_bit_to_single_precision,	/* 0x1	
512	//	FCVTPS	//ARM64Op_fcvtps_scalar_Single_precision_to_32_bit,	/* 0x1E	
513	//	FCVTPU	//ARM64Op_fcvtpu_scalar_Single_precision_to_32_bit,	/* 0x1E	
514	//	FCVTMS	//ARM64Op_fcvtms_scalar_Single_precision_to_32_bit,	/* 0x1	
515	//	FCVTMU	//ARM64Op_fcvtmu_scalar_Single_precision_to_32_bit,	/* 0x1	

1	in_use	Opcode	//Opcode	BINARY	OI
516	//	FCVTZS	//ARM64Op_fcvtzs_scalar_integer_Single_precision_to_32_bit,	/* 0	
517	//	FCVTZU	//ARM64Op_fcvtzu_scalar_integer_Single_precision_to_32_bit,	/* 0	
518	//	FCVTNS	//ARM64Op_fcvtns_scalar_Double_precision_to_32_bit,	/* 0x1	
519	//	FCVTNU	//ARM64Op_fcvtnu_scalar_Double_precision_to_32_bit,	/* 0x1	
520	//	SCVTF	//ARM64Op_scvtf_scalar_integer_32_bit_to_double_precision,	/* 0	
521	//	UCVTF	//ARM64Op_ucvtf_scalar_integer_32_bit_to_double_precision,	/* 0	
522	//	FCVTAS	//ARM64Op_fcvtas_scalar_Double_precision_to_32_bit,	/* 0x1	
523	//	FCVTAU	//ARM64Op_fcvtau_scalar_Double_precision_to_32_bit,	/* 0x1	
524	//	FCVTPS	//ARM64Op_fcvtps_scalar_Double_precision_to_32_bit,	/* 0x1	
525	//	FCVTPU	//ARM64Op_fcvtpu_scalar_Double_precision_to_32_bit,	/* 0x1	
526	//	FCVTMS	//ARM64Op_fcvtms_scalar_Double_precision_to_32_bit,	/* 0x	
527	//	FCVTMU	//ARM64Op_fcvtmu_scalar_Double_precision_to_32_bit,	/* 0x	
528	//	FCVTZS	//ARM64Op_fcvtzs_scalar_integer_Double_precision_to_32_bit,	/* (	
529	//	FCVTZU	//ARM64Op_fcvtzu_scalar_integer_Double_precision_to_32_bit,	/* (	
530	//	FCVTNS	//ARM64Op_fcvtns_scalar_Single_precision_to_64_bit,	/* 0x9E	
531	//	FCVTNU	//ARM64Op_fcvtnu_scalar_Single_precision_to_64_bit,	/* 0x9E	
532	//	SCVTF	//ARM64Op_scvtf_scalar_integer_64_bit_to_single_precision,	/* 0x	
533	//	UCVTF	//ARM64Op_ucvtf_scalar_integer_64_bit_to_single_precision,	/* 0x	
534	//	FCVTAS	//ARM64Op_fcvtas_scalar_Single_precision_to_64_bit,	/* 0x9E	
535	//	FCVTAU	//ARM64Op_fcvtau_scalar_Single_precision_to_64_bit,	/* 0x9E	
536	//	FCVTPS	//ARM64Op_fcvtps_scalar_Single_precision_to_64_bit,	/* 0x9E	
537	//	FCVTPU	//ARM64Op_fcvtpu_scalar_Single_precision_to_64_bit,	/* 0x9E	
538	//	FCVTMS	//ARM64Op_fcvtms_scalar_Single_precision_to_64_bit,	/* 0x9	
539	//	FCVTMU	//ARM64Op_fcvtmu_scalar_Single_precision_to_64_bit,	/* 0x9	
540	//	FCVTZS	//ARM64Op_fcvtzs_scalar_integer_Single_precision_to_64_bit,	/* 0	
541	//	FCVTZU	//ARM64Op_fcvtzu_scalar_integer_Single_precision_to_64_bit,	/* 0	
542	//	FCVTNS	//ARM64Op_fcvtns_scalar_Double_precision_to_64_bit,	/* 0x9	
543	//	FCVTNU	//ARM64Op_fcvtnu_scalar_Double_precision_to_64_bit,	/* 0x9	
544	//	SCVTF	//ARM64Op_scvtf_scalar_integer_64_bit_to_double_precision,	/* 0	
545	//	UCVTF	//ARM64Op_ucvtf_scalar_integer_64_bit_to_double_precision,	/* 0	
546	//	FCVTAS	//ARM64Op_fcvtas_scalar_Double_precision_to_64_bit,	/* 0x9	
547	//	FCVTAU	//ARM64Op_fcvtau_scalar_Double_precision_to_64_bit,	/* 0x9	
548	//	FMOV	//ARM64Op_fmov_general_Double_precision_to_64_bit,	/* 0x	
549	//	FMOV	//ARM64Op_fmov_general_64_bit_to_double_precision,	/* 0x	

1	in_use	Opcode	//Opcode	BINARY	Ol
550	//	FCVTPS	//ARM64Op_fcvtps_scalar_Double_precision_to_64_bit,	/* 0x9	
551	//	FCVTPU	//ARM64Op_fcvtpu_scalar_Double_precision_to_64_bit,	/* 0x9	
552	//	FCVTMS	//ARM64Op_fcvtms_scalar_Double_precision_to_64_bit,	/* 0x9	
553	//	FCVTMU	//ARM64Op_fcvtmu_scalar_Double_precision_to_64_bit,	/* 0x9	
554	//	FCVTZS	//ARM64Op_fcvtzs_scalar_integer_Double_precision_to_64_bit,	/* (	
555	//	FCVTZU	//ARM64Op_fcvtzu_scalar_integer_Double_precision_to_64_bit,	/* (	
556	//	FMOV	//ARM64Op_fmov_general_Top_half_of_128_bit_to_64_bit,	/* 0	
557	//	FMOV	//ARM64Op_fmov_general_64_bit_to_top_half_of_128_bit,	/* 0x	
558	//	**Floating-point data-processing (3 source)**	/* Floating-point data-processing (3 source) */		
559	//	FMADD	//ARM64Op_fmadd_Single_precision,	/* 0x1F0000(	
560	//	FMSUB	//ARM64Op_fmsub_Single_precision,	/* 0x1F0080(	
561	//	FNMADD	//ARM64Op_fnmadd_Single_precision,	/* 0x1F2000	
562	//	FNMSUB	//ARM64Op_fnmsub_Single_precision,	/* 0x1F2080	
563	//	FMADD	//ARM64Op_fmadd_Double_precision,	/* 0x1F400(	
564	//	FMSUB	//ARM64Op_fmsub_Double_precision,	/* 0x1F408(	
565	//	FNMADD	//ARM64Op_fnmadd_Double_precision,	/* 0x1F600(	
566	//	FNMSUB	//ARM64Op_fnmsub_Double_precision,	/* 0x1F608(	
567	//	**AdvSIMD scalar three same**	/* AdvSIMD scalar three same */		
568	//	SQADD	//ARM64Op_sqadd_Scalar,	/* 0x5E200C00S	
569	//	SQSUB	//ARM64Op_sqsub_Scalar,	/* 0x5E202C00S(	
570	//	CMGT	//ARM64Op_cmgt_register_Scalar,	/* 0x5E20340(	
571	//	CMGE	//ARM64Op_cmge_register_Scalar,	/* 0x5E203C(	
572	//	SSHL	//ARM64Op_sshl_Scalar,	/* 0x5E204400SSI	
573	//	SQSHL	//ARM64Op_sqshl_register_Scalar,	/* 0x5E204C0(	
574	//	SRSHL	//ARM64Op_srshl_Scalar,	/* 0x5E205400SR	
575	//	SQRSHL	//ARM64Op_sqrshl_Scalar,	/* 0x5E205C00S(	
576	//	ADD	//ARM64Op_add_vector_Scalar,	/* 0x5E208400	
577	//	CMTST	//ARM64Op_cmtst_Scalar,	/* 0x5E208C00C\	
578	//	SQDMULH	//ARM64Op_sqdmulh_vector_Scalar,	/* 0x5E20B4	
579	//	FMULX	//ARM64Op_fmulx_Scalar,	/* 0x5E20DC00F\	
580	//	FCMEQ	//ARM64Op_fcmeq_register_Scalar,	/* 0x5E20E40	
581	//	FRECPS	//ARM64Op_frecps_Scalar,	/* 0x5E20FC00F\	
582	//	FRSQRTS	//ARM64Op_frsqrts_Scalar,	/* 0x5EA0FC00F\	
583	//	UQADD	//ARM64Op_uqadd_Scalar,	/* 0x7E200C00U	

1	in_use	Opcode	//Opcode	BINARY OI
584	//	UQSUB	//ARM64Op_uqsub_Scalar,	/* 0x7E202C00U
585	//	CMHI	//ARM64Op_cmhi_register_Scalar,	/* 0x7E20340(
586	//	CMHS	//ARM64Op_cmhs_register_Scalar,	/* 0x7E203C0
587	//	USHL	//ARM64Op_ushl_Scalar,	/* 0x7E204400US
588	//	UQSHL	//ARM64Op_uqshl_register_Scalar,	/* 0x7E204C0(
589	//	URSHL	//ARM64Op_urshl_Scalar,	/* 0x7E205400UR
590	//	UQRSHL	//ARM64Op_uqrshl_Scalar,	/* 0x7E205C00U(
591	//	SUB	//ARM64Op_sub_vector_Scalar,	/* 0x7E208400
592	//	CMEQ	//ARM64Op_cmeq_register_Scalar,	/* 0x7E208C0
593	//	SQRDMULH	//ARM64Op_sqrdmulh_vector_Scalar,	/* 0x7E20B4
594	//	FCMGE	//ARM64Op_fcmge_register_Scalar,	/* 0x7E20E40
595	//	FACGE	//ARM64Op_facge_Scalar,	/* 0x7E20EC00F/
596	//	FABD	//ARM64Op_fabd_Scalar,	/* 0x7EA0D400FA
597	//	FCMGT	//ARM64Op_fcmgt_register_Scalar,	/* 0x7EA0E40
598	//	FACGT	//ARM64Op_facgt_Scalar,	/* 0x7EA0EC00FA
599	//	**AdvSIMD scalar three different**	/* AdvSIMD scalar three different */	
600	//	SQDMLAL	//ARM64Op_sqdmlal_vector_Scalar,	/* 0x5E20900
601	//	SQDMLAL2	//ARM64Op_sqdmlal2_vector_Scalar,	/* 0x5E2090(
602	//	SQDMLSL	//ARM64Op_sqdmlsl_vector_Scalar,	/* 0x5E20B0(
603	//	SQDMLSL2	//ARM64Op_sqdmlsl2_vector_Scalar,	/* 0x5E20B0(
604	//	SQDMULL	//ARM64Op_sqdmull_vector_Scalar,	/* 0x5E20D0(
605	//	SQDMULL2	//ARM64Op_sqdmull2_vector_Scalar,	/* 0x5E20D0
606	//	**AdvSIMD scalar two-reg misc**	/* AdvSIMD scalar two-reg misc */	
607	//	SUQADD	//ARM64Op_suqadd_Scalar,	/* 0x5E203800S
608	//	SQABS	//ARM64Op_sqabs_Scalar,	/* 0x5E207800S(
609	//	CMGT	//ARM64Op_cmgt_zero_Scalar,	/* 0x5E208800
610	//	CMEQ	//ARM64Op_cmeq_zero_Scalar,	/* 0x5E20980(
611	//	CMLT	//ARM64Op_cmlt_zero_Scalar,	/* 0x5E20A800(
612	//	ABS	//ARM64Op_abs_Scalar,	/* 0x5E20B800AB
613	//	SQXTN	//ARM64Op_sqxtn_Scalar,	/* 0x5E214800SC
614	//	SQXTN2	//ARM64Op_sqxtn2_Scalar,	/* 0x5E214800S(
615	//	FCVTNS	//ARM64Op_fcvtns_vector_Scalar,	/* 0x5E21A80(
616	//	FCVTMS	//ARM64Op_fcvtms_vector_Scalar,	/* 0x5E21B80
617	//	FCVTAS	//ARM64Op_fcvtas_vector_Scalar,	/* 0x5E21C80(

1	in_use	Opcode	//Opcode	BINARY OI
618	//	SCVTF	//ARM64Op_scvtf_vector_integer_Scalar,	/* 0x5E21D8
619	//	FCMGT	//ARM64Op_fcmgt_zero_Scalar,	/* 0x5EA0C80(
620	//	FCMEQ	//ARM64Op_fcmeq_zero_Scalar,	/* 0x5EA0D80
621	//	FCMLT	//ARM64Op_fcmlt_zero_Scalar,	/* 0x5EA0E800
622	//	FCVTPS	//ARM64Op_fcvtps_vector_Scalar,	/* 0x5EA1A80(
623	//	FCVTZS	//ARM64Op_fcvtzs_vector_integer_Scalar,	/* 0x5EA1B
624	//	FRECPE	//ARM64Op_frecpe_Scalar,	/* 0x5EA1D800FI
625	//	FRECPX	//ARM64Op_frecpx,	/* 0x5EA1F800FREC
626	//	USQADD	//ARM64Op_usqadd_Scalar,	/* 0x7E203800U
627	//	SQNEG	//ARM64Op_sqneg_Scalar,	/* 0x7E207800S(
628	//	CMGE	//ARM64Op_cmge_zero_Scalar,	/* 0x7E20880(
629	//	CMLE	//ARM64Op_cmle_zero_Scalar,	/* 0x7E209800
630	//	NEG	//ARM64Op_neg_vector_Scalar,	/* 0x7E20B800
631	//	SQXTUN	//ARM64Op_sqxtun_Scalar,	/* 0x7E212800S(
632	//	SQXTUN2	//ARM64Op_sqxtun2_Scalar,	/* 0x7E212800S
633	//	UQXTN	//ARM64Op_uqxtn_Scalar,	/* 0x7E214800UC
634	//	UQXTN2	//ARM64Op_uqxtn2_Scalar,	/* 0x7E214800U(
635	//	FCVTXN	//ARM64Op_fcvtxn_Scalar,	/* 0x7E216800FC
636	//	FCVTXN2	//ARM64Op_fcvtxn2_Scalar,	/* 0x7E216800F(
637	//	FCVTNU	//ARM64Op_fcvtnu_vector_Scalar,	/* 0x7E21A80(
638	//	FCVTMU	//ARM64Op_fcvtmu_vector_Scalar,	/* 0x7E21B80
639	//	FCVTAU	//ARM64Op_fcvtau_vector_Scalar,	/* 0x7E21C80(
640	//	UCVTF	//ARM64Op_ucvtf_vector_integer_Scalar,	/* 0x7E21D8
641	//	FCMGE	//ARM64Op_fcmge_zero_Scalar,	/* 0x7EA0C80
642	//	FCMLE	//ARM64Op_fcmle_zero_Scalar,	/* 0x7EA0D80(
643	//	FCVTPU	//ARM64Op_fcvtpu_vector_Scalar,	/* 0x7EA1A80(
644	//	FCVTZU	//ARM64Op_fcvtzu_vector_integer_Scalar,	/* 0x7EA1B
645	//	FRSQRTE	//ARM64Op_frsqrte_Scalar,	/* 0x7EA1D800FF
646	//	**AdvSIMD scalar pairwise**	/* AdvSIMD scalar pairwise */	
647	//	ADDP	//ARM64Op_addp_scalar,	/* 0x5E31B800AI
648	//	FMAXNMP	//ARM64Op_fmaxnmp_scalar,	/* 0x7E30C800
649	//	FADDP	//ARM64Op_faddp_scalar,	/* 0x7E30D800FA
650	//	FMAXP	//ARM64Op_fmaxp_scalar,	/* 0x7E30F800FM
651	//	FMINNMP	//ARM64Op_fminnmp_scalar,	/* 0x7EB0C800I

1	in_use	Opcode	//Opcode	BINARY	Ol
652	//	FMINP	//ARM64Op_fminp_scalar,	/* 0x7EB0F800FM	
653	//	**AdvSIMD scalar copy**	/* AdvSIMD scalar copy */		
654	//	DUP	//ARM64Op_dup_element_Scalar,	/* 0x5E00040	
655	//	**AdvSIMD scalar x indexed element**	/* AdvSIMD scalar x indexed element */		
656	//	SQDMLAL	//ARM64Op_sqdmlal_by_element_Scalar,	/* 0x5F003	
657	//	SQDMLAL2	//ARM64Op_sqdmlal2_by_element_Scalar,	/* 0x5F00	
658	//	SQDMLSL	//ARM64Op_sqdmlsl_by_element_Scalar,	/* 0x5F007	
659	//	SQDMLSL2	//ARM64Op_sqdmlsl2_by_element_Scalar,	/* 0x5F00	
660	//	SQDMULL	//ARM64Op_sqdmull_by_element_Scalar,	/* 0x5F00E	
661	//	SQDMULL2	//ARM64Op_sqdmull2_by_element_Scalar,	/* 0x5F00	
662	//	SQDMULH	//ARM64Op_sqdmulh_by_element_Scalar,	/* 0x5F00	
663	//	SQRDMULH	//ARM64Op_sqrdmulh_by_element_Scalar,	/* 0x5F00	
664	//	FMLA	//ARM64Op_fmla_by_element_Scalar,	/* 0x5F8010	
665	//	FMLS	//ARM64Op_fmls_by_element_Scalar,	/* 0x5F8050	
666	//	FMUL	//ARM64Op_fmul_by_element_Scalar,	/* 0x5F8090	
667	//	FMULX	//ARM64Op_fmulx_by_element_Scalar,	/* 0x7F809(	
668	//	**AdvSIMD scalar shift by immediate**	/* AdvSIMD scalar shift by immediate */		
669	//	SSHR	//ARM64Op_sshr_Scalar,	/* 0x5F000400SSI	
670	//	SSRA	//ARM64Op_ssra_Scalar,	/* 0x5F001400SSI	
671	//	SRSHR	//ARM64Op_srshr_Scalar,	/* 0x5F002400SR	
672	//	SRSRA	//ARM64Op_srsra_Scalar,	/* 0x5F003400SR	
673	//	SHL	//ARM64Op_shl_Scalar,	/* 0x5F005400SHL	
674	//	SQSHL	//ARM64Op_sqshl_immediate_Scalar,	/* 0x5F0074	
675	//	SQSHRN	//ARM64Op_sqshrn_Scalar,	/* 0x5F009400S(	
676	//	SQSHRN2	//ARM64Op_sqshrn2_Scalar,	/* 0x5F009400S	
677	//	SQRSHRN	//ARM64Op_sqrshrn_Scalar,	/* 0x5F009C00S	
678	//	SQRSHRN2	//ARM64Op_sqrshrn2_Scalar,	/* 0x5F009C00S	
679	//	SCVTF	//ARM64Op_scvtf_vector_fixed_point_Scalar,	/* 0x5F00E	
680	//	FCVTZS	//ARM64Op_fcvtzs_vector_fixed_point_Scalar,	/* 0x5F00I	
681	//	USHR	//ARM64Op_ushr_Scalar,	/* 0x7F000400US	
682	//	USRA	//ARM64Op_usra_Scalar,	/* 0x7F001400US	
683	//	URSHR	//ARM64Op_urshr_Scalar,	/* 0x7F002400UR	
684	//	URSRA	//ARM64Op_ursra_Scalar,	/* 0x7F003400UR	
685	//	SRI	//ARM64Op_sri_Scalar,	/* 0x7F004400SRI	

1	in_use	Opcode	//Opcode	BINARY	OI
686	//	SLI	//ARM64Op_sli_Scalar,	/* 0x7F005400SLI	
687	//	SQSHLU	//ARM64Op_sqshlu_Scalar,	/* 0x7F006400SC	
688	//	UQSHL	//ARM64Op_uqshl_immediate_Scalar,	/* 0x7F0074	
689	//	SQSHRUN	//ARM64Op_sqshrun_Scalar,	/* 0x7F008400S	
690	//	SQSHRUN2	//ARM64Op_sqshrun2_Scalar,	/* 0x7F008400S	
691	//	SQRSHRUN	//ARM64Op_sqrshrun_Scalar,	/* 0x7F008C00S	
692	//	SQRSHRUN2	//ARM64Op_sqrshrun2_Scalar,	/* 0x7F008C00S	
693	//	UQSHRN	//ARM64Op_uqshrn_Scalar,	/* 0x7F009400U	
694	//	UQRSHRN	//ARM64Op_uqrshrn_Scalar,	/* 0x7F009C00U	
695	//	UQRSHRN2	//ARM64Op_uqrshrn2_Scalar,	/* 0x7F009C00U	
696	//	UCVTF	//ARM64Op_ucvtf_vector_fixed_point_Scalar,	/* 0x7F00E	
697	//	FCVTZU	//ARM64Op_fcvtzu_vector_fixed_point_Scalar,	/* 0x7F00I	
698	//	**Crypto three-reg SHA**	/* Crypto three-reg SHA */		
699	//	SHA1C	//ARM64Op_sha1c,	/* 0x5E000000SHA1	
700	//	SHA1P	//ARM64Op_sha1p,	/* 0x5E001000SHA1	
701	//	SHA1M	//ARM64Op_sha1m,	/* 0x5E002000SHA	
702	//	SHA1SU0	//ARM64Op_sha1su0,	/* 0x5E003000SHA	
703	//	SHA256H	//ARM64Op_sha256h,	/* 0x5E004000SHA	
704	//	SHA256H2	//ARM64Op_sha256h2,	/* 0x5E005000SH	
705	//	SHA256SU1	//ARM64Op_sha256su1,	/* 0x5E006000SH	
706	//	**Crypto two-reg SHA**	/* Crypto two-reg SHA */		
707	//	SHA1H	//ARM64Op_sha1h,	/* 0x5E280800SHA1	
708	//	SHA1SU1	//ARM64Op_sha1su1,	/* 0x5E281800SHA	
709	//	SHA256SU0	//ARM64Op_sha256su0,	/* 0x5E282800SH	
710	//	**Crypto AES**	/* Crypto AES */		
711	//	AESE	//ARM64Op_aese,	/* 0x4E284800AESE	
712	//	AESD	//ARM64Op_aesd,	/* 0x4E285800AESD	
713	//	AESMC	//ARM64Op_aesmc,	/* 0x4E286800AES	
714	//	AESIMC	//ARM64Op_aesimc,	/* 0x4E287800AESI	
715	//	**AdvSIMD three same**	/* AdvSIMD three same */		
716	//	SHADD	//ARM64Op_shadd,	/* 0x0E200400SHAI	
717	//	SQADD	//ARM64Op_sqadd_Vector,	/* 0x0E200C00S	
718	//	SRHADD	//ARM64Op_srhadd,	/* 0x0E201400SRH	
719	//	SHSUB	//ARM64Op_shsub,	/* 0x0E202400SHSU	

1	in_use	Opcode	//Opcode	BINARY Ol
720	//	SQSUB	//ARM64Op_sqsub_Vector,	/* 0x0E202C00S
721	//	CMGT	//ARM64Op_cmgt_register_Vector,	/* 0x0E20340(
722	//	CMGE	//ARM64Op_cmge_register_Vector,	/* 0x0E203C(
723	//	SSHL Vector	//ARM64Op_sshl vector,	/* 0x0E204400SSH
724	//	SQSHL	//ARM64Op_sqshl_register_Vector,	/* 0x0E204C0(
725	//	SRSHL	//ARM64Op_srshl_Vector,	/* 0x0E205400SR
726	//	SQRSHL	//ARM64Op_sqrshl_Vector,	/* 0x0E205C00S(
727	//	SMAX	//ARM64Op_smax,	/* 0x0E206400SMA)
728	//	SMIN	//ARM64Op_smin,	/* 0x0E206C00SMIN
729	//	SABD	//ARM64Op_sabd,	/* 0x0E207400SABD
730	//	SABA	//ARM64Op_saba,	/* 0x0E207C00SABA
731	//	ADD	//ARM64Op_add_vector_Vector,	/* 0x0E208400
732	//	CMTST	//ARM64Op_cmtst_Vector,	/* 0x0E208C00CI
733	//	MLA	//ARM64Op_mla_vector,	/* 0x0E209400ML/
734	//	MUL	//ARM64Op_mul_vector,	/* 0x0E209C00MU
735	//	SMAXP	//ARM64Op_smaxp,	/* 0x0E20A400SMA
736	//	SMINP	//ARM64Op_sminp,	/* 0x0E20AC00SMI
737	//	SQDMULH	//ARM64Op_sqdmulh_vector_Vector,	/* 0x0E20B4
738	//	ADDP	//ARM64Op_addp_vector,	/* 0x0E20BC00AI
739	//	FMAXNM	//ARM64Op_fmaxnm_vector,	/* 0x0E20C400I
740	//	FMLA	//ARM64Op_fmla_vector,	/* 0x0E20CC00FM
741	//	FADD	//ARM64Op_fadd_vector,	/* 0x0E20D400FA
742	//	FMULX	//ARM64Op_fmulx_Vector,	/* 0x0E20DC00FI
743	//	FCMEQ	//ARM64Op_fcmeq_register_Vector,	/* 0x0E20E4(
744	//	FMAX	//ARM64Op_fmax_vector,	/* 0x0E20F400FM
745	//	FRECPS	//ARM64Op_frecps_Vector,	/* 0x0E20FC00FI
746	//	AND	//ARM64Op_and_vector,	/* 0x0E201C00AN
747	//	BIC	//ARM64Op_bic_vector_register,	/* 0x0E601C00I
748	//	FMINNM	//ARM64Op_fminnm_vector,	/* 0x0EA0C400I
749	//	FMLS	//ARM64Op_fmls_vector,	/* 0x0EA0CC00FM
750	//	FSUB	//ARM64Op_fsub_vector,	/* 0x0EA0D400FS
751	//	FMIN	//ARM64Op_fmin_vector,	/* 0x0EA0F400FM
752	//	FRSQRTS	//ARM64Op_frsqrts_Vector,	/* 0x0EA0FC00FI
753	//	ORR	//ARM64Op_orr_vector_register,	/* 0x0EA01C00(

1	in_use	Opcode	//Opcode	BINARY	Ol
754	//	ORN	//ARM64Op_orn_vector,	/* 0x0EE01C00	OR
755	//	UHADD	//ARM64Op_uhadd,	/* 0x2E200400	UHA
756	//	UQADD	//ARM64Op_uqadd_Vector,	/* 0x2E200C00	U
757	//	URHADD	//ARM64Op_urhadd,	/* 0x2E201400	URH
758	//	UHSUB	//ARM64Op_uhsub,	/* 0x2E202400	UHS
759	//		//ARM64Op_ _Vector,	/* 0x2E202C00	
760	//	CMHI	//ARM64Op_cmhi_register_Vector,	/* 0x2E2034	0
761	//	CMHS	//ARM64Op_cmhs_register_Vector,	/* 0x2E203C	0
762	//	USHL	//ARM64Op_ushl_Vector,	/* 0x2E204400	US
763	//	UQSHL	//ARM64Op_uqshl_register_Vector,	/* 0x2E204C	0
764	//	URSHL	//ARM64Op_urshl_Vector,	/* 0x2E205400	UR
765	//	UQRSHL	//ARM64Op_uqrshl_Vector,	/* 0x2E205C00	U
766	//	UMAX	//ARM64Op_umax,	/* 0x2E206400	UMA
767	//	UMIN	//ARM64Op_umin,	/* 0x2E206C00	UMIN
768	//	UABD	//ARM64Op_uabd,	/* 0x2E207400	UABD
769	//	UABA	//ARM64Op_uaba,	/* 0x2E207C00	UABA
770	//	SUB	//ARM64Op_sub_vector_Vector,	/* 0x2E208400	
771	//	CMEQ	//ARM64Op_cmeq_register_Vector,	/* 0x2E208C	0
772	//	MLS	//ARM64Op_mls_vector,	/* 0x2E209400	ML
773	//	PMUL	//ARM64Op_pmul,	/* 0x2E209C00	PMU
774	//	UMAXP	//ARM64Op_umaxp,	/* 0x2E20A400	UMA
775	//	UMINP	//ARM64Op_uminp,	/* 0x2E20AC00	UMI
776	//	SQRDMULH	//ARM64Op_sqrdmulh_vector_Vector,	/* 0x2E20B4	
777	//	FMAXNMP	//ARM64Op_fmaxnmp_vector,	/* 0x2E20B400	
778	//	FADDP	//ARM64Op_faddp_vector,	/* 0x2E20D400	FA
779	//	FMUL	//ARM64Op_fmul_vector,	/* 0x2E20DC00	FM
780	//	FCMGE	//ARM64Op_fcmge_register_Vector,	/* 0x2E20E4	0
781	//	FACGE	//ARM64Op_facge_Vector,	/* 0x2E20EC00	F
782	//	FMAXP	//ARM64Op_fmaxp_vector,	/* 0x2E20F400	FM
783	//	FDIV	//ARM64Op_fdiv_vector,	/* 0x2E20FC00	FDI
784	//	EOR	//ARM64Op_eor_vector,	/* 0x2E201C00	EO
785	//	BSL	//ARM64Op_bsl,	/* 0x2E601C00	BSL
786	//	FMINNMP	//ARM64Op_fminnmp_vector,	/* 0x2EA0C400	
787	//	FABD	//ARM64Op_fabd_Vector,	/* 0x2EA0D400	FA

1	in_use	Opcode	//Opcode	BINARY	OI
788	//	FCMGT	//ARM64Op_fcmgt_register_Vector,	/* 0x2EA0E40	
789	//	FACGT	//ARM64Op_facgt_Vector,	/* 0x2EA0EC00F/	
790	//	FMINP	//ARM64Op_fminp_vector,	/* 0x2EA0F400FM	
791	//	BIT	//ARM64Op_bit,	/* 0x2EA01C00BIT	
792	//	BIF	//ARM64Op_bif,	/* 0x2EE01C00BIF	
793	//	**AdvSIMD three different**	/* AdvSIMD three different */		
794	//	SADDL	//ARM64Op_saddl,	/* 0x0E200000SADD	
795	//	SADDL2	//ARM64Op_saddl2,	/* 0x4E200000SADI	
796	//	SADDW	//ARM64Op_saddw,	/* 0x0E201000SAD	
797	//	SADDW2	//ARM64Op_saddw2,	/* 0x4E201000SAD	
798	//	SSUBL	//ARM64Op_ssubl,	/* 0x0E202000SSUB	
799	//	SSUBL2	//ARM64Op_ssubl2,	/* 0x4E202000SSUE	
800	//	SSUBW	//ARM64Op_ssubw,	/* 0x0E203000SSUI	
801	//	SSUBW2	//ARM64Op_ssubw2,	/* 0x4E203000SSU	
802	//	ADDHN	//ARM64Op_addhn,	/* 0x0E204000ADDI	
803	//	ADDHN2	//ARM64Op_addhn2,	/* 0x4E204000ADD	
804	//	SABAL	//ARM64Op_sabal,	/* 0x0E205000SABA	
805	//	SABAL2	//ARM64Op_sabal2,	/* 0x4E205000SABA	
806	//	SUBHN	//ARM64Op_subhn,	/* 0x0E206000SUBH	
807	//	SUBHN2	//ARM64Op_subhn2,	/* 0x4E206000SUB	
808	//	SABDL	//ARM64Op_sabdl,	/* 0x0E207000SABD	
809	//	SABDL2	//ARM64Op_sabdl2,	/* 0x4E207000SABI	
810	//	SMLAL	//ARM64Op_smlal_vector,	/* 0x0E208000SM	
811	//	SMLAL2	//ARM64Op_smlal2_vector,	/* 0x4E208000SM	
812	//	SQDMLAL	//ARM64Op_sqdmlal_vector_Vector,	/* 0x0E2090(	
813	//	SQDMLAL2	//ARM64Op_sqdmlal2_vector_Vector,	/* 0x4E2090(	
814	//	SMLSL	//ARM64Op_smlsl_vector,	/* 0x0E20A000SM	
815	//	SMLSL2	//ARM64Op_smlsl2_vector,	/* 0x4E20A000SM	
816	//	SQDMLSL	//ARM64Op_sqdmlsl_vector_Vector,	/* 0x0E20B0(	
817	//	SQDMLSL2	//ARM64Op_sqdmlsl2_vector_Vector,	/* 0x4E20B0	
818	//	SMULL	//ARM64Op_smull_vector,	/* 0x0E20C000SM	
819	//	SMULL2	//ARM64Op_smull2_vector,	/* 0x4E20C000SI	
820	//	SQDMULL	//ARM64Op_sqdmull_vector_Vector,	/* 0x0E20D0(	
821	//	SQDMULL2	//ARM64Op_sqdmull2_vector_Vector,	/* 0x4E20D0	

1	in_use	Opcode	//Opcode	BINARY OI
822	//	PMULL	//ARM64Op_pmull,	/* 0x0E20E000PMUL
823	//	PMULL2	//ARM64Op_pmull2,	/* 0x4E20E000PMU
824	//	UADDL	//ARM64Op_uaddl,	/* 0x2E200000UADD
825	//	UADDL2	//ARM64Op_uaddl2,	/* 0x6E200000UADI
826	//	UADDW	//ARM64Op_uaddw,	/* 0x2E201000UAD
827	//	UADDW2	//ARM64Op_uaddw2,	/* 0x6E201000UAD
828	//	USUBL	//ARM64Op_usubl,	/* 0x2E202000USUB
829	//	USUBL2	//ARM64Op_usubl2,	/* 0x6E202000USUI
830	//	USUBW	//ARM64Op_usubw,	/* 0x2E203000USU
831	//	USUBW2	//ARM64Op_usubw2,	/* 0x6E203000USU
832	//	RADDHN	//ARM64Op_raddhn,	/* 0x2E204000RAD
833	//	RADDHN2	//ARM64Op_raddhn2,	/* 0x6E204000RAD
834	//	UABAL	//ARM64Op_uabal,	/* 0x2E205000UABA
835	//	UABAL2	//ARM64Op_uabal2,	/* 0x6E205000UABA
836	//	RSUBHN	//ARM64Op_rsubhn,	/* 0x2E206000RSUI
837	//	RSUBHN2	//ARM64Op_rsubhn2,	/* 0x6E206000RSU
838	//	UABDL	//ARM64Op_uabdl,	/* 0x2E207000UABD
839	//	UABDL2	//ARM64Op_uabdl2,	/* 0x6E207000UABI
840	//	UMLAL	//ARM64Op_umlal_vector,	/* 0x2E208000UM
841	//	UMLAL2	//ARM64Op_umlal2_vector,	/* 0x6E208000UM
842	//	UMLSL	//ARM64Op_umlsl_vector,	/* 0x2E20A000UM
843	//	UMLSL2	//ARM64Op_umlsl2_vector,	/* 0x6E20A000UM
844	//	UMULL	//ARM64Op_umull_vector,	/* 0x2E20C000UM
845	//	UMULL2	//ARM64Op_umull2_vector,	/* 0x6E20C000UI
846	//	**AdvSIMD two-reg misc**	/* AdvSIMD two-reg misc */	
847	//	REV64	//ARM64Op_rev64,	/* 0x0E200800REV6
848	//	REV16	//ARM64Op_rev16_vector,	/* 0x0E201800RE
849	//	SADDLP	//ARM64Op_saddlp,	/* 0x0E202800SADI
850	//	SUQADD	//ARM64Op_suqadd_Vector,	/* 0x0E203800S
851	//	CLS	//ARM64Op_cls_vector,	/* 0x0E204800CLS
852	//	CNT	//ARM64Op_cnt,	/* 0x0E205800CNT
853	//	SADALP	//ARM64Op_sadalp,	/* 0x0E206800SADA
854	//	SQABS	//ARM64Op_sqabs_Vector,	/* 0x0E207800S
855	//	CMGT	//ARM64Op_cmgt_zero_Vector,	/* 0x0E208800

1	in_use	Opcode	//Opcode	BINARY Ol
856	//	CMEQ	//ARM64Op_cmeq_zero_Vector,	/* 0x0E20980(
857	//	CMLT	//ARM64Op_cmlt_zero_Vector,	/* 0x0E20A800
858	//	ABS	//ARM64Op_abs_Vector,	/* 0x0E20B800AB
859	//	XTN	//ARM64Op_xtn,	/* 0x0E212800XTN
860	//	XTN2	//ARM64Op_xtn2,	/* 0x0E212800XTN2
861	//	SQXTN	//ARM64Op_sqxtn_Vector,	/* 0x0E214800SC
862	//	SQXTN2	//ARM64Op_sqxtn2_Vector,	/* 0x0E214800S(
863	//	FCVTN	//ARM64Op_fcvtn,	/* 0x0E216800FCVTN
864	//	FCVTN2	//ARM64Op_fcvtn2,	/* 0x0E216800FCVT
865	//	FCVTL	//ARM64Op_fcvtl,	/* 0x0E217800FCVTL
866	//	FCVTL2	//ARM64Op_fcvtl2,	/* 0x0E217800FCVTL
867	//	FRINTN	//ARM64Op_frintn_vector,	/* 0x0E218800FRI
868	//	FRINTM	//ARM64Op_frintm_vector,	/* 0x0E219800FR
869	//	FCVTNS	//ARM64Op_fcvtns_vector_Vector,	/* 0x0E21A80(
870	//	FCVTMS	//ARM64Op_fcvtms_vector_Vector,	/* 0x0E21B80
871	//	FCVTAS	//ARM64Op_fcvtas_vector_Vector,	/* 0x0E21C80(
872	//	SCVTF	//ARM64Op_scvtf_vector_integer_Vector,	/* 0x0E21D8
873	//	FCMGT	//ARM64Op_fcmgt_zero_Vector,	/* 0x0EA0C80(
874	//	FCMEQ	//ARM64Op_fcmeq_zero_Vector,	/* 0x0EA0D80
875	//	FCMLT	//ARM64Op_fcmlt_zero_Vector,	/* 0x0EA0E800
876	//	FABS	//ARM64Op_fabs_vector,	/* 0x0EA0F800FA
877	//	FRINTP	//ARM64Op_frintp_vector,	/* 0x0EA18800FRI
878	//	FRINTZ	//ARM64Op_frintz_vector,	/* 0x0EA19800FRI
879	//	FCVTPS	//ARM64Op_fcvtps_vector_Vector,	/* 0x0EA1A80(
880	//	FCVTZS	//ARM64Op_fcvtzs_vector_integer_Vector,	/* 0x0EA1B
881	//	URECPE	//ARM64Op_urecpe,	/* 0x0EA1C800URE
882	//	FRECPE	//ARM64Op_frecpe_Vector,	/* 0x0EA1D800F
883	//	REV32	//ARM64Op_rev32_vector,	/* 0x2E200800RE
884	//	UADDLP	//ARM64Op_uaddlp,	/* 0x2E202800UADI
885	//	USQADD	//ARM64Op_usqadd_Vector,	/* 0x2E203800U
886	//	CLZ	//ARM64Op_clz_vector,	/* 0x2E204800CLZ
887	//	UADALP	//ARM64Op_uadalp,	/* 0x2E206800UAD/
888	//	SQNEG	//ARM64Op_sqneg_Vector,	/* 0x2E207800S(
889	//	CMGE	//ARM64Op_cmge_zero_Vector,	/* 0x2E20880(

1	in_use	Opcode	//Opcode	BINARY OI
890	//	CMLE	//ARM64Op_cmle_zero_Vector,	/* 0x2E209800
891	//	NEG	//ARM64Op_neg_vector_Vector,	/* 0x2E20B80(
892	//	SQXTUN	//ARM64Op_sqxtun_Vector,	/* 0x2E212800S(
893	//	SQXTUN2	//ARM64Op_sqxtun2_Vector,	/* 0x2E212800S
894	//	SHLL	//ARM64Op_shll,	/* 0x2E213800SHLL
895	//	SHLL2	//ARM64Op_shll2,	/* 0x2E213800SHLL2
896	//	UQXTN	//ARM64Op_uqxtn_Vector,	/* 0x2E214800U(
897	//	UQXTN2	//ARM64Op_uqxtn2_Vector,	/* 0x2E214800U(
898	//	FCVTXN	//ARM64Op_fcvtxn_Vector,	/* 0x2E216800FC
899	//	FCVTXN2	//ARM64Op_fcvtxn2_Vector,	/* 0x2E216800F(
900	//	FRINTA	//ARM64Op_frinta_vector,	/* 0x2E218800FRI
901	//	FRINTX	//ARM64Op_frintx_vector,	/* 0x2E219800FRI
902	//	FCVTNU	//ARM64Op_fcvtnu_vector_Vector,	/* 0x2E21A80(
903	//	FCVTMU	//ARM64Op_fcvtmu_vector_Vector,	/* 0x2E21B80
904	//	FCVTAU	//ARM64Op_fcvtau_vector_Vector,	/* 0x2E21C80(
905	//	UCVTF	//ARM64Op_ucvtf_vector_integer_Vector,	/* 0x2E21D8
906	//	NOT	//ARM64Op_not,	/* 0x2E205800NOT
907	//	RBIT	//ARM64Op_rbit_vector,	/* 0x2E605800RBIT
908	//	FCMGE	//ARM64Op_fcmge_zero_Vector,	/* 0x2EA0C80
909	//	FCMLE	//ARM64Op_fcmle_zero_Vector,	/* 0x2EA0D80(
910	//	FNEG	//ARM64Op_fneg_vector,	/* 0x2EA0F800FN
911	//	FRINTI	//ARM64Op_frinti_vector,	/* 0x2EA19800FRI
912	//	FCVTPU	//ARM64Op_fcvtpu_vector_Vector,	/* 0x2EA1A80(
913	//	FCVTZU	//ARM64Op_fcvtzu_vector_integer_Vector,	/* 0x2EA1B
914	//	URSQRTE	//ARM64Op_ursqrte,	/* 0x2EA1C800URS(
915	//	FRSQRTE	//ARM64Op_frsqrte_Vector,	/* 0x2EA1D800FF
916	//	FSQRT	//ARM64Op_fsqrt_vector,	/* 0x2EA1F800FS(
917	//	**AdvSIMD across lanes**	/* AdvSIMD across lanes */	
918	//	SADDLV	//ARM64Op_saddlv,	/* 0x0E303800SADI
919	//	SMAXV	//ARM64Op_smaxv,	/* 0x0E30A800SMA
920	//	SMINV	//ARM64Op_sminv,	/* 0x0E31A800SMIN
921	//	ADDV	//ARM64Op_addv,	/* 0x0E31B800ADDV
922	//	UADDLV	//ARM64Op_uaddlv,	/* 0x2E303800UADI
923	//	UMAXV	//ARM64Op_umaxv,	/* 0x2E30A800UMA

1	in_use	Opcode	//Opcode	BINARY OI
924	//	UMINV	//ARM64Op_uminv,	/* 0x2E31A800UMIN
925	//	FMAXNMV	//ARM64Op_fmaxnmv,	/* 0x2E30C800FM
926	//	FMAXV	//ARM64Op_fmaxv,	/* 0x2E30F800FMA)
927	//	FMINNMV	//ARM64Op_fminnmv,	/* 0x2EB0C800FM
928	//	FMINV	//ARM64Op_fminv,	/* 0x2EB0F800FMIN
929	//	**AdvSIMD copy**	/* AdvSIMD copy */	
930	//	DUP	//ARM64Op_dup_element_Vector,	/* 0x0E00040
931	//	DUP	//ARM64Op_dup_general,	/* 0x0E000C00DU
932	//	SMOV	//ARM64Op_smov_32_bit,	/* 0x0E002C00SI
933	//	UMOV	//ARM64Op_umov_32_bit,	/* 0x0E003C00U
934	//	INS	//ARM64Op_ins_general,	/* 0x4E001C00INS
935	//	SMOV	//ARM64Op_smov_64_bit,	/* 0x4E002C00SI
936	//	UMOV	//ARM64Op_umov_64_bit,	/* 0x4E003C00U
937	//	INS	//ARM64Op_ins_element,	/* 0x6E000400INS
938	//	**AdvSIMD vector x indexed element**	/* AdvSIMD vector x indexed element */	
939	//	SMLAL	//ARM64Op_smlal_by_element,	/* 0x0F002000
940	//	SMLAL2	//ARM64Op_smlal2_by_element,	/* 0x0F00200(
941	//	SQDMLAL	//ARM64Op_sqdmlal_by_element_Vector,	/* 0x0F003
942	//	SQDMLAL2	//ARM64Op_sqdmlal2_by_element_Vector,	/* 0x0F00
943	//	SMLSL	//ARM64Op_smlsl_by_element,	/* 0x0F006000
944	//	SMLSL2	//ARM64Op_smlsl2_by_element,	/* 0x0F00600(
945	//	SQDMLSL	//ARM64Op_sqdmlsl_by_element_Vector,	/* 0x0F007
946	//	SQDMLSL2	//ARM64Op_sqdmlsl2_by_element_Vector,	/* 0x0F00
947	//	MUL	//ARM64Op_mul_by_element,	/* 0x0F008000
948	//	SMULL	//ARM64Op_smull_by_element,	/* 0x0F00A000
949	//	SMULL2	//ARM64Op_smull2_by_element,	/* 0x0F00A00(
950	//	SQDMULL	//ARM64Op_sqdmull_by_element_Vector,	/* 0x0F00E
951	//	SQDMULL2	//ARM64Op_sqdmull2_by_element_Vector,	/* 0x0F00
952	//	SQDMULH	//ARM64Op_sqdmulh_by_element_Vector,	/* 0x0F00
953	//	SQRDMULH	//ARM64Op_sqrdmulh_by_element_Vector,	/* 0x0F00
954	//	FMLA	//ARM64Op_fmla_by_element_Vector,	/* 0x0F801(
955	//	FMLS	//ARM64Op_fmls_by_element_Vector,	/* 0x0F8050
956	//	FMUL	//ARM64Op_fmul_by_element_Vector,	/* 0x0F809(
957	//	MLA	//ARM64Op_mla_by_element,	/* 0x2F000000

1	in_use	Opcode	//Opcode	BINARY OI
958	//	UMLAL	//ARM64Op_umlal_by_element,	/* 0x2F002000
959	//	UMLAL2	//ARM64Op_umlal2_by_element,	/* 0x2F00200(
960	//	MLS	//ARM64Op_mls_by_element,	/* 0x2F004000M
961	//	UMLSL	//ARM64Op_umlsl_by_element,	/* 0x2F006000
962	//	UMLSL2	//ARM64Op_umlsl2_by_element,	/* 0x2F00600(
963	//	UMULL	//ARM64Op_umull_by_element,	/* 0x2F00A000
964	//	UMULL2	//ARM64Op_umull2_by_element,	/* 0x2F00A00(
965	//	FMULX	//ARM64Op_fmulx_by_element_Vector,	/* 0x2F809(
966	//	**AdvSIMD modified immediate**	/* AdvSIMD modified immediate */	
967	//	MOVI	//ARM64Op_movi_32_bit_shifted_immediate,	/* 0x0F00
968	//	ORR	//ARM64Op_orr_vector_immediate_32_bit,	/* 0x0F001
969	//	MOVI	//ARM64Op_movi_16_bit_shifted_immediate,	/* 0x0F00
970	//	ORR	//ARM64Op_orr_vector_immediate_16_bit,	/* 0x0F009
971	//	MOVI	//ARM64Op_movi_32_bit_shifting_ones,	/* 0x0F00C4
972	//	MOVI	//ARM64Op_movi_8_bit,	/* 0x0F00E400MO
973	//	FMOV	//ARM64Op_fmov_vector_immediate_Single_precision,	/* 0x(
974	//	MVNI	//ARM64Op_mvni_32_bit_shifted_immediate,	/* 0x2F00
975	//	BIC	//ARM64Op_bic_vector_immediate_32_bit,	/* 0x2F001
976	//	MVNI	//ARM64Op_mvni_16_bit_shifted_immediate,	/* 0x2F00
977	//	BIC	//ARM64Op_bic_vector_immediate_16_bit,	/* 0x2F009
978	//	MVNI	//ARM64Op_mvni_32_bit_shifting_ones,	/* 0x2F00C4
979	//	MOVI	//ARM64Op_movi_64_bit_scalar,	/* 0x2F00E400
980	//	MOVI	//ARM64Op_movi_64_bit_vector,	/* 0x6F00E400
981	//	FMOV	//ARM64Op_fmov_vector_immediate_Double_precision,	/* 0x
982	//	**AdvSIMD shift by immediate**	/* AdvSIMD shift by immediate */	
983	//	SSHR	//ARM64Op_sshr_Vector,	/* 0x0F000400SS
984	//	SSRA	//ARM64Op_ssra_Vector,	/* 0x0F001400SS
985	//	SRSHR	//ARM64Op_srshr_Vector,	/* 0x0F002400SR
986	//	SRSRA	//ARM64Op_srsra_Vector,	/* 0x0F003400SR
987	//	SHL	//ARM64Op_shl_Vector,	/* 0x0F005400SHL
988	//	SQSHL	//ARM64Op_sqshl_immediate_Vector,	/* 0x0F0074
989	//	SHRN	//ARM64Op_shrn,	/* 0x0F008400SHRN
990	//	SHRN2	//ARM64Op_shrn2,	/* 0x0F008400SHRN
991	//	RSHRN	//ARM64Op_rshrn,	/* 0x0F008C00RSHR

1	in_use	Opcode	//Opcode	BINARY	OI
992	//	RSHRN2	//ARM64Op_rshrn2,	/* 0x0F008C00RSHI	
993	//	SQSHRN	//ARM64Op_sqshrn_Vector,	/* 0x0F009400S(	
994	//	SQSHRN2	//ARM64Op_sqshrn2_Vector,	/* 0x0F009400S	
995	//	SQRSHRN	//ARM64Op_sqrshrn_Vector,	/* 0x0F009C00S	
996	//	SQRSHRN2	//ARM64Op_sqrshrn2_Vector,	/* 0x0F009C00S	
997	//	SSHLL	//ARM64Op_sshll,	/* 0x0F00A400SSHLL	
998	//	SSHLL2	//ARM64Op_sshll2,	/* 0x0F00A400SSHL	
999	//	SCVTF	//ARM64Op_scvtf_vector_fixed_point_Vector,	/* 0x0F00E	
1000	//	FCVTZS	//ARM64Op_fcvtzs_vector_fixed_point_Vector,	/* 0x0F00	
1001	//	USHR	//ARM64Op_ushr_Vector,	/* 0x2F000400US	
1002	//	USRA	//ARM64Op_usra_Vector,	/* 0x2F001400US	
1003	//	URSHR	//ARM64Op_urshr_Vector,	/* 0x2F002400UR	
1004	//	URSRA	//ARM64Op_ursra_Vector,	/* 0x2F003400UR	
1005	//	SRI	//ARM64Op_sri_Vector,	/* 0x2F004400SRI	
1006	//	SLI	//ARM64Op_sli_Vector,	/* 0x2F005400SLI	
1007	//	SQSHLU	//ARM64Op_sqshlu_Vector,	/* 0x2F006400S(	
1008	//	UQSHL	//ARM64Op_uqshl_immediate_Vector,	/* 0x2F0074	
1009	//	SQSHRUN	//ARM64Op_sqshrun_Vector,	/* 0x2F008400S	
1010	//	SQSHRUN2	//ARM64Op_sqshrun2_Vector,	/* 0x2F008400S	
1011	//	SQRSHRUN	//ARM64Op_sqrshrun_Vector,	/* 0x2F008C00S	
1012	//	SQRSHRUN2	//ARM64Op_sqrshrun2_Vector,	/* 0x2F008C00	
1013	//	UQSHRN	//ARM64Op_uqshrn_Vector,	/* 0x2F009400U(	
1014	//	UQRSHRN	//ARM64Op_uqrshrn_Vector,	/* 0x2F009C00U	
1015	//	UQRSHRN2	//ARM64Op_uqrshrn2_Vector,	/* 0x2F009C00U	
1016	//	USHLL	//ARM64Op_ushll,	/* 0x2F00A400USHLL	
1017	//	USHLL2	//ARM64Op_ushll2,	/* 0x2F00A400USHL	
1018	//	UCVTF	//ARM64Op_ucvtf_vector_fixed_point_Vector,	/* 0x2F00E	
1019	//	FCVTZU	//ARM64Op_fcvtzu_vector_fixed_point_Vector,	/* 0x2F00	
1020	//	**AdvSIMD TBL/TBX**	/* AdvSIMD TBL/TBX */		
1021	//	TBL	//ARM64Op_tbl_Single_register_table,	/* 0x0E00000(	
1022	//	TBX	//ARM64Op_tbx_Single_register_table,	/* 0x0E00100	
1023	//	TBL	//ARM64Op_tbl_Two_register_table,	/* 0x0E00200(	
1024	//	TBX	//ARM64Op_tbx_Two_register_table,	/* 0x0E00300	
1025	//	TBL	//ARM64Op_tbl_Three_register_table,	/* 0x0E00400	

1	in_use	Opcode	//Opcode	BINARY	Ol
1026	//	TBX	//ARM64Op_tbx_Three_register_table,	/* 0x0E0050(	
1027	//	TBL	//ARM64Op_tbl_Four_register_table,	/* 0x0E00600(	
1028	//	TBX	//ARM64Op_tbx_Four_register_table,	/* 0x0E00700	
1029	//	**AdvSIMD ZIP/UZP/TRN**	/* AdvSIMD ZIP/UZP/TRN */		
1030	//	UZP1	//ARM64Op_uzp1,	/* 0x0E001800UZP1	
1031	//	TRN1	//ARM64Op_trn1,	/* 0x0E002800TRN1	
1032	//	ZIP1	//ARM64Op_zip1,	/* 0x0E003800ZIP1	
1033	//	UZP2	//ARM64Op_uzp2,	/* 0x0E005800UZP2	
1034	//	TRN2	//ARM64Op_trn2,	/* 0x0E006800TRN2	
1035	//	ZIP2	//ARM64Op_zip2,	/* 0x0E007800ZIP2	
1036	//	**AdvSIMD EXT**	/* AdvSIMD EXT */		
1037	//	EXT	//ARM64Op_ext,	/* 0x2E000000EXT	
1038	//	**Loads and stores**	/* Loads and stores */		
1039	//	**AdvSIMD load/store multiple structures**	/* AdvSIMD load/store multiple structures */		
1040	//	ST4	//ARM64Op_st4_multiple_structures_No_offset,	/* 0x0C0(	
1041	//	ST1	//ARM64Op_st1_multiple_structures_Four_registers,	/* 0x0C(	
1042	//	ST3	//ARM64Op_st3_multiple_structures_No_offset,	/* 0x0C0(	
1043	//	ST1	//ARM64Op_st1_multiple_structures_Three_registers,	/* 0x0C	
1044	//	ST1	//ARM64Op_st1_multiple_structures_One_register,	/* 0x0C(	
1045	//	ST2	//ARM64Op_st2_multiple_structures_No_offset,	/* 0x0C0(	
1046	//	ST1	//ARM64Op_st1_multiple_structures_Two_registers,	/* 0x0C	
1047	//	LD4	//ARM64Op_ld4_multiple_structures_No_offset,	/* 0x0C4(	
1048	//	LD1	//ARM64Op_ld1_multiple_structures_Four_registers,	/* 0x0C4	
1049	//	LD3	//ARM64Op_ld3_multiple_structures_No_offset,	/* 0x0C4(	
1050	//	LD1	//ARM64Op_ld1_multiple_structures_Three_registers,	/* 0x0C	
1051	//	LD1	//ARM64Op_ld1_multiple_structures_One_register,	/* 0x0C4	
1052	//	LD2	//ARM64Op_ld2_multiple_structures_No_offset,	/* 0x0C4(	
1053	//	LD1	//ARM64Op_ld1_multiple_structures_Two_registers,	/* 0x0C	
1054	//	**AdvSIMD load/store multiple structures (pc**	/* AdvSIMD load/store multiple structures (post-indexed) */		
1055	//	ST4	//ARM64Op_st4_multiple_structures_Register_offset,	/* 0x0C8	
1056	//	ST1	//ARM64Op_st1_multiple_structures_Four_registers_register_offset, /*		
1057	//	ST3	//ARM64Op_st3_multiple_structures_Register_offset,	/* 0x0C8	
1058	//	ST1	//ARM64Op_st1_multiple_structures_Three_registers_register_offset,/*		
1059	//	ST1	//ARM64Op_st1_multiple_structures_One_register_register_offset, /* (		

1	in_use	Opcode	//Opcode	BINARY	Ol
106(	//	ST2	//ARM64Op_st2_multiple_structures_Register_offset,	/* 0x0C8	
1061	//	ST1	//ARM64Op_st1_multiple_structures_Two_registers_register_offset, /*		
1062	//	ST4	//ARM64Op_st4_multiple_structures_Immediate_offset,	/* 0x0(	
1063	//	ST1	//ARM64Op_st1_multiple_structures_Four_registers_immediate_offset,		
1064	//	ST3	//ARM64Op_st3_multiple_structures_Immediate_offset,	/* 0x0(	
1065	//	ST1	//ARM64Op_st1_multiple_structures_Three_registers_immediate_offse		
1066	//	ST1	//ARM64Op_st1_multiple_structures_One_register_immediate_offset, /		
1067	//	ST2	//ARM64Op_st2_multiple_structures_Immediate_offset,	/* 0x0(	
1068	//	ST1	//ARM64Op_st1_multiple_structures_Two_registers_immediate_offset,		
1069	//	LD4	//ARM64Op_ld4_multiple_structures_Register_offset,	/* 0x0C(	
107(	//	LD1	//ARM64Op_ld1_multiple_structures_Four_registers_register_offset, /*		
1071	//	LD3	//ARM64Op_ld3_multiple_structures_Register_offset,	/* 0x0C(	
1072	//	LD1	//ARM64Op_ld1_multiple_structures_Three_registers_register_offset,/*		
1073	//	LD1	//ARM64Op_ld1_multiple_structures_One_register_register_offset, /* (		
1074	//	LD2	//ARM64Op_ld2_multiple_structures_Register_offset,	/* 0x0C(	
1075	//	LD1	//ARM64Op_ld1_multiple_structures_Two_registers_register_offset, /*		
1076	//	LD4	//ARM64Op_ld4_multiple_structures_Immediate_offset,	/* 0x0(	
1077	//	LD1	//ARM64Op_ld1_multiple_structures_Four_registers_immediate_offset,		
1078	//	LD3	//ARM64Op_ld3_multiple_structures_Immediate_offset,	/* 0x0(	
1079	//	LD1	//ARM64Op_ld1_multiple_structures_Three_registers_immediate_offse		
108(	//	LD1	//ARM64Op_ld1_multiple_structures_One_register_immediate_offset, /		
1081	//	LD2	//ARM64Op_ld2_multiple_structures_Immediate_offset,	/* 0x0(	
1082	//	LD1	//ARM64Op_ld1_multiple_structures_Two_registers_immediate_offset,		
1083	//	**AdvSIMD load/store single structure**	/* AdvSIMD load/store single structure */		
1084	//	ST1	//ARM64Op_st1_single_structure_8_bit,	/* 0x0D0000(	
1085	//	ST3	//ARM64Op_st3_single_structure_8_bit,	/* 0x0D0020(	
1086	//	ST1	//ARM64Op_st1_single_structure_16_bit,	/* 0x0D0040	
1087	//	ST3	//ARM64Op_st3_single_structure_16_bit,	/* 0x0D0060	
1088	//	ST1	//ARM64Op_st1_single_structure_32_bit,	/* 0x0D0080	
1089	//	ST1	//ARM64Op_st1_single_structure_64_bit,	/* 0x0D0084	
109(	//	ST3	//ARM64Op_st3_single_structure_32_bit,	/* 0x0D00A(	
1091	//	ST3	//ARM64Op_st3_single_structure_64_bit,	/* 0x0D00A4	
1092	//	ST2	//ARM64Op_st2_single_structure_8_bit,	/* 0x0D2000(	
1093	//	ST4	//ARM64Op_st4_single_structure_8_bit,	/* 0x0D2020(	

1	in_use	Opcode	//Opcode	BINARY	OI
1094	//	ST2	//ARM64Op_st2_single_structure_16_bit,	/* 0x0D2040	
1095	//	ST4	//ARM64Op_st4_single_structure_16_bit,	/* 0x0D2060	
1096	//	ST2	//ARM64Op_st2_single_structure_32_bit,	/* 0x0D2080	
1097	//	ST2	//ARM64Op_st2_single_structure_64_bit,	/* 0x0D2084	
1098	//	ST4	//ARM64Op_st4_single_structure_32_bit,	/* 0x0D20A0	
1099	//	ST4	//ARM64Op_st4_single_structure_64_bit,	/* 0x0D20A4	
1100	//	LD1	//ARM64Op_ld1_single_structure_8_bit,	/* 0x0D4000(	
1101	//	LD3	//ARM64Op_ld3_single_structure_8_bit,	/* 0x0D4020(	
1102	//	LD1	//ARM64Op_ld1_single_structure_16_bit,	/* 0x0D4040	
1103	//	LD3	//ARM64Op_ld3_single_structure_16_bit,	/* 0x0D4060	
1104	//	LD1	//ARM64Op_ld1_single_structure_32_bit,	/* 0x0D4080	
1105	//	LD1	//ARM64Op_ld1_single_structure_64_bit,	/* 0x0D4084	
1106	//	LD3	//ARM64Op_ld3_single_structure_32_bit,	/* 0x0D40A0	
1107	//	LD3	//ARM64Op_ld3_single_structure_64_bit,	/* 0x0D40A4	
1108	//	LD1R	//ARM64Op_ld1r_No_offset,	/* 0x0D40C000LI	
1109	//	LD3R	//ARM64Op_ld3r_No_offset,	/* 0x0D40E000LI	
1110	//	LD2	//ARM64Op_ld2_single_structure_8_bit,	/* 0x0D6000(	
1111	//	LD4	//ARM64Op_ld4_single_structure_8_bit,	/* 0x0D6020(	
1112	//	LD2	//ARM64Op_ld2_single_structure_16_bit,	/* 0x0D6040	
1113	//	LD4	//ARM64Op_ld4_single_structure_16_bit,	/* 0x0D6060	
1114	//	LD2	//ARM64Op_ld2_single_structure_32_bit,	/* 0x0D6080	
1115	//	LD2	//ARM64Op_ld2_single_structure_64_bit,	/* 0x0D6084	
1116	//	LD4	//ARM64Op_ld4_single_structure_32_bit,	/* 0x0D60A0	
1117	//	LD4	//ARM64Op_ld4_single_structure_64_bit,	/* 0x0D60A4	
1118	//	LD2R	//ARM64Op_ld2r_No_offset,	/* 0x0D60C000LI	
1119	//	LD4R	//ARM64Op_ld4r_No_offset,	/* 0x0D60E000LI	
1120	//	**AdvSIMD load/store single structure (post-**	/* AdvSIMD load/store single structure (post-indexed) */		
1121	//	ST1	//ARM64Op_st1_single_structure_8_bit_register_offset,	/* 0x0D	
1122	//	ST3	//ARM64Op_st3_single_structure_8_bit_register_offset,	/* 0x0D	
1123	//	ST1	//ARM64Op_st1_single_structure_16_bit_register_offset,	/* 0x0I	
1124	//	ST3	//ARM64Op_st3_single_structure_16_bit_register_offset,	/* 0x0I	
1125	//	ST1	//ARM64Op_st1_single_structure_32_bit_register_offset,	/* 0x0I	
1126	//	ST1	//ARM64Op_st1_single_structure_64_bit_register_offset,	/* 0x0I	
1127	//	ST3	//ARM64Op_st3_single_structure_32_bit_register_offset,	/* 0x0I	

1	in_use	Opcode	//Opcode	BINARY	Ol
1128	//	ST3	//ARM64Op_st3_single_structure_64_bit_register_offset,	/* 0x0D	
1129	//	ST1	//ARM64Op_st1_single_structure_8_bit_immediate_offset,	/* 0x(	
1130	//	ST3	//ARM64Op_st3_single_structure_8_bit_immediate_offset,	/* 0x(	
1131	//	ST1	//ARM64Op_st1_single_structure_16_bit_immediate_offset,	/* 0x	
1132	//	ST3	//ARM64Op_st3_single_structure_16_bit_immediate_offset,	/* 0x	
1133	//	ST1	//ARM64Op_st1_single_structure_32_bit_immediate_offset,	/* 0x	
1134	//	ST1	//ARM64Op_st1_single_structure_64_bit_immediate_offset,	/* 0x	
1135	//	ST3	//ARM64Op_st3_single_structure_32_bit_immediate_offset,	/* 0x	
1136	//	ST3	//ARM64Op_st3_single_structure_64_bit_immediate_offset,	/* 0x	
1137	//	ST2	//ARM64Op_st2_single_structure_8_bit_register_offset,	/* 0x0D	
1138	//	ST4	//ARM64Op_st4_single_structure_8_bit_register_offset,	/* 0x0D	
1139	//	ST2	//ARM64Op_st2_single_structure_16_bit_register_offset,	/* 0x0D	
1140	//	ST4	//ARM64Op_st4_single_structure_16_bit_register_offset,	/* 0x0D	
1141	//	ST2	//ARM64Op_st2_single_structure_32_bit_register_offset,	/* 0x0D	
1142	//	ST2	//ARM64Op_st2_single_structure_64_bit_register_offset,	/* 0x0D	
1143	//	ST4	//ARM64Op_st4_single_structure_32_bit_register_offset,	/* 0x0D	
1144	//	ST4	//ARM64Op_st4_single_structure_64_bit_register_offset,	/* 0x0D	
1145	//	ST2	//ARM64Op_st2_single_structure_8_bit_immediate_offset,	/* 0x(	
1146	//	ST4	//ARM64Op_st4_single_structure_8_bit_immediate_offset,	/* 0x(	
1147	//	ST2	//ARM64Op_st2_single_structure_16_bit_immediate_offset,	/* 0x	
1148	//	ST4	//ARM64Op_st4_single_structure_16_bit_immediate_offset,	/* 0x	
1149	//	ST2	//ARM64Op_st2_single_structure_32_bit_immediate_offset,	/* 0x	
1150	//	ST2	//ARM64Op_st2_single_structure_64_bit_immediate_offset,	/* 0x	
1151	//	ST4	//ARM64Op_st4_single_structure_32_bit_immediate_offset,	/* 0x	
1152	//	ST4	//ARM64Op_st4_single_structure_64_bit_immediate_offset,	/* 0x	
1153	//	LD1	//ARM64Op_ld1_single_structure_8_bit_register_offset,	/* 0x0D	
1154	//	LD3	//ARM64Op_ld3_single_structure_8_bit_register_offset,	/* 0x0D	
1155	//	LD1	//ARM64Op_ld1_single_structure_16_bit_register_offset,	/* 0x0D	
1156	//	LD3	//ARM64Op_ld3_single_structure_16_bit_register_offset,	/* 0x0D	
1157	//	LD1	//ARM64Op_ld1_single_structure_32_bit_register_offset,	/* 0x0D	
1158	//	LD1	//ARM64Op_ld1_single_structure_64_bit_register_offset,	/* 0x0D	
1159	//	LD3	//ARM64Op_ld3_single_structure_32_bit_register_offset,	/* 0x0D	
1160	//	LD3	//ARM64Op_ld3_single_structure_64_bit_register_offset,	/* 0x0D	
1161	//	LD1R	//ARM64Op_ld1r_Register_offset,	/* 0x0DC0C00(	

1	in_use	Opcode	//Opcode	BINARY	OI
1162	//	LD3R	//ARM64Op_ld3r_Register_offset,	/* 0x0DC0E000	
1163	//	LD1	//ARM64Op_ld1_single_structure_8_bit_immediate_offset,	/* 0x(	
1164	//	LD3	//ARM64Op_ld3_single_structure_8_bit_immediate_offset,	/* 0x(	
1165	//	LD1	//ARM64Op_ld1_single_structure_16_bit_immediate_offset,	/* 0x	
1166	//	LD3	//ARM64Op_ld3_single_structure_16_bit_immediate_offset,	/* 0x	
1167	//	LD1	//ARM64Op_ld1_single_structure_32_bit_immediate_offset,	/* 0x	
1168	//	LD1	//ARM64Op_ld1_single_structure_64_bit_immediate_offset,	/* 0x	
1169	//	LD3	//ARM64Op_ld3_single_structure_32_bit_immediate_offset,	/* 0x	
1170	//	LD3	//ARM64Op_ld3_single_structure_64_bit_immediate_offset,	/* 0x	
1171	//	LD1R	//ARM64Op_ld1r_Immediate_offset,	/* 0x0DDFC0(	
1172	//	LD3R	//ARM64Op_ld3r_Immediate_offset,	/* 0x0DDFE0(	
1173	//	LD2	//ARM64Op_ld2_single_structure_8_bit_register_offset,	/* 0x0D	
1174	//	LD4	//ARM64Op_ld4_single_structure_8_bit_register_offset,	/* 0x0D	
1175	//	LD2	//ARM64Op_ld2_single_structure_16_bit_register_offset,	/* 0x0[	
1176	//	LD4	//ARM64Op_ld4_single_structure_16_bit_register_offset,	/* 0x0[	
1177	//	LD2	//ARM64Op_ld2_single_structure_32_bit_register_offset,	/* 0x0[	
1178	//	LD2	//ARM64Op_ld2_single_structure_64_bit_register_offset,	/* 0x0[	
1179	//	LD4	//ARM64Op_ld4_single_structure_32_bit_register_offset,	/* 0x0[	
1180	//	LD4	//ARM64Op_ld4_single_structure_64_bit_register_offset,	/* 0x0[	
1181	//	LD2R	//ARM64Op_ld2r_Register_offset,	/* 0x0DE0C000	
1182	//	LD4R	//ARM64Op_ld4r_Register_offset,	/* 0x0DE0E000	
1183	//	LD2	//ARM64Op_ld2_single_structure_8_bit_immediate_offset,	/* 0x(	
1184	//	LD4	//ARM64Op_ld4_single_structure_8_bit_immediate_offset,	/* 0x(	
1185	//	LD2	//ARM64Op_ld2_single_structure_16_bit_immediate_offset,	/* 0x	
1186	//	LD4	//ARM64Op_ld4_single_structure_16_bit_immediate_offset,	/* 0x	
1187	//	LD2	//ARM64Op_ld2_single_structure_32_bit_immediate_offset,	/* 0x	
1188	//	LD2	//ARM64Op_ld2_single_structure_64_bit_immediate_offset,	/* 0x	
1189	//	LD4	//ARM64Op_ld4_single_structure_32_bit_immediate_offset,	/* 0x	
1190	//	LD4	//ARM64Op_ld4_single_structure_64_bit_immediate_offset,	/* 0x	
1191	//	LD2R	//ARM64Op_ld2r_Immediate_offset,	/* 0x0DFFC0(	
1192	//	LD4R	//ARM64Op_ld4r_Immediate_offset,	/* 0x0DFFE0(	

	in_use	Opcode	//BINARY	Opcode	Opcodecomments
2		**UNALLOCATED**	/* UNALLOCATED */		
3		BAD	0x00000000,/* BAD		badinvalid operation */
4		**Branch,exception generation and syst**	/* Branch,exception generation and system Instruction */		
5		**Compare _ Branch (immediate)**	/* Compare _ Branch (immediate) */		
6		CBZ	0x34000000,/* CBZ	cbz_W	*/
7		CBNZ	0x35000000,/* CBNZ	cbnz_W	*/
8		CBZ	0xB4000000,/* CBZ	cbz_X	*/
9		CBNZ	0xB5000000,/* CBNZ	cbnz_X	*/
10		**Test bit & branch (immediate)**	/* Test bit & branch (immediate) */		
11		TBZ	0x36000000,/* TBZ	tbz	*/
12		TBNZ	0x37000000,/* TBNZ	tbnz	*/
13		**Conditional branch (immediate)**	/* 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	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	msr_imm	*/
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 (register)**	/* 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 (immediate)**	/* 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 stxr_W */
59		STLXR	0x88008000,/* STLXR stlxr_W */
60		STXP	0x88200000,/* STXP stxp_W */
61		STLXP	0x88208000,/* STLXP stlxp_W */
62		LDXR	0x88400000,/* LDXR ldxr_W */
63		LDAXR	0x88408000,/* LDAXR ldaxr_W */
64		LDXP	0x88600000,/* LDXP ldxp_W */
65		LDAXP	0x88608000,/* LDAXP ldaxp_W */
66		STLR	0x88808000,/* STLR stlr_W */
67		LDAR	0x88C08000,/* LDAR ldar_W */
68		STXR	0xC8000000,/* STXR stxr_X */
69		STLXR	0xC8008000,/* STLXR stlxr_X */
70		STXP	0xC8200000,/* STXP stxp_X */
71		STLXP	0xC8208000,/* STLXP stlxp_X */
72		LDXR	0xC8400000,/* LDXR ldxr_X */
73		LDAXR	0xC8408000,/* LDAXR ldaxr_X */
74		LDXP	0xC8600000,/* LDXP ldxp_X */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
75		LDAXP	0xC8608000,/* LDAXP	ldaxp_X */	
76		STLR	0xC8808000,/* STLR	stlr_X */	
77		LDAR	0xC8C08000,/* LDAR	ldar_X */	
78		**Load register (literal)**	/* Load register (literal) */		
79		LDR	0x18000000,/* LDR	ldr_W */	
80		LDR	0x1C000000,/* LDR	ldr_S */	
81		LDR	0x58000000,/* LDR	ldr_X */	
82		LDR	0x5C000000,/* LDR	ldr_D */	
83		LDRSW	0x98000000,/* LDRSW	ldrsw */	
84		LDR	0x9C000000,/* LDR	ldr_Q */	
85		PRFM	0xD8000000,/* PRFM	prfm */	
86		**Load/store no-allocate pair (offset)**	/* Load/store no-allocate pair (offset) */		
87		STNP	0x28000000,/* STNP	stnp_W */	
88		LDNP	0x28400000,/* LDNP	ldnp_W */	
89		STNP	0x2C000000,/* STNP	stnp_S */	
90		LDNP	0x2C400000,/* LDNP	ldnp_S */	
91		STNP	0x6C000000,/* STNP	stnp_D */	
92		LDNP	0x6C400000,/* LDNP	ldnp_D */	
93		STNP	0xA8000000,/* STNP	stnp_X */	
94		LDNP	0xA8400000,/* LDNP	ldnp_X */	
95		STNP	0xAC000000,/* STNP	stnp_Q */	
96		LDNP	0xAC400000,/* LDNP	ldnp_Q */	
97		**Load/store register pair (post-indexed)**	/* Load/store register pair (post-indexed) */		
98		STP	0x28800000,/* STP	stp_post_W */	
99		LDP	0x28C00000,/* LDP	ldp_post_W */	
100		STP	0x2C800000,/* STP	stp_post_S */	
101		LDP	0x2CC00000,/* LDP	ldp_post_S */	
102		LDPSW	0x68C00000,/* LDPSW	ldpsw_post */	
103		STP	0x6C800000,/* STP	stp_post_D */	
104		LDP	0x6CC00000,/* LDP	ldp_post_D */	
105		STP	0xA8800000,/* STP	stp_post_X */	
106		LDP	0xA8C00000,/* LDP	ldp_post_X */	
107		STP	0xAC800000,/* STP	stp_post_Q */	
108		LDP	0xACC00000,/* LDP	ldp_post_Q */	
109		**Load/store register pair (offset)**	/* Load/store register pair (offset) */		

1	in_use	Opcode	//BINARY Opcode Opcodecomments
110		STP	0x29000000,/* STP stp_off_W */
111		LDP	0x29400000,/* LDP ldp_off_W */
112		STP	0x2D000000,/* STP stp_off_S */
113		LDP	0x2D400000,/* LDP ldp_off_S */
114		LDPSW	0x69400000,/* LDPSW ldpsw_off */
115		STP	0x6D000000,/* STP stp_off_D */
116		LDP	0x6D400000,/* LDP ldp_off_D */
117		STP	0xA9000000,/* STP stp_off_X */
118		LDP	0xA9400000,/* LDP ldp_off_X */
119		STP	0xAD000000,/* STP stp_off_Q */
120		LDP	0xAD400000,/* LDP ldp_off_Q */
121		**Load/store register pair (pre-indexed)**	/* Load/store register pair (pre-indexed) */
122		STP	0x29800000,/* STP stp_pre_W */
123		LDP	0x29C00000,/* LDP ldp_pre_W */
124		STP	0x2D800000,/* STP stp_pre_S */
125		LDP	0x2DC00000,/* LDP ldp_pre_S */
126		LDPSW	0x69C00000,/* LDPSW ldpsw_pre */
127		STP	0x6D800000,/* STP stp_pre_D */
128		LDP	0x6DC00000,/* LDP ldp_pre_D */
129		STP	0xA9800000,/* STP stp_pre_X */
130		LDP	0xA9C00000,/* LDP ldp_pre_X */
131		STP	0xAD800000,/* STP stp_pre_Q */
132		LDP	0xADC00000,/* LDP ldp_pre_Q */
133		**Load/store register (unscaled immediate)**	/* Load/store register (unscaled immediate) */
134		STURB	0x38000000,/* STURB sturb */
135		LDURB	0x38400000,/* LDURB ldurb */
136		LDURSB	0x38800000,/* LDURSB ldursb_X */
137		LDURSB	0x38C00000,/* LDURSB ldursb_W */
138		STUR	0x3C000000,/* STUR stur_B */
139		LDUR	0x3C400000,/* LDUR ldur_B */
140		STUR	0x3C800000,/* STUR stur_Q */
141		LDUR	0x3CC00000,/* LDUR ldur_Q */
142		STURH	0x78000000,/* STURH sturh */
143		LDURH	0x78400000,/* LDURH ldurh */
144		LDURSH	0x78800000,/* LDURSH ldursh_X */
145		LDURSH	0x78C00000,/* LDURSH ldursh_W */
146		STUR	0x7C000000,/* STUR stur_H */
147		LDUR	0x7C400000,/* LDUR ldur_H */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
148		STUR	0xB8000000,/* STUR	stur_W */	
149		LDUR	0xB8400000,/* LDUR	ldur_W */	
150		LDURSW	0xB8800000,/* LDURSW	ldursw */	
151		STUR	0xBC000000,/* STUR	stur_S */	
152		LDUR	0xBC400000,/* LDUR	ldur_S */	
153		STUR	0xF8000000,/* STUR	stur_X */	
154		LDUR	0xF8400000,/* LDUR	ldur_X */	
155		PRFUM	0xF8800000,/* PRFUM	prfum */	
156		STUR	0xFC000000,/* STUR	stur_D */	
157		LDUR	0xFC400000,/* LDUR	ldur_D */	
158		**Load/store register (immediate post-indexe**	/* Load/store register (immediate post-indexed) */		
159		STRB	0x38000400,/* STRB	strb_post */	
160		LDRB	0x38400400,/* LDRB	ldrb_post */	
161		LDRSB	0x38800400,/* LDRSB	ldrsb_post_X */	
162		LDRSB	0x38C00400,/* LDRSB	ldrsb_post_W */	
163		STR	0x3C000400,/* STR	str_post_B */	
164		LDR	0x3C400400,/* LDR	ldr_post_B */	
165		STR	0x3C800400,/* STR	str_post_Q */	
166		LDR	0x3CC00400,/* LDR	ldr_post_Q */	
167		STRH	0x78000400,/* STRH	strh_post */	
168		LDRH	0x78400400,/* LDRH	ldrh_post */	
169		LDRSH	0x78800400,/* LDRSH	ldrsh_post_X */	
170		LDRSH	0x78C00400,/* LDRSH	ldrsh_post_W */	
171		STR	0x7C000400,/* STR	str_post_H */	
172		LDR	0x7C400400,/* LDR	ldr_post_H */	
173		STR	0xB8000400,/* STR	str_post_W */	
174		LDR	0xB8400400,/* LDR	ldr_post_W */	
175		LDRSW	0xB8800400,/* LDRSW	ldrsw_post */	
176		STR	0xBC000400,/* STR	str_post_S */	
177		LDR	0xBC400400,/* LDR	ldr_post_S */	
178		STR	0xF8000400,/* STR	str_post_X */	
179		LDR	0xF8400400,/* LDR	ldr_post_X */	
180		STR	0xFC000400,/* STR	str_post_D */	
181		LDR	0xFC400400,/* LDR	ldr_post_D */	
182		**Load/store register (unprivileged)**	/* Load/store register (unprivileged) */		
183		STTRB	0x38000800,/* STTRB	sttrb */	
184		LDTRB	0x38400800,/* LDTRB	ldtrb */	
185		LDTRSB	0x38800800,/* LDTRSB	ldtrsb_X */	

1	in_use	Opcode	//BINARY Opcode Opcodecomments
186		LDTRSB	0x38C00800,/* LDTRSB ldtrsb_W */
187		STTRH	0x78000800,/* STTRH sttrh */
188		LDTRH	0x78400800,/* LDTRH ldtrh */
189		LDTRSH	0x78800800,/* LDTRSH ldtrsh_X */
190		LDTRSH	0x78C00800,/* LDTRSH ldtrsh_W */
191		STTR	0xB8000800,/* STTR sttr_W */
192		LDTR	0xB8400800,/* LDTR ldtr_W */
193		LDTRSW	0xB8800800,/* LDTRSW ldtrsw */
194		STTR	0xF8000800,/* STTR sttr_X */
195		LDTR	0xF8400800,/* LDTR ldtr_X */
196		**Load/store register (immediate pre-indexed**	/* Load/store register (immediate pre-indexed) */
197		STRB	0x38000C00,/* STRB strb_pre */
198		LDRB	0x38400C00,/* LDRB ldrb_pre */
199		LDRSB	0x38800C00,/* LDRSB ldrsb_pre_X */
200		LDRSB	0x38C00C00,/* LDRSB ldrsb_pre_W */
201		STR	0x3C000C00,/* STR str_pre_B */
202		LDR	0x3C400C00,/* LDR ldr_pre_B */
203		STR	0x3C800C00,/* STR str_pre_Q */
204		LDR	0x3CC00C00,/* LDR ldr_pre_Q */
205		STRH	0x78000C00,/* STRH strh_pre */
206		LDRH	0x78400C00,/* LDRH ldrh_pre */
207		LDRSH	0x78800C00,/* LDRSH ldrsh_pre_X */
208		LDRSH	0x78C00C00,/* LDRSH ldrsh_pre_W */
209		STR	0x7C000C00,/* STR str_pre_H */
210		LDR	0x7C400C00,/* LDR ldr_pre_H */
211		STR	0xB8000C00,/* STR str_pre_W */
212		LDR	0xB8400C00,/* LDR ldr_pre_W */
213		LDRSW	0xB8800C00,/* LDRSW ldrsw_pre */
214		STR	0xBC000C00,/* STR str_pre_S */
215		LDR	0xBC400C00,/* LDR ldr_pre_S */
216		STR	0xF8000C00,/* STR str_pre_X */
217		LDR	0xF8400C00,/* LDR ldr_pre_X */
218		STR	0xFC000C00,/* STR str_pre_D */
219		LDR	0xFC400C00,/* LDR ldr_pre_D */
220		**Load/store register (register offset)**	/* Load/store register (register offset) */
221		STRB	0x38200800,/* STRB strb_off */
222		LDRB	0x38600800,/* LDRB ldrb_off */
223		LDRSB	0x38A00800,/* LDRSB ldrsb_off_X */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
224		LDRSB	0x38E00800,/* LDRSB	ldrsb_off_W */	
225		STR	0x3C200800,/* STR	str_off_B */	
226		LDR	0x3C600800,/* LDR	ldr_off_B */	
227		STR	0x3CA00800,/* STR	str_off_Q */	
228		LDR	0x3CE00800,/* LDR	ldr_off_Q */	
229		STRH	0x78200800,/* STRH	strh_off */	
230		LDRH	0x78600800,/* LDRH	ldrh_off */	
231		LDRSH	0x78A00800,/* LDRSH	ldrsh_off_X */	
232		LDRSH	0x78E00800,/* LDRSH	ldrsh_off_W */	
233		STR	0x7C200800,/* STR	str_off_H */	
234		LDR	0x7C600800,/* LDR	ldr_off_H */	
235		STR	0xB8200800,/* STR	str_off_W */	
236		LDR	0xB8600800,/* LDR	ldr_off_W */	
237		LDRSW	0xB8A00800,/* LDRSW	ldrsw_off */	
238		STR	0xBC200800,/* STR	str_off_S */	
239		LDR	0xBC600800,/* LDR	ldr_off_S */	
240		STR	0xF8200800,/* STR	str_off_D */	
241		LDR	0xF8600800,/* LDR	ldr_off_D */	
243		STR	0xFC200800,/* STR	str_off_D */	
244		LDR	0xFC600800,/* LDR	ldr_off_D */	
242		PRFM	0xF8A00800,/* PRFM	prfm_off */	
245		**Load/store register (unsigned immediate)**	/* Load/store register (unsigned immediate) */		
246		STRB	0x39000000,/* STRB	strb_imm */	
247		LDRB	0x39400000,/* LDRB	ldrb_imm */	
248		LDRSB	0x39800000,/* LDRSB	ldrsb_imm_X */	
249		LDRSB	0x39C00000,/* LDRSB	ldrsb_imm_W */	
250		STR	0x3D000000,/* STR	str_imm_B */	
251		LDR	0x3D400000,/* LDR	ldr_imm_B */	
252		STR	0x3D800000,/* STR	str_imm_Q */	
253		LDR	0x3DC00000,/* LDR	ldr_imm_Q */	
254		STRH	0x79000000,/* STRH	strh_imm */	
255		LDRH	0x79400000,/* LDRH	ldrh_imm */	
256		LDRSH	0x79800000,/* LDRSH	ldrsh_imm_X */	
257		LDRSH	0x79C00000,/* LDRSH	ldrsh_imm_W */	
258		STR	0x7D000000,/* STR	str_imm_H */	
259		LDR	0x7D400000,/* LDR	ldr_imm_H */	
260		STR	0xB9000000,/* STR	str_imm_W */	
261		LDR	0xB9400000,/* LDR	ldr_imm_W */	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
262		LDRSW	0xB9800000,/* LDRSW	ldrsw_imm */	
263		STR	0xBD000000,/* STR	str_imm_S */	
264		LDR	0xBD400000,/* LDR	ldr_imm_S */	
265		STR	0xF9000000,/* STR	str_imm_X */	
266		LDR	0xF9400000,/* LDR	ldr_imm_X */	
268		STR	0xFD000000,/* STR	str_imm_D */	
269		LDR	0xFD400000,/* LDR	ldr_imm_D */	
267		PRFM	0xF9800000,/* PRFM	prfm_imm */	
270	**Data processing – Immediate**		/* 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	add_imm_W */	
276		ADDS	0x31000000,/* ADDS	adds_imm_W */	
277		SUB	0x51000000,/* SUB	sub_imm_W */	
278		SUBS	0x71000000,/* SUBS	subs_imm_W */	
279		ADD	0x91000000,/* ADD	add_imm_X */	
280		ADDS	0xB1000000,/* ADDS	adds_imm_X */	
281		SUB	0xD1000000,/* SUB	sub_imm_X */	
282		SUBS	0xF1000000,/* SUBS	subs_imm_X */	
283	**Logical (immediate)**		/* Logical (immediate) */		
284		AND	0x12000000,/* AND	and_imm_W */	
285		ORR	0x32000000,/* ORR	orr_imm_W */	
286		EOR	0x52000000,/* EOR	eor_imm_W */	
287		ANDS	0x72000000,/* ANDS	ands_imm_W */	
288		AND	0x92000000,/* AND	and_imm_X */	
289		ORR	0xB2000000,/* ORR	orr_imm_X */	
290		EOR	0xD2000000,/* EOR	eor_imm_X */	
291		ANDS	0xF2000000,/* ANDS	ands_imm_X */	
292	**Move wide (immediate)**		/* Move wide (immediate) */		
293		MOVN	0x12800000,/* MOVN	movn_W */	
294		MOVZ	0x52800000,/* MOVZ	movz_W */	
295		MOVK	0x72800000,/* MOVK	movk_W */	
296		MOVN	0x92800000,/* MOVN	movn_X */	
297		MOVZ	0xD2800000,/* MOVZ	movz_X */	
298		MOVK	0xF2800000,/* MOVK	movk_X */	
299	**Bitfield**		/* Bitfield */		

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
300		SBFM	0x13000000,/* SBFM		sbfm_W */
301		BFM	0x33000000,/* BFM		bfm_W */
302		UBFM	0x53000000,/* UBFM		ubfm_W */
303		SBFM	0x93400000,/* SBFM		sbfm_X */
304		BFM	0xB3400000,/* BFM		bfm_X */
305		UBFM	0xD3400000,/* UBFM		ubfm_X */
306		**Extract**	/* Extract */		
307		EXTR	0x13800000,/* EXTR		extr_W */
308		EXTR	0x93C00000,/* EXTR		extr_X */
309		**Data Processing – register**	/* Data Processing – register */		
310		**Logical (shifted register)**	/* Logical (shifted register) */		
311		AND	0x0A000000,/* AND		and_W */
312		BIC	0x0A200000,/* BIC		bic_W */
313		ORR	0x2A000000,/* ORR		orr_W */
314		ORN	0x2A200000,/* ORN		orn_W */
315		EOR	0x4A000000,/* EOR		eor_W */
316		EON	0x4A200000,/* EON		eon_W */
317		ANDS	0x6A000000,/* ANDS		ands_W */
318		BICS	0x6A200000,/* BICS		bics_W */
319		AND	0x8A000000,/* AND		and_X */
320		BIC	0x8A200000,/* BIC		bic_X */
321		ORR	0xAA000000,/* ORR		orr_X */
322		ORN	0xAA200000,/* ORN		orn_X */
323		EOR	0xCA000000,/* EOR		eor_X */
324		EON	0xCA200000,/* EON		eon_X */
325		ANDS	0xEA000000,/* ANDS		ands_X */
326		BICS	0xEA200000,/* BICS		bics_X */
327		**Add/subtract (shifted register)**	/* Add/subtract (shifted register) */		
328		ADD	0x0B000000,/* ADD		add_W */
329		ADDS	0x2B000000,/* ADDS		adds_W */
330		SUB	0x4B000000,/* SUB		sub_W */
331		SUBS	0x6B000000,/* SUBS		subs_W */
332		ADD	0x8B000000,/* ADD		add_X */
333		ADDS	0xAB000000,/* ADDS		adds_X */
334		SUB	0xCB000000,/* SUB		sub_X */
335		SUBS	0xEB000000,/* SUBS		subs_X */
336		**Add/subtract (extended register)**	/* Add/subtract (extended register) */		
337		ADD	0x0B200000,/* ADD		add_ext_W */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
338		ADDS	0x2B200000,/* ADDS	adds_ext_W */	
339		SUB	0x4B200000,/* SUB	sub_ext_W */	
340		SUBS	0x6B200000,/* SUBS	subs_ext_W */	
341		ADD	0x8B200000,/* ADD	add_ext_X */	
342		ADDS	0xAB200000,/* ADDS	adds_ext_X */	
343		SUB	0xCB200000,/* SUB	sub_ext_X */	
344		SUBS	0xEB200000,/* SUBS	subs_ext_X */	
345		**Add/subtract (with carry)**	/* Add/subtract (with carry) */		
346		ADC	0x1A000000,/* ADC	adc_W */	
347		ADCS	0x3A000000,/* ADCS	adcs_W */	
348		SBC	0x5A000000,/* SBC	sbc_W */	
349		SBCS	0x7A000000,/* SBCS	sbcs_W */	
350		ADC	0x9A000000,/* ADC	adc_X */	
351		ADCS	0xBA000000,/* ADCS	adcs_X */	
352		SBC	0xDA000000,/* SBC	sbc_X */	
353		SBCS	0xFA000000,/* SBCS	sbcs_X */	
354		**Conditional compare (register)**	/* Conditional compare (register) */		
355		CCMN	0x3A400000,/* CCMN	ccmn_W */	
356		CCMN	0xBA400000,/* CCMN	ccmn_X */	
357		CCMP	0x7A400000,/* CCMP	ccmp_W */	
358		CCMP	0xFA400000,/* CCMP	ccmp_X */	
359		**Conditional compare (immediate)**	/* Conditional compare (immediate) */		
360		CCMN	0x3A400800,/* CCMN	ccmn_imm_W */	
361		CCMN	0xBA400800,/* CCMN	ccmn_imm_X */	
362		CCMP	0x7A400800,/* CCMP	ccmp_imm_W */	
363		CCMP	0xFA400800,/* CCMP	ccmp_imm_X */	
364		**Conditional select**	/* Conditional select */		
365		CSEL	0x1A800000,/* CSEL	csel_W */	
366		CSINC	0x1A800400,/* CSINC	csinc_W */	
367		CSINV	0x5A800000,/* CSINV	csinv_W */	
368		CSNEG	0x5A800400,/* CSNEG	csneg_W */	
369		CSEL	0x9A800000,/* CSEL	csel_X */	
370		CSINC	0x9A800400,/* CSINC	csinc_X */	
371		CSINV	0xDA800000,/* CSINV	csinv_X */	
372		CSNEG	0xDA800400,/* CSNEG	csneg_X */	
373		**Data-processing (3 source)**	/* Data-processing (3 source) */		
374		MADD	0x1B000000,/* MADD	madd_W */	
375		MADD	0x9B000000,/* MADD	madd_X */	

1	in_use	Opcode	//BINARY Opcode Opcodecomments
376		SMADDL	0x9B200000,/* SMADDL smaddl */
377		UMADDL	0x9BA00000,/* UMADDL umaddl */
378		MSUB	0x1B008000,/* MSUB msub_W */
379		MSUB	0x9B008000,/* MSUB msub_X */
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 udiv_W */
394		UDIV	0x9AC00800,/* UDIV udiv_X */
395		SDIV	0x1AC00C00,/* SDIV sdiv_W */
396		SDIV	0x9AC00C00,/* SDIV sdiv_X */
397		LSLV	0x1AC02000,/* LSLV lslv_W */
398		LSLV	0x9AC02000,/* LSLV lslv_X */
399		LSRV	0x1AC02400,/* LSRV lsrv_W */
400		LSRV	0x9AC02400,/* LSRV lsrv_X */
401		ASRV	0x1AC02800,/* ASRV asrv_W */
402		ASRV	0x9AC02800,/* ASRV asrv_X */
403		RORV	0x1AC02C00,/* RORV rorv_W */
404		RORV	0x9AC02C00,/* RORV rorv_X */
405		**Data-processing (1 source)**	/* Data-processing (1 source) */
406		RBIT	0x5AC00000,/* RBIT rbit_W */
407		RBIT	0xDAC00000,/* RBIT rbit_X */
408		CLZ	0x5AC01000,/* CLZ clz_W */
409		CLZ	0xDAC01000,/* CLZ clz_X */
410		CLS	0x5AC01400,/* CLS cls_W */
411		CLS	0xDAC01400,/* CLS cls_X */
412		REV	0x5AC00800,/* REV rev_W */
413		REV	0xDAC00C00,/* REV rev_X */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
414		REV16	0xDAC00400,/* REV16 rev16_W */
415		REV16	0x5AC00400,/* REV16 rev16_X */
416		REV32	0xDAC00800,/* REV32 rev32 */
417	//	**Data Processing – SIMD and floating p**	/* Data Processing – SIMD and floating point */
418	//	**Floating-point<->fixed-point conversions**	/* Floating-point<->fixed-point conversions */
419	//	SCVTF	//0x1E020000,/* SCVTF ARM64Op_scvtf_scalar_fixed_point_32_bit_to_single_
420	//	UCVTF	//0x1E030000,/* UCVTF ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_single_
421	//	FCVTZS	//0x1ED80000,/* FCVTZS ARM64Op_fcvtzs_scalar_fixed_point_Single_precisio
422	//	FCVTZU	//0x1ED90000,/* FCVTZU ARM64Op_fcvtzu_scalar_fixed_point_Single_precisio
423	//	SCVTF	//0x1E020000,/* SCVTF ARM64Op_scvtf_scalar_fixed_point_32_bit_to_double
424	//	UCVTF	//0x1E030000,/* UCVTF ARM64Op_ucvtf_scalar_fixed_point_32_bit_to_double
425	//	FCVTZS	//0x1ED80000,/* FCVTZS ARM64Op_fcvtzs_scalar_fixed_point_Double_precisi
426	//	FCVTZU	//0x1ED90000,/* FCVTZU ARM64Op_fcvtzu_scalar_fixed_point_Double_precis
427	//	SCVTF	//0x9E020000,/* SCVTF ARM64Op_scvtf_scalar_fixed_point_64_bit_to_single_
428	//	UCVTF	//0x9E030000,/* UCVTF ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_single_
429	//	FCVTZS	//0x9ED80000,/* FCVTZS ARM64Op_fcvtzs_scalar_fixed_point_Single_precisio
430	//	FCVTZU	//0x9ED90000,/* FCVTZU ARM64Op_fcvtzu_scalar_fixed_point_Single_precisio
431	//	SCVTF	//0x9E020000,/* SCVTF ARM64Op_scvtf_scalar_fixed_point_64_bit_to_double
432	//	UCVTF	//0x9E030000,/* UCVTF ARM64Op_ucvtf_scalar_fixed_point_64_bit_to_double
433	//	FCVTZS	//0x9ED80000,/* FCVTZS ARM64Op_fcvtzs_scalar_fixed_point_Double_precisi
434	//	FCVTZU	//0x9ED90000,/* FCVTZU ARM64Op_fcvtzu_scalar_fixed_point_Double_precis
435	//	**Floating-point conditional compare**	/* Floating-point conditional compare */
436	//	FCCMP	//0x1E200400,/* FCCMP ARM64Op_fccmp_Single_precision */
437	//	FCCMPE	//0x1E200410,/* FCCMPE ARM64Op_fccmpe_Single_precision */
438	//	FCCMP	//0x1E600400,/* FCCMP ARM64Op_fccmp_Double_precision */
439	//	FCCMPE	//0x1E600410,/* FCCMPE ARM64Op_fccmpe_Double_precision */
440	//	**Floating-point data-processing (2 source)**	/* Floating-point data-processing (2 source) */
441	//	FMUL	//0x1E200800,/* FMUL ARM64Op_fmul_scalar_Single_precision */
442	//	FDIV	//0x1E201800,/* FDIV ARM64Op_fdiv_scalar_Single_precision */
443	//	FADD	//0x1E202800,/* FADD ARM64Op_fadd_scalar_Single_precision */
444	//	FSUB	//0x1E203800,/* FSUB ARM64Op_fsub_scalar_Single_precision */
445	//	FMAX	//0x1E204800,/* FMAX ARM64Op_fmax_scalar_Single_precision */
446	//	FMIN	//0x1E205800,/* FMIN ARM64Op_fmin_scalar_Single_precision */
447	//	FMAXNM	//0x1E206800,/* FMAXNM ARM64Op_fmaxnm_scalar_Single_precision */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
448	//	FMINNM	//0x1E207800,/*	FMINNM	ARM64Op_fminnm_scalar_Single_precision */
449	//	FNMUL	//0x1E208800,/*	FNMUL	ARM64Op_fnmul_Single_precision */
450	//	FMUL	//0x1E600800,/*	FMUL	ARM64Op_fmul_scalar_Double_precision */
451	//	FDIV	//0x1E601800,/*	FDIV	ARM64Op_fdiv_scalar_Double_precision */
452	//	FADD	//0x1E602800,/*	FADD	ARM64Op_fadd_scalar_Double_precision */
453	//	FSUB	//0x1E603800,/*	FSUB	ARM64Op_fsub_scalar_Double_precision */
454	//	FMAX	//0x1E604800,/*	FMAX	ARM64Op_fmax_scalar_Double_precision */
455	//	FMIN	//0x1E605800,/*	FMIN	ARM64Op_fmin_scalar_Double_precision */
456	//	FMAXNM	//0x1E606800,/*	FMAXNM	ARM64Op_fmaxnm_scalar_Double_precision */
457	//	FMINNM	//0x1E607800,/*	FMINNM	ARM64Op_fminnm_scalar_Double_precision */
458	//	FNMUL	//0x1E608800,/*	FNMUL	ARM64Op_fnmul_Double_precision */
459	//	**Floating-point conditional select**	/* Floating-point conditional select */		
460	//	FCSEL	//0x1E200C00,/*	FCSEL	ARM64Op_fcsel_Single_precision */
461	//	FCSEL	//0x1E600C00,/*	FCSEL	ARM64Op_fcsel_Double_precision */
462	//	**Floating-point immediate**	/* Floating-point immediate */		
463	//	FMOV	//0x1E201000,/*	FMOV	ARM64Op_fmov_scalar_immediate_Single_precision
464	//	FMOV	//0x1E601000,/*	FMOV	ARM64Op_fmov_scalar_immediate_Double_precision
465	//	**Floating-point compare**	/* Floating-point compare */		
466	//	FCMP	//0x1E202000,/*	FCMP	ARM64Op_fcmp_Single_precision */
467	//	FCMP	//0x1E202008,/*	FCMP	ARM64Op_fcmp_Single_precision_zero */
468	//	FCMPE	//0x1E202010,/*	FCMPE	ARM64Op_fcmpe_Single_precision */
469	//	FCMPE	//0x1E202018,/*	FCMPE	ARM64Op_fcmpe_Single_precision_zero */
470	//	FCMP	//0x1E602000,/*	FCMP	ARM64Op_fcmp_Double_precision */
471	//	FCMP	//0x1E602008,/*	FCMP	ARM64Op_fcmp_Double_precision_zero */
472	//	FCMPE	//0x1E602010,/*	FCMPE	ARM64Op_fcmpe_Double_precision */
473	//	FCMPE	//0x1E602018,/*	FCMPE	ARM64Op_fcmpe_Double_precision_zero */
474	//	**Floating-point data-processing (1 source)**	/* Floating-point data-processing (1 source) */		
475	//	FMOV	//0x1E204000,/*	FMOV	ARM64Op_fmov_register_Single_precision */
476	//	FABS	//0x1E20C000,/*	FABS	ARM64Op_fabs_scalar_Single_precision */
477	//	FNEG	//0x1E214000,/*	FNEG	ARM64Op_fneg_scalar_Single_precision */
478	//	FSQRT	//0x1E21C000,/*	FSQRT	ARM64Op_fsqrt_scalar_Single_precision */
479	//	FCVT	//0x1E22C000,/*	FCVT	ARM64Op_fcvt_Single_precision_to_double_precision
480	//	FCVT	//0x1E23C000,/*	FCVT	ARM64Op_fcvt_Single_precision_to_half_precision */
481	//	FRINTN	//0x1E244000,/*	FRINTN	ARM64Op_frintn_scalar_Single_precision */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
482	//	FRINTP	//0x1E24C000,/*	FRINTP	ARM64Op_frintp_scalar_Single_precision */
483	//	FRINTM	//0x1E254000,/*	FRINTM	ARM64Op_frintm_scalar_Single_precision */
484	//	FRINTZ	//0x1E25C000,/*	FRINTZ	ARM64Op_frintz_scalar_Single_precision */
485	//	FRINTA	//0x1E264000,/*	FRINTA	ARM64Op_frinta_scalar_Single_precision */
486	//	FRINTX	//0x1E274000,/*	FRINTX	ARM64Op_frintx_scalar_Single_precision */
487	//	FRINTI	//0x1E27C000,/*	FRINTI	ARM64Op_frinti_scalar_Single_precision */
488	//	FMOV	//0x1E604000,/*	FMOV	ARM64Op_fmov_register_Double_precision */
489	//	FABS	//0x1E60C000,/*	FABS	ARM64Op_fabs_scalar_Double_precision */
490	//	FNEG	//0x1E614000,/*	FNEG	ARM64Op_fneg_scalar_Double_precision */
491	//	FSQRT	//0x1E61C000,/*	FSQRT	ARM64Op_fsqrt_scalar_Double_precision */
492	//	FCVT	//0x1E62C000,/*	FCVT	ARM64Op_fcvt_Double_precision_to_single_precision
493	//	FCVT	//0x1E63C000,/*	FCVT	ARM64Op_fcvt_Double_precision_to_half_precision */
494	//	FRINTN	//0x1E644000,/*	FRINTN	ARM64Op_frintn_scalar_Double_precision */
495	//	FRINTP	//0x1E64C000,/*	FRINTP	ARM64Op_frintp_scalar_Double_precision */
496	//	FRINTM	//0x1E654000,/*	FRINTM	ARM64Op_frintm_scalar_Double_precision */
497	//	FRINTZ	//0x1E65C000,/*	FRINTZ	ARM64Op_frintz_scalar_Double_precision */
498	//	FRINTA	//0x1E664000,/*	FRINTA	ARM64Op_frinta_scalar_Double_precision */
499	//	FRINTX	//0x1E674000,/*	FRINTX	ARM64Op_frintx_scalar_Double_precision */
500	//	FRINTI	//0x1E67C000,/*	FRINTI	ARM64Op_frinti_scalar_Double_precision */
501	//	FCVT	//0x1EE24000,/*	FCVT	ARM64Op_fcvt_Half_precision_to_single_precision */
502	//	FCVT	//0x1EE2C000,/*	FCVT	ARM64Op_fcvt_Half_precision_to_double_precision */
503	//	**Floating-point<->integer conversions**	/* Floating-point<->integer conversions */		
504	//	FCVTNS	//0x1E200000,/*	FCVTNS	ARM64Op_fcvtns_scalar_Single_precision_to_32_bit
505	//	FCVTNU	//0x1E210000,/*	FCVTNU	ARM64Op_fcvtnu_scalar_Single_precision_to_32_bi
506	//	SCVTF	//0x1E220000,/*	SCVTF	ARM64Op_scvtf_scalar_integer_32_bit_to_single_pre
507	//	UCVTF	//0x1E230000,/*	UCVTF	ARM64Op_ucvtf_scalar_integer_32_bit_to_single_pre
508	//	FCVTAS	//0x1E240000,/*	FCVTAS	ARM64Op_fcvtas_scalar_Single_precision_to_32_bit
509	//	FCVTAU	//0x1E250000,/*	FCVTAU	ARM64Op_fcvtau_scalar_Single_precision_to_32_bit
510	//	FMOV	//0x1E260000,/*	FMOV	ARM64Op_fmov_general_Single_precision_to_32_bit
511	//	FMOV	//0x1E270000,/*	FMOV	ARM64Op_fmov_general_32_bit_to_single_precision
512	//	FCVTPS	//0x1E280000,/*	FCVTPS	ARM64Op_fcvtps_scalar_Single_precision_to_32_bit
513	//	FCVTPU	//0x1E290000,/*	FCVTPU	ARM64Op_fcvtpu_scalar_Single_precision_to_32_bit
514	//	FCVTMS	//0x1E300000,/*	FCVTMS	ARM64Op_fcvtms_scalar_Single_precision_to_32_b
515	//	FCVTMU	//0x1E310000,/*	FCVTMU	ARM64Op_fcvtmu_scalar_Single_precision_to_32_b

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
516	//	FCVTZS	//0x1E380000,/*	FCVTZS	ARM64Op_fcvtzs_scalar_integer_Single_precision_to
517	//	FCVTZU	//0x1E390000,/*	FCVTZU	ARM64Op_fcvtzu_scalar_integer_Single_precision_to
518	//	FCVTNS	//0x1E600000,/*	FCVTNS	ARM64Op_fcvtns_scalar_Double_precision_to_32_b
519	//	FCVTNU	//0x1E610000,/*	FCVTNU	ARM64Op_fcvtnu_scalar_Double_precision_to_32_b
520	//	SCVTF	//0x1E620000,/*	SCVTF	ARM64Op_scvtf_scalar_integer_32_bit_to_double_pre
521	//	UCVTF	//0x1E630000,/*	UCVTF	ARM64Op_ucvtf_scalar_integer_32_bit_to_double_pr
522	//	FCVTAS	//0x1E640000,/*	FCVTAS	ARM64Op_fcvtas_scalar_Double_precision_to_32_b
523	//	FCVTAU	//0x1E650000,/*	FCVTAU	ARM64Op_fcvtau_scalar_Double_precision_to_32_b
524	//	FCVTPS	//0x1E680000,/*	FCVTPS	ARM64Op_fcvtps_scalar_Double_precision_to_32_b
525	//	FCVTPU	//0x1E690000,/*	FCVTPU	ARM64Op_fcvtpu_scalar_Double_precision_to_32_b
526	//	FCVTMS	//0x1E700000,/*	FCVTMS	ARM64Op_fcvtms_scalar_Double_precision_to_32_b
527	//	FCVTMU	//0x1E710000,/*	FCVTMU	ARM64Op_fcvtmu_scalar_Double_precision_to_32_
528	//	FCVTZS	//0x1E780000,/*	FCVTZS	ARM64Op_fcvtzs_scalar_integer_Double_precision_t
529	//	FCVTZU	//0x1E790000,/*	FCVTZU	ARM64Op_fcvtzu_scalar_integer_Double_precision_
530	//	FCVTNS	//0x9E200000,/*	FCVTNS	ARM64Op_fcvtns_scalar_Single_precision_to_64_bit
531	//	FCVTNU	//0x9E210000,/*	FCVTNU	ARM64Op_fcvtnu_scalar_Single_precision_to_64_bi
532	//	SCVTF	//0x9E220000,/*	SCVTF	ARM64Op_scvtf_scalar_integer_64_bit_to_single_pre
533	//	UCVTF	//0x9E230000,/*	UCVTF	ARM64Op_ucvtf_scalar_integer_64_bit_to_single_pre
534	//	FCVTAS	//0x9E244000,/*	FCVTAS	ARM64Op_fcvtas_scalar_Single_precision_to_64_bit
535	//	FCVTAU	//0x9E250000,/*	FCVTAU	ARM64Op_fcvtau_scalar_Single_precision_to_64_bit
536	//	FCVTPS	//0x9E280000,/*	FCVTPS	ARM64Op_fcvtps_scalar_Single_precision_to_64_bit
537	//	FCVTPU	//0x9E290000,/*	FCVTPU	ARM64Op_fcvtpu_scalar_Single_precision_to_64_bit
538	//	FCVTMS	//0x9E300000,/*	FCVTMS	ARM64Op_fcvtms_scalar_Single_precision_to_64_b
539	//	FCVTMU	//0x9E318000,/*	FCVTMU	ARM64Op_fcvtmu_scalar_Single_precision_to_64_b
540	//	FCVTZS	//0x9E380000,/*	FCVTZS	ARM64Op_fcvtzs_scalar_integer_Single_precision_to
541	//	FCVTZU	//0x9E390000,/*	FCVTZU	ARM64Op_fcvtzu_scalar_integer_Single_precision_to
542	//	FCVTNS	//0x9E200000,/*	FCVTNS	ARM64Op_fcvtns_scalar_Double_precision_to_64_b
543	//	FCVTNU	//0x9E210000,/*	FCVTNU	ARM64Op_fcvtnu_scalar_Double_precision_to_64_b
544	//	SCVTF	//0x9E620000,/*	SCVTF	ARM64Op_scvtf_scalar_integer_64_bit_to_double_pre
545	//	UCVTF	//0x9E630000,/*	UCVTF	ARM64Op_ucvtf_scalar_integer_64_bit_to_double_pr
546	//	FCVTAS	//0x9E640000,/*	FCVTAS	ARM64Op_fcvtas_scalar_Double_precision_to_64_b
547	//	FCVTAU	//0x9E650000,/*	FCVTAU	ARM64Op_fcvtau_scalar_Double_precision_to_64_b
548	//	FMOV	//0x9E660000,/*	FMOV	ARM64Op_fmov_general_Double_precision_to_64_bit
549	//	FMOV	//0x9E670000,/*	FMOV	ARM64Op_fmov_general_64_bit_to_double_precision

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
550	//	FCVTPS	//0x9E680000,/*	FCVTPS	ARM64Op_fcvtps_scalar_Double_precision_to_64_b
551	//	FCVTPU	//0x9E690000,/*	FCVTPU	ARM64Op_fcvtpu_scalar_Double_precision_to_64_b
552	//	FCVTMS	//0x9E700000,/*	FCVTMS	ARM64Op_fcvtms_scalar_Double_precision_to_64_I
553	//	FCVTMU	//0x9E710000,/*	FCVTMU	ARM64Op_fcvtmu_scalar_Double_precision_to_64_
554	//	FCVTZS	//0x9E780000,/*	FCVTZS	ARM64Op_fcvtzs_scalar_integer_Double_precision_t
555	//	FCVTZU	//0x9E790000,/*	FCVTZU	ARM64Op_fcvtzu_scalar_integer_Double_precision_
556	//	FMOV	//0x9EAE0000,/*	FMOV	ARM64Op_fmov_general_Top_half_of_128_bit_to_64
557	//	FMOV	//0x9EAF0000,/*	FMOV	ARM64Op_fmov_general_64_bit_to_top_half_of_128_
558	//	**Floating-point data-processing (3 source)**	/* Floating-point data-processing (3 source) */		
559	//	FMADD	//0x1F000000,/*	FMADD	ARM64Op_fmadd_Single_precision */
560	//	FMSUB	//0x1F008000,/*	FMSUB	ARM64Op_fmsub_Single_precision */
561	//	FNMADD	//0x1F200000,/*	FNMADD	ARM64Op_fnmadd_Single_precision */
562	//	FNMSUB	//0x1F208000,/*	FNMSUB	ARM64Op_fnmsub_Single_precision */
563	//	FMADD	//0x1F400000,/*	FMADD	ARM64Op_fmadd_Double_precision */
564	//	FMSUB	//0x1F408000,/*	FMSUB	ARM64Op_fmsub_Double_precision */
565	//	FNMADD	//0x1F600000,/*	FNMADD	ARM64Op_fnmadd_Double_precision */
566	//	FNMSUB	//0x1F608000,/*	FNMSUB	ARM64Op_fnmsub_Double_precision */
567	//	**AdvSIMD scalar three same**	/* AdvSIMD scalar three same */		
568	//	SQADD	//0x5E200C00,/*	SQADD	ARM64Op_sqadd_Scalar */
569	//	SQSUB	//0x5E202C00,/*	SQSUB	ARM64Op_sqsub_Scalar */
570	//	CMGT	//0x5E203400,/*	CMGT	ARM64Op_cmgt_register_Scalar */
571	//	CMGE	//0x5E203C00,/*	CMGE	ARM64Op_cmge_register_Scalar */
572	//	SSHL	//0x5E204400,/*	SSHL	ARM64Op_sshl_Scalar */
573	//	SQSHL	//0x5E204C00,/*	SQSHL	ARM64Op_sqshl_register_Scalar */
574	//	SRSHL	//0x5E205400,/*	SRSHL	ARM64Op_srshl_Scalar */
575	//	SQRSHL	//0x5E205C00,/*	SQRSHL	ARM64Op_sqrshl_Scalar */
576	//	ADD	//0x5E208400,/*	ADD	ARM64Op_add_vector_Scalar */
577	//	CMTST	//0x5E208C00,/*	CMTST	ARM64Op_cmtst_Scalar */
578	//	SQDMULH	//0x5E20B400,/*	SQDMULH	ARM64Op_sqdmulh_vector_Scalar */
579	//	FMULX	//0x5E20DC00,/*	FMULX	ARM64Op_fmulx_Scalar */
580	//	FCMEQ	//0x5E20E400,/*	FCMEQ	ARM64Op_fcmeq_register_Scalar */
581	//	FRECPS	//0x5E20FC00,/*	FRECPS	ARM64Op_frecps_Scalar */
582	//	FRSQRTS	//0x5EA0FC00,/*	FRSQRTS	ARM64Op_frsqrts_Scalar */
583	//	UQADD	//0x7E200C00,/*	UQADD	ARM64Op_uqadd_Scalar */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
584	//	UQSUB	//0x7E202C00,/*	UQSUB	ARM64Op_uqsub_Scalar */
585	//	CMHI	//0x7E203400,/*	CMHI	ARM64Op_cmhi_register_Scalar */
586	//	CMHS	//0x7E203C00,/*	CMHS	ARM64Op_cmhs_register_Scalar */
587	//	USHL	//0x7E204400,/*	USHL	ARM64Op_ushl_Scalar */
588	//	UQSHL	//0x7E204C00,/*	UQSHL	ARM64Op_uqshl_register_Scalar */
589	//	URSHL	//0x7E205400,/*	URSHL	ARM64Op_urshl_Scalar */
590	//	UQRSHL	//0x7E205C00,/*	UQRSHL	ARM64Op_uqrshl_Scalar */
591	//	SUB	//0x7E208400,/*	SUB	ARM64Op_sub_vector_Scalar */
592	//	CMEQ	//0x7E208C00,/*	CMEQ	ARM64Op_cmeq_register_Scalar */
593	//	SQRDMULH	//0x7E20B400,/*	SQRDMULH	ARM64Op_sqrdmulh_vector_Scalar */
594	//	FCMGE	//0x7E20E400,/*	FCMGE	ARM64Op_fcmge_register_Scalar */
595	//	FACGE	//0x7E20EC00,/*	FACGE	ARM64Op_facge_Scalar */
596	//	FABD	//0x7EA0D400,/*	FABD	ARM64Op_fabd_Scalar */
597	//	FCMGT	//0x7EA0E400,/*	FCMGT	ARM64Op_fcmgt_register_Scalar */
598	//	FACGT	//0x7EA0EC00,/*	FACGT	ARM64Op_facgt_Scalar */
599	//	**AdvSIMD scalar three different**	/* AdvSIMD scalar three different */		
600	//	SQDMLAL	//0x5E209000,/*	SQDMLAL	ARM64Op_sqdmlal_vector_Scalarwrites to low half
601	//	SQDMLAL2	//0x5E209000,/*	SQDMLAL2	ARM64Op_sqdmlal2_vector_Scalarwrites to high ha
602	//	SQDMLSL	//0x5E20B000,/*	SQDMLSL	ARM64Op_sqdmlsl_vector_Scalarwrites to low half
603	//	SQDMLSL2	//0x5E20B000,/*	SQDMLSL2	ARM64Op_sqdmlsl2_vector_Scalarwrites to high ha
604	//	SQDMULL	//0x5E20D000,/*	SQDMULL	ARM64Op_sqdmull_vector_Scalarwrites to low half
605	//	SQDMULL2	//0x5E20D000,/*	SQDMULL2	ARM64Op_sqdmull2_vector_Scalarwrites to high h
606	//	**AdvSIMD scalar two-reg misc**	/* AdvSIMD scalar two-reg misc */		
607	//	SUQADD	//0x5E203800,/*	SUQADD	ARM64Op_suqadd_Scalar */
608	//	SQABS	//0x5E207800,/*	SQABS	ARM64Op_sqabs_Scalar */
609	//	CMGT	//0x5E208800,/*	CMGT	ARM64Op_cmgt_zero_Scalar */
610	//	CMEQ	//0x5E209800,/*	CMEQ	ARM64Op_cmeq_zero_Scalar */
611	//	CMLT	//0x5E20A800,/*	CMLT	ARM64Op_cmlt_zero_Scalar */
612	//	ABS	//0x5E20B800,/*	ABS	ARM64Op_abs_Scalar */
613	//	SQXTN	//0x5E214800,/*	SQXTN	ARM64Op_sqxtn_Scalarwrites to low half of the dest.
614	//	SQXTN2	//0x5E214800,/*	SQXTN2	ARM64Op_sqxtn2_Scalarwrites to high half of the de
615	//	FCVTNS	//0x5E21A800,/*	FCVTNS	ARM64Op_fcvtns_vector_Scalar */
616	//	FCVTMS	//0x5E21B800,/*	FCVTMS	ARM64Op_fcvtms_vector_Scalar */
617	//	FCVTAS	//0x5E21C800,/*	FCVTAS	ARM64Op_fcvtas_vector_Scalar */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
618	//	SCVTF	//0x5E21D800,/*	SCVTF	ARM64Op_scvtf_vector_integer_Scalar */
619	//	FCMGT	//0x5EA0C800,/*	FCMGT	ARM64Op_fcmgt_zero_Scalar */
620	//	FCMEQ	//0x5EA0D800,/*	FCMEQ	ARM64Op_fcmeq_zero_Scalar */
621	//	FCMLT	//0x5EA0E800,/*	FCMLT	ARM64Op_fcmlt_zero_Scalar */
622	//	FCVTPS	//0x5EA1A800,/*	FCVTPS	ARM64Op_fcvtps_vector_Scalar */
623	//	FCVTZS	//0x5EA1B800,/*	FCVTZS	ARM64Op_fcvtzs_vector_integer_Scalar */
624	//	FRECPE	//0x5EA1D800,/*	FRECPE	ARM64Op_frecpe_Scalar */
625	//	FRECPX	//0x5EA1F800,/*	FRECPX	ARM64Op_frecpx */
626	//	USQADD	//0x7E203800,/*	USQADD	ARM64Op_usqadd_Scalar */
627	//	SQNEG	//0x7E207800,/*	SQNEG	ARM64Op_sqneg_Scalar */
628	//	CMGE	//0x7E208800,/*	CMGE	ARM64Op_cmge_zero_Scalar */
629	//	CMLE	//0x7E209800,/*	CMLE	ARM64Op_cmle_zero_Scalar */
630	//	NEG	//0x7E20B800,/*	NEG	ARM64Op_neg_vector_Scalar */
631	//	SQXTUN	//0x7E212800,/*	SQXTUN	ARM64Op_sqxtun_Scalarwrites to low half of the des
632	//	SQXTUN2	//0x7E212800,/*	SQXTUN2	ARM64Op_sqxtun2_Scalarwrites to high half of the
633	//	UQXTN	//0x7E214800,/*	UQXTN	ARM64Op_uqxtn_Scalarwrites to low half of the dest.
634	//	UQXTN2	//0x7E214800,/*	UQXTN2	ARM64Op_uqxtn2_Scalarwrites to high half of the de
635	//	FCVTXN	//0x7E216800,/*	FCVTXN	ARM64Op_fcvtxn_Scalarwrites to low half of the dest
636	//	FCVTXN2	//0x7E216800,/*	FCVTXN2	ARM64Op_fcvtxn2_Scalarwrites to high half of the d
637	//	FCVTNU	//0x7E21A800,/*	FCVTNU	ARM64Op_fcvtnu_vector_Scalar */
638	//	FCVTMU	//0x7E21B800,/*	FCVTMU	ARM64Op_fcvtmu_vector_Scalar */
639	//	FCVTAU	//0x7E21C800,/*	FCVTAU	ARM64Op_fcvtau_vector_Scalar */
640	//	UCVTF	//0x7E21D800,/*	UCVTF	ARM64Op_ucvtf_vector_integer_Scalar */
641	//	FCMGE	//0x7EA0C800,/*	FCMGE	ARM64Op_fcmge_zero_Scalar */
642	//	FCMLE	//0x7EA0D800,/*	FCMLE	ARM64Op_fcmle_zero_Scalar */
643	//	FCVTPU	//0x7EA1A800,/*	FCVTPU	ARM64Op_fcvtpu_vector_Scalar */
644	//	FCVTZU	//0x7EA1B800,/*	FCVTZU	ARM64Op_fcvtzu_vector_integer_Scalar */
645	//	FRSQRTE	//0x7EA1D800,/*	FRSQRTE	ARM64Op_frsqrte_Scalar */
646	//	**AdvSIMD scalar pairwise**	/* AdvSIMD scalar pairwise */		
647	//	ADDP	//0x5E31B800,/*	ADDP	ARM64Op_addp_scalar */
648	//	FMAXNMP	//0x7E30C800,/*	FMAXNMP	ARM64Op_fmaxnmp_scalar */
649	//	FADDP	//0x7E30D800,/*	FADDP	ARM64Op_faddp_scalar */
650	//	FMAXP	//0x7E30F800,/*	FMAXP	ARM64Op_fmaxp_scalar */
651	//	FMINNMP	//0x7EB0C800,/*	FMINNMP	ARM64Op_fminnmp_scalar */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
652	//	FMINP	//0x7EB0F800,/* FMINP	ARM64Op_fminp_scalar */	
653	//	**AdvSIMD scalar copy**	/* AdvSIMD scalar copy */		
654	//	DUP	//0x5E000400,/* DUP	ARM64Op_dup_element_Scalar */	
655	//	**AdvSIMD scalar x indexed element**	/* AdvSIMD scalar x indexed element */		
656	//	SQDMLAL	//0x5F003000,/* SQDMLAL	ARM64Op_sqdmlal_by_element_Scalar */	
657	//	SQDMLAL2	//0x5F003000,/* SQDMLAL2	ARM64Op_sqdmlal2_by_element_Scalar */	
658	//	SQDMLSL	//0x5F007000,/* SQDMLSL	ARM64Op_sqdmlsl_by_element_Scalar */	
659	//	SQDMLSL2	//0x5F007000,/* SQDMLSL2	ARM64Op_sqdmlsl2_by_element_Scalar */	
660	//	SQDMULL	//0x5F00B000,/* SQDMULL	ARM64Op_sqdmull_by_element_Scalar */	
661	//	SQDMULL2	//0x5F00B000,/* SQDMULL2	ARM64Op_sqdmull2_by_element_Scalar */	
662	//	SQDMULH	//0x5F00C000,/* SQDMULH	ARM64Op_sqdmulh_by_element_Scalar */	
663	//	SQRDMULH	//0x5F00D000,/* SQRDMULH	ARM64Op_sqrdmulh_by_element_Scalar */	
664	//	FMLA	//0x5F801000,/* FMLA	ARM64Op_fmla_by_element_Scalar */	
665	//	FMLS	//0x5F805000,/* FMLS	ARM64Op_fmls_by_element_Scalar */	
666	//	FMUL	//0x5F809000,/* FMUL	ARM64Op_fmul_by_element_Scalar */	
667	//	FMULX	//0x7F809000,/* FMULX	ARM64Op_fmulx_by_element_Scalar */	
668	//	**AdvSIMD scalar shift by immediate**	/* AdvSIMD scalar shift by immediate */		
669	//	SSHR	//0x5F000400,/* SSHR	ARM64Op_sshr_Scalarimmh != 0000 */	
670	//	SSRA	//0x5F001400,/* SSRA	ARM64Op_ssra_Scalarimmh != 0000 */	
671	//	SRSHR	//0x5F002400,/* SRSHR	ARM64Op_srshr_Scalarimmh != 0000 */	
672	//	SRSRA	//0x5F003400,/* SRSRA	ARM64Op_srsra_Scalarimmh != 0000 */	
673	//	SHL	//0x5F005400,/* SHL	ARM64Op_shl_Scalarimmh != 0000 */	
674	//	SQSHL	//0x5F007400,/* SQSHL	ARM64Op_sqshl_immediate_Scalarimmh != 0000 */	
675	//	SQSHRN	//0x5F009400,/* SQSHRN	ARM64Op_sqshrn_Scalarimmh != 0000 */	
676	//	SQSHRN2	//0x5F009400,/* SQSHRN2	ARM64Op_sqshrn2_Scalarimmh != 0000 */	
677	//	SQRSHRN	//0x5F009C00,/* SQRSHRN	ARM64Op_sqrshrn_Scalarimmh != 0000 */	
678	//	SQRSHRN2	//0x5F009C00,/* SQRSHRN2	ARM64Op_sqrshrn2_Scalarimmh != 0000 */	
679	//	SCVTF	//0x5F00E400,/* SCVTF	ARM64Op_scvtf_vector_fixed_point_Scalarimmh != 0(	
680	//	FCVTZS	//0x5F00FC00,/* FCVTZS	ARM64Op_fcvtzs_vector_fixed_point_Scalarimmh !=	
681	//	USHR	//0x7F000400,/* USHR	ARM64Op_ushr_Scalarimmh != 0000 */	
682	//	USRA	//0x7F001400,/* USRA	ARM64Op_usra_Scalarimmh != 0000 */	
683	//	URSHR	//0x7F002400,/* URSHR	ARM64Op_urshr_Scalarimmh != 0000 */	
684	//	URSRA	//0x7F003400,/* URSRA	ARM64Op_ursra_Scalarimmh != 0000 */	
685	//	SRI	//0x7F004400,/* SRI	ARM64Op_sri_Scalarimmh != 0000 */	

1	in_use	Opcode	//BINARY Opcode Opcodecomments
686	//	SLI	//0x7F005400,/* SLI ARM64Op_sli_Scalarimmh != 0000 */
687	//	SQSHLU	//0x7F006400,/* SQSHLU ARM64Op_sqshlu_Scalarimmh != 0000 */
688	//	UQSHL	//0x7F007400,/* UQSHL ARM64Op_uqshl_immediate_Scalarimmh != 0000 */
689	//	SQSHRUN	//0x7F008400,/* SQSHRUN ARM64Op_sqshrun_Scalarimmh != 0000 */
690	//	SQSHRUN2	//0x7F008400,/* SQSHRUN2 ARM64Op_sqshrun2_Scalarimmh != 0000 */
691	//	SQRSHRUN	//0x7F008C00,/* SQRSHRUN ARM64Op_sqrshrun_Scalarimmh != 0000 */
692	//	SQRSHRUN2	//0x7F008C00,/* SQRSHRUN2 ARM64Op_sqrshrun2_Scalarimmh != 0000 */
693	//	UQSHRN	//0x7F009400,/* UQSHRN ARM64Op_uqshrn_Scalarimmh != 0000 */
694	//	UQRSHRN	//0x7F009C00,/* UQRSHRN ARM64Op_uqrshrn_Scalarimmh != 0000 */
695	//	UQRSHRN2	//0x7F009C00,/* UQRSHRN2 ARM64Op_uqrshrn2_Scalarimmh != 0000 */
696	//	UCVTF	//0x7F00E400,/* UCVTF ARM64Op_ucvtf_vector_fixed_point_Scalarimmh != 0
697	//	FCVTZU	//0x7F00FC00,/* FCVTZU ARM64Op_fcvtzu_vector_fixed_point_Scalarimmh !=
698	//	**Crypto three-reg SHA**	/* Crypto three-reg SHA */
699	//	SHA1C	//0x5E000000,/* SHA1C ARM64Op_sha1c */
700	//	SHA1P	//0x5E001000,/* SHA1P ARM64Op_sha1p */
701	//	SHA1M	//0x5E002000,/* SHA1M ARM64Op_sha1m */
702	//	SHA1SU0	//0x5E003000,/* SHA1SU0 ARM64Op_sha1su0 */
703	//	SHA256H	//0x5E004000,/* SHA256H ARM64Op_sha256h */
704	//	SHA256H2	//0x5E005000,/* SHA256H2 ARM64Op_sha256h2 */
705	//	SHA256SU1	//0x5E006000,/* SHA256SU1 ARM64Op_sha256su1 */
706	//	**Crypto two-reg SHA**	/* Crypto two-reg SHA */
707	//	SHA1H	//0x5E280800,/* SHA1H ARM64Op_sha1h */
708	//	SHA1SU1	//0x5E281800,/* SHA1SU1 ARM64Op_sha1su1 */
709	//	SHA256SU0	//0x5E282800,/* SHA256SU0 ARM64Op_sha256su0 */
710	//	**Crypto AES**	/* Crypto AES */
711	//	AESE	//0x4E284800,/* AESE ARM64Op_aese */
712	//	AESD	//0x4E285800,/* AESD ARM64Op_aesd */
713	//	AESMC	//0x4E286800,/* AESMC ARM64Op_aesmc */
714	//	AESIMC	//0x4E287800,/* AESIMC ARM64Op_aesimc */
715	//	**AdvSIMD three same**	/* AdvSIMD three same */
716	//	SHADD	//0x0E200400,/* SHADD ARM64Op_shadd */
717	//	SQADD	//0x0E200C00,/* SQADD ARM64Op_sqadd_Vector */
718	//	SRHADD	//0x0E201400,/* SRHADD ARM64Op_srhadd */
719	//	SHSUB	//0x0E202400,/* SHSUB ARM64Op_shsub */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
720	//	SQSUB	//0x0E202C00,/*	SQSUB	ARM64Op_sqsub_Vector */
721	//	CMGT	//0x0E203400,/*	CMGT	ARM64Op_cmgt_register_Vector */
722	//	CMGE	//0x0E203C00,/*	CMGE	ARM64Op_cmge_register_Vector */
723	//	SSHL Vector	//0x0E204400,/*	SSHL Vecto	ARM64Op_sshl vector */
724	//	SQSHL	//0x0E204C00,/*	SQSHL	ARM64Op_sqshl_register_Vector */
725	//	SRSHL	//0x0E205400,/*	SRSHL	ARM64Op_srshl_Vector */
726	//	SQRSHL	//0x0E205C00,/*	SQRSHL	ARM64Op_sqrshl_Vector */
727	//	SMAX	//0x0E206400,/*	SMAX	ARM64Op_smax */
728	//	SMIN	//0x0E206C00,/*	SMIN	ARM64Op_smin */
729	//	SABD	//0x0E207400,/*	SABD	ARM64Op_sabd */
730	//	SABA	//0x0E207C00,/*	SABA	ARM64Op_saba */
731	//	ADD	//0x0E208400,/*	ADD	ARM64Op_add_vector_Vector */
732	//	CMTST	//0x0E208C00,/*	CMTST	ARM64Op_cmtst_Vector */
733	//	MLA	//0x0E209400,/*	MLA	ARM64Op_mla_vector */
734	//	MUL	//0x0E209C00,/*	MUL	ARM64Op_mul_vector */
735	//	SMAXP	//0x0E20A400,/*	SMAXP	ARM64Op_smaxp */
736	//	SMINP	//0x0E20AC00,/*	SMINP	ARM64Op_sminp */
737	//	SQDMULH	//0x0E20B400,/*	SQDMULH	ARM64Op_sqdmulh_vector_Vector */
738	//	ADDP	//0x0E20BC00,/*	ADDP	ARM64Op_addp_vector */
739	//	FMAXNM	//0x0E20C400,/*	FMAXNM	ARM64Op_fmaxnm_vector */
740	//	FMLA	//0x0E20CC00,/*	FMLA	ARM64Op_fmla_vector */
741	//	FADD	//0x0E20D400,/*	FADD	ARM64Op_fadd_vector */
742	//	FMULX	//0x0E20DC00,/*	FMULX	ARM64Op_fmulx_Vector */
743	//	FCMEQ	//0x0E20E400,/*	FCMEQ	ARM64Op_fcmeq_register_Vector */
744	//	FMAX	//0x0E20F400,/*	FMAX	ARM64Op_fmax_vector */
745	//	FRECPS	//0x0E20FC00,/*	FRECPS	ARM64Op_frecps_Vector */
746	//	AND	//0x0E201C00,/*	AND	ARM64Op_and_vector */
747	//	BIC	//0x0E601C00,/*	BIC	ARM64Op_bic_vector_register */
748	//	FMINNM	//0x0EA0C400,/*	FMINNM	ARM64Op_fminnm_vector */
749	//	FMLS	//0x0EA0CC00,/*	FMLS	ARM64Op_fmls_vector */
750	//	FSUB	//0x0EA0D400,/*	FSUB	ARM64Op_fsub_vector */
751	//	FMIN	//0x0EA0F400,/*	FMIN	ARM64Op_fmin_vector */
752	//	FRSQRTS	//0x0EA0FC00,/*	FRSQRTS	ARM64Op_frsqrts_Vector */
753	//	ORR	//0x0EA01C00,/*	ORR	ARM64Op_orr_vector_register */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
754	//	ORN	//0x0EE01C00,/* ORN ARM64Op_orn_vector */
755	//	UHADD	//0x2E200400,/* UHADD ARM64Op_uhadd */
756	//	UQADD	//0x2E200C00,/* UQADD ARM64Op_uqadd_Vector */
757	//	URHADD	//0x2E201400,/* URHADD ARM64Op_urhadd */
758	//	UHSUB	//0x2E202400,/* UHSUB ARM64Op_uhsub */
759	//		//0x2E202C00,/* ARM64Op_ _Vector */
760	//	CMHI	//0x2E203400,/* CMHI ARM64Op_cmhi_register_Vector */
761	//	CMHS	//0x2E203C00,/* CMHS ARM64Op_cmhs_register_Vector */
762	//	USHL	//0x2E204400,/* USHL ARM64Op_ushl_Vector */
763	//	UQSHL	//0x2E204C00,/* UQSHL ARM64Op_uqshl_register_Vector */
764	//	URSHL	//0x2E205400,/* URSHL ARM64Op_urshl_Vector */
765	//	UQRSHL	//0x2E205C00,/* UQRSHL ARM64Op_uqrshl_Vector */
766	//	UMAX	//0x2E206400,/* UMAX ARM64Op_umax */
767	//	UMIN	//0x2E206C00,/* UMIN ARM64Op_umin */
768	//	UABD	//0x2E207400,/* UABD ARM64Op_uabd */
769	//	UABA	//0x2E207C00,/* UABA ARM64Op_uaba */
770	//	SUB	//0x2E208400,/* SUB ARM64Op_sub_vector_Vector */
771	//	CMEQ	//0x2E208C00,/* CMEQ ARM64Op_cmeq_register_Vector */
772	//	MLS	//0x2E209400,/* MLS ARM64Op_mls_vector */
773	//	PMUL	//0x2E209C00,/* PMUL ARM64Op_pmul */
774	//	UMAXP	//0x2E20A400,/* UMAXP ARM64Op_umaxp */
775	//	UMINP	//0x2E20AC00,/* UMINP ARM64Op_uminp */
776	//	SQRDMULH	//0x2E20B400,/* SQRDMULH ARM64Op_sqrdmulh_vector_Vector */
777	//	FMAXNMP	//0x2E20B400,/* FMAXNMP ARM64Op_fmaxnmp_vector */
778	//	FADDP	//0x2E20D400,/* FADDP ARM64Op_faddp_vector */
779	//	FMUL	//0x2E20DC00,/* FMUL ARM64Op_fmul_vector */
780	//	FCMGE	//0x2E20E400,/* FCMGE ARM64Op_fcmge_register_Vector */
781	//	FACGE	//0x2E20EC00,/* FACGE ARM64Op_facge_Vector */
782	//	FMAXP	//0x2E20F400,/* FMAXP ARM64Op_fmaxp_vector */
783	//	FDIV	//0x2E20FC00,/* FDIV ARM64Op_fdiv_vector */
784	//	EOR	//0x2E201C00,/* EOR ARM64Op_eor_vector */
785	//	BSL	//0x2E601C00,/* BSL ARM64Op_bsl */
786	//	FMINNMP	//0x2EA0C400,/* FMINNMP ARM64Op_fminnmp_vector */
787	//	FABD	//0x2EA0D400,/* FABD ARM64Op_fabd_Vector */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
788	//	FCMGT	//0x2EA0E400,/* FCMGT ARM64Op_fcmgt_register_Vector */
789	//	FACGT	//0x2EA0EC00,/* FACGT ARM64Op_facgt_Vector */
790	//	FMINP	//0x2EA0F400,/* FMINP ARM64Op_fminp_vector */
791	//	BIT	//0x2EA01C00,/* BIT ARM64Op_bit */
792	//	BIF	//0x2EE01C00,/* BIF ARM64Op_bif */
793	//	**AdvSIMD three different**	/* AdvSIMD three different */
794	//	SADDL	//0x0E200000,/* SADDL ARM64Op_saddlwrites to low half of the dest. register
795	//	SADDL2	//0x4E200000,/* SADDL2 ARM64Op_saddl2writes to high half of the dest. regis
796	//	SADDW	//0x0E201000,/* SADDW ARM64Op_saddwwrites to low half of the dest. regist
797	//	SADDW2	//0x4E201000,/* SADDW2 ARM64Op_saddw2writes to high half of the dest. reg
798	//	SSUBL	//0x0E202000,/* SSUBL ARM64Op_ssublwrites to low half of the dest. register
799	//	SSUBL2	//0x4E202000,/* SSUBL2 ARM64Op_ssubl2writes to high half of the dest. regist
800	//	SSUBW	//0x0E203000,/* SSUBW ARM64Op_ssubwwrites to low half of the dest. registe
801	//	SSUBW2	//0x4E203000,/* SSUBW2 ARM64Op_ssubw2writes to high half of the dest. reg
802	//	ADDHN	//0x0E204000,/* ADDHN ARM64Op_addhnwrites to low half of the dest. registe
803	//	ADDHN2	//0x4E204000,/* ADDHN2 ARM64Op_addhn2writes to high half of the dest. regi
804	//	SABAL	//0x0E205000,/* SABAL ARM64Op_sabalwrites to low half of the dest. register
805	//	SABAL2	//0x4E205000,/* SABAL2 ARM64Op_sabal2writes to high half of the dest. regist
806	//	SUBHN	//0x0E206000,/* SUBHN ARM64Op_subhnwrites to low half of the dest. registe
807	//	SUBHN2	//0x4E206000,/* SUBHN2 ARM64Op_subhn2writes to high half of the dest. regi
808	//	SABDL	//0x0E207000,/* SABDL ARM64Op_sabdlwrites to low half of the dest. register
809	//	SABDL2	//0x4E207000,/* SABDL2 ARM64Op_sabdl2writes to high half of the dest. regist
810	//	SMLAL	//0x0E208000,/* SMLAL ARM64Op_smlal_vectorwrites to low half of the dest. r
811	//	SMLAL2	//0x4E208000,/* SMLAL2 ARM64Op_smlal2_vectorwrites to high half of the des
812	//	SQDMLAL	//0x0E209000,/* SQDMLAL ARM64Op_sqdmlal_vector_Vectorwrites to low half
813	//	SQDMLAL2	//0x4E209000,/* SQDMLAL2 ARM64Op_sqdmlal2_vector_Vectorwrites to high h
814	//	SMLSL	//0x0E20A000,/* SMLSL ARM64Op_smlsl_vectorwrites to low half of the dest. r
815	//	SMLSL2	//0x4E20A000,/* SMLSL2 ARM64Op_smlsl2_vectorwrites to high half of the des
816	//	SQDMLSL	//0x0E20B000,/* SQDMLSL ARM64Op_sqdmlsl_vector_Vectorwrites to low half
817	//	SQDMLSL2	//0x4E20B000,/* SQDMLSL2 ARM64Op_sqdmlsl2_vector_Vectorwrites to high h
818	//	SMULL	//0x0E20C000,/* SMULL ARM64Op_smull_vectorwrites to low half of the dest.
819	//	SMULL2	//0x4E20C000,/* SMULL2 ARM64Op_smull2_vectorwrites to high half of the des
820	//	SQDMULL	//0x0E20D000,/* SQDMULL ARM64Op_sqdmull_vector_Vectorwrites to low half
821	//	SQDMULL2	//0x4E20D000,/* SQDMULL2 ARM64Op_sqdmull2_vector_Vectorwrites to high h

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
822	//	PMULL	//0x0E20E000,/*	PMULL	ARM64Op_pmullwrites to low half of the dest. register
823	//	PMULL2	//0x4E20E000,/*	PMULL2	ARM64Op_pmull2writes to high half of the dest. regis
824	//	UADDL	//0x2E200000,/*	UADDL	ARM64Op_uaddlwrites to low half of the dest. register
825	//	UADDL2	//0x6E200000,/*	UADDL2	ARM64Op_uaddl2writes to high half of the dest. regis
826	//	UADDW	//0x2E201000,/*	UADDW	ARM64Op_uaddwwrites to low half of the dest. regist
827	//	UADDW2	//0x6E201000,/*	UADDW2	ARM64Op_uaddw2writes to high half of the dest. reg
828	//	USUBL	//0x2E202000,/*	USUBL	ARM64Op_usublwrites to low half of the dest. register
829	//	USUBL2	//0x6E202000,/*	USUBL2	ARM64Op_usubl2writes to high half of the dest. regis
830	//	USUBW	//0x2E203000,/*	USUBW	ARM64Op_usubwwrites to low half of the dest. regist
831	//	USUBW2	//0x6E203000,/*	USUBW2	ARM64Op_usubw2writes to high half of the dest. reg
832	//	RADDHN	//0x2E204000,/*	RADDHN	ARM64Op_raddhnwrites to low half of the dest. regis
833	//	RADDHN2	//0x6E204000,/*	RADDHN2	ARM64Op_raddhn2writes to high half of the dest. re
834	//	UABAL	//0x2E205000,/*	UABAL	ARM64Op_uabalwrites to low half of the dest. register
835	//	UABAL2	//0x6E205000,/*	UABAL2	ARM64Op_uabal2writes to high half of the dest. regis
836	//	RSUBHN	//0x2E206000,/*	RSUBHN	ARM64Op_rsubhnwrites to low half of the dest. regis
837	//	RSUBHN2	//0x6E206000,/*	RSUBHN2	ARM64Op_rsubhn2writes to high half of the dest. re
838	//	UABDL	//0x2E207000,/*	UABDL	ARM64Op_uabdlwrites to low half of the dest. register
839	//	UABDL2	//0x6E207000,/*	UABDL2	ARM64Op_uabdl2writes to high half of the dest. regis
840	//	UMLAL	//0x2E208000,/*	UMLAL	ARM64Op_umlal_vectorwrites to low half of the dest.
841	//	UMLAL2	//0x6E208000,/*	UMLAL2	ARM64Op_umlal2_vectorwrites to high half of the des
842	//	UMLSL	//0x2E20A000,/*	UMLSL	ARM64Op_umlsl_vectorwrites to low half of the dest.
843	//	UMLSL2	//0x6E20A000,/*	UMLSL2	ARM64Op_umlsl2_vectorwrites to high half of the des
844	//	UMULL	//0x2E20C000,/*	UMULL	ARM64Op_umull_vectorwrites to low half of the dest.
845	//	UMULL2	//0x6E20C000,/*	UMULL2	ARM64Op_umull2_vectorwrites to high half of the de
846	//	**AdvSIMD two-reg misc**	/* AdvSIMD two-reg misc */		
847	//	REV64	//0x0E200800,/*	REV64	ARM64Op_rev64 */
848	//	REV16	//0x0E201800,/*	REV16	ARM64Op_rev16_vector */
849	//	SADDLP	//0x0E202800,/*	SADDLP	ARM64Op_saddlp */
850	//	SUQADD	//0x0E203800,/*	SUQADD	ARM64Op_suqadd_Vector */
851	//	CLS	//0x0E204800,/*	CLS	ARM64Op_cls_vector */
852	//	CNT	//0x0E205800,/*	CNT	ARM64Op_cnt */
853	//	SADALP	//0x0E206800,/*	SADALP	ARM64Op_sadalp */
854	//	SQABS	//0x0E207800,/*	SQABS	ARM64Op_sqabs_Vector */
855	//	CMGT	//0x0E208800,/*	CMGT	ARM64Op_cmgt_zero_Vector */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
856	//	CMEQ	//0x0E209800,/*	CMEQ	ARM64Op_cmeq_zero_Vector */
857	//	CMLT	//0x0E20A800,/*	CMLT	ARM64Op_cmlt_zero_Vector */
858	//	ABS	//0x0E20B800,/*	ABS	ARM64Op_abs_Vector */
859	//	XTN	//0x0E212800,/*	XTN	ARM64Op_xtn */
860	//	XTN2	//0x0E212800,/*	XTN2	ARM64Op_xtn2 */
861	//	SQXTN	//0x0E214800,/*	SQXTN	ARM64Op_sqxtn_Vector */
862	//	SQXTN2	//0x0E214800,/*	SQXTN2	ARM64Op_sqxtn2_Vector */
863	//	FCVTN	//0x0E216800,/*	FCVTN	ARM64Op_fcvtn */
864	//	FCVTN2	//0x0E216800,/*	FCVTN2	ARM64Op_fcvtn2 */
865	//	FCVTL	//0x0E217800,/*	FCVTL	ARM64Op_fcvtl */
866	//	FCVTL2	//0x0E217800,/*	FCVTL2	ARM64Op_fcvtl2 */
867	//	FRINTN	//0x0E218800,/*	FRINTN	ARM64Op_frintn_vector */
868	//	FRINTM	//0x0E219800,/*	FRINTM	ARM64Op_frintm_vector */
869	//	FCVTNS	//0x0E21A800,/*	FCVTNS	ARM64Op_fcvtns_vector_Vector */
870	//	FCVTMS	//0x0E21B800,/*	FCVTMS	ARM64Op_fcvtms_vector_Vector */
871	//	FCVTAS	//0x0E21C800,/*	FCVTAS	ARM64Op_fcvtas_vector_Vector */
872	//	SCVTF	//0x0E21D800,/*	SCVTF	ARM64Op_scvtf_vector_integer_Vector */
873	//	FCMGT	//0x0EA0C800,/*	FCMGT	ARM64Op_fcmgt_zero_Vector */
874	//	FCMEQ	//0x0EA0D800,/*	FCMEQ	ARM64Op_fcmeq_zero_Vector */
875	//	FCMLT	//0x0EA0E800,/*	FCMLT	ARM64Op_fcmlt_zero_Vector */
876	//	FABS	//0x0EA0F800,/*	FABS	ARM64Op_fabs_vector */
877	//	FRINTP	//0x0EA18800,/*	FRINTP	ARM64Op_frintp_vector */
878	//	FRINTZ	//0x0EA19800,/*	FRINTZ	ARM64Op_frintz_vector */
879	//	FCVTPS	//0x0EA1A800,/*	FCVTPS	ARM64Op_fcvtps_vector_Vector */
880	//	FCVTZS	//0x0EA1B800,/*	FCVTZS	ARM64Op_fcvtzs_vector_integer_Vector */
881	//	URECPE	//0x0EA1C800,/*	URECPE	ARM64Op_urecpe */
882	//	FRECPE	//0x0EA1D800,/*	FRECPE	ARM64Op_frecpe_Vector */
883	//	REV32	//0x2E200800,/*	REV32	ARM64Op_rev32_vector */
884	//	UADDLP	//0x2E202800,/*	UADDLP	ARM64Op_uaddlp */
885	//	USQADD	//0x2E203800,/*	USQADD	ARM64Op_usqadd_Vector */
886	//	CLZ	//0x2E204800,/*	CLZ	ARM64Op_clz_vector */
887	//	UADALP	//0x2E206800,/*	UADALP	ARM64Op_uadalp */
888	//	SQNEG	//0x2E207800,/*	SQNEG	ARM64Op_sqneg_Vector */
889	//	CMGE	//0x2E208800,/*	CMGE	ARM64Op_cmge_zero_Vector */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
890	//	CMLE	//0x2E209800,/*	CMLE	ARM64Op_cmle_zero_Vector */
891	//	NEG	//0x2E20B800,/*	NEG	ARM64Op_neg_vector_Vector */
892	//	SQXTUN	//0x2E212800,/*	SQXTUN	ARM64Op_sqxtun_Vector */
893	//	SQXTUN2	//0x2E212800,/*	SQXTUN2	ARM64Op_sqxtun2_Vector */
894	//	SHLL	//0x2E213800,/*	SHLL	ARM64Op_shll */
895	//	SHLL2	//0x2E213800,/*	SHLL2	ARM64Op_shll2 */
896	//	UQXTN	//0x2E214800,/*	UQXTN	ARM64Op_uqxtn_Vector */
897	//	UQXTN2	//0x2E214800,/*	UQXTN2	ARM64Op_uqxtn2_Vector */
898	//	FCVTXN	//0x2E216800,/*	FCVTXN	ARM64Op_fcvtxn_Vector */
899	//	FCVTXN2	//0x2E216800,/*	FCVTXN2	ARM64Op_fcvtxn2_Vector */
900	//	FRINTA	//0x2E218800,/*	FRINTA	ARM64Op_frinta_vector */
901	//	FRINTX	//0x2E219800,/*	FRINTX	ARM64Op_frintx_vector */
902	//	FCVTNU	//0x2E21A800,/*	FCVTNU	ARM64Op_fcvtnu_vector_Vector */
903	//	FCVTMU	//0x2E21B800,/*	FCVTMU	ARM64Op_fcvtmu_vector_Vector */
904	//	FCVTAU	//0x2E21C800,/*	FCVTAU	ARM64Op_fcvtau_vector_Vector */
905	//	UCVTF	//0x2E21D800,/*	UCVTF	ARM64Op_ucvtf_vector_integer_Vector */
906	//	NOT	//0x2E205800,/*	NOT	ARM64Op_not */
907	//	RBIT	//0x2E605800,/*	RBIT	ARM64Op_rbit_vector */
908	//	FCMGE	//0x2EA0C800,/*	FCMGE	ARM64Op_fcmge_zero_Vector */
909	//	FCMLE	//0x2EA0D800,/*	FCMLE	ARM64Op_fcmle_zero_Vector */
910	//	FNEG	//0x2EA0F800,/*	FNEG	ARM64Op_fneg_vector */
911	//	FRINTI	//0x2EA19800,/*	FRINTI	ARM64Op_frinti_vector */
912	//	FCVTPU	//0x2EA1A800,/*	FCVTPU	ARM64Op_fcvtpu_vector_Vector */
913	//	FCVTZU	//0x2EA1B800,/*	FCVTZU	ARM64Op_fcvtzu_vector_integer_Vector */
914	//	URSQRTE	//0x2EA1C800,/*	URSQRTE	ARM64Op_ursqrte */
915	//	FRSQRTE	//0x2EA1D800,/*	FRSQRTE	ARM64Op_frsqrte_Vector */
916	//	FSQRT	//0x2EA1F800,/*	FSQRT	ARM64Op_fsqrt_vector */
917	//	**AdvSIMD across lanes**	/* AdvSIMD across lanes */		
918	//	SADDLV	//0x0E303800,/*	SADDLV	ARM64Op_saddlv */
919	//	SMAXV	//0x0E30A800,/*	SMAXV	ARM64Op_smaxv */
920	//	SMINV	//0x0E31A800,/*	SMINV	ARM64Op_sminv */
921	//	ADDV	//0x0E31B800,/*	ADDV	ARM64Op_addv */
922	//	UADDLV	//0x2E303800,/*	UADDLV	ARM64Op_uaddlv */
923	//	UMAXV	//0x2E30A800,/*	UMAXV	ARM64Op_umaxv */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
924	//	UMINV	//0x2E31A800,/* UMINV ARM64Op_uminv */
925	//	FMAXNMV	//0x2E30C800,/* FMAXNMV ARM64Op_fmaxnmv */
926	//	FMAXV	//0x2E30F800,/* FMAXV ARM64Op_fmaxv */
927	//	FMINNMV	//0x2EB0C800,/* FMINNMV ARM64Op_fminnmv */
928	//	FMINV	//0x2EB0F800,/* FMINV ARM64Op_fminv */
929	//	**AdvSIMD copy**	/* AdvSIMD copy */
930	//	DUP	//0x0E000400,/* DUP ARM64Op_dup_element_Vector */
931	//	DUP	//0x0E000C00,/* DUP ARM64Op_dup_general */
932	//	SMOV	//0x0E002C00,/* SMOV ARM64Op_smov_32_bit */
933	//	UMOV	//0x0E003C00,/* UMOV ARM64Op_umov_32_bit */
934	//	INS	//0x4E001C00,/* INS ARM64Op_ins_general */
935	//	SMOV	//0x4E002C00,/* SMOV ARM64Op_smov_64_bit */
936	//	UMOV	//0x4E003C00,/* UMOV ARM64Op_umov_64_bit */
937	//	INS	//0x6E000400,/* INS ARM64Op_ins_element */
938	//	**AdvSIMD vector x indexed element**	/* AdvSIMD vector x indexed element */
939	//	SMLAL	//0x0F002000,/* SMLAL ARM64Op_smlal_by_element */
940	//	SMLAL2	//0x0F002000,/* SMLAL2 ARM64Op_smlal2_by_element */
941	//	SQDMLAL	//0x0F003000,/* SQDMLAL ARM64Op_sqdmlal_by_element_Vector */
942	//	SQDMLAL2	//0x0F003000,/* SQDMLAL2 ARM64Op_sqdmlal2_by_element_Vector */
943	//	SMLSL	//0x0F006000,/* SMLSL ARM64Op_smlsl_by_element */
944	//	SMLSL2	//0x0F006000,/* SMLSL2 ARM64Op_smlsl2_by_element */
945	//	SQDMLSL	//0x0F007000,/* SQDMLSL ARM64Op_sqdmlsl_by_element_Vector */
946	//	SQDMLSL2	//0x0F007000,/* SQDMLSL2 ARM64Op_sqdmlsl2_by_element_Vector */
947	//	MUL	//0x0F008000,/* MUL ARM64Op_mul_by_element */
948	//	SMULL	//0x0F00A000,/* SMULL ARM64Op_smull_by_element */
949	//	SMULL2	//0x0F00A000,/* SMULL2 ARM64Op_smull2_by_element */
950	//	SQDMULL	//0x0F00B000,/* SQDMULL ARM64Op_sqdmull_by_element_Vector */
951	//	SQDMULL2	//0x0F00B000,/* SQDMULL2 ARM64Op_sqdmull2_by_element_Vector */
952	//	SQDMULH	//0x0F00C000,/* SQDMULH ARM64Op_sqdmulh_by_element_Vector */
953	//	SQRDMULH	//0x0F00D000,/* SQRDMULH ARM64Op_sqrdmulh_by_element_Vector */
954	//	FMLA	//0x0F801000,/* FMLA ARM64Op_fmla_by_element_Vector */
955	//	FMLS	//0x0F805000,/* FMLS ARM64Op_fmls_by_element_Vector */
956	//	FMUL	//0x0F809000,/* FMUL ARM64Op_fmul_by_element_Vector */
957	//	MLA	//0x2F000000,/* MLA ARM64Op_mla_by_element */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
958	//	UMLAL	//0x2F002000,/*	UMLAL	ARM64Op_umlal_by_element */
959	//	UMLAL2	//0x2F002000,/*	UMLAL2	ARM64Op_umlal2_by_element */
960	//	MLS	//0x2F004000,/*	MLS	ARM64Op_mls_by_element */
961	//	UMLSL	//0x2F006000,/*	UMLSL	ARM64Op_umlsl_by_element */
962	//	UMLSL2	//0x2F006000,/*	UMLSL2	ARM64Op_umlsl2_by_element */
963	//	UMULL	//0x2F00A000,/*	UMULL	ARM64Op_umull_by_element */
964	//	UMULL2	//0x2F00A000,/*	UMULL2	ARM64Op_umull2_by_element */
965	//	FMULX	//0x2F809000,/*	FMULX	ARM64Op_fmulx_by_element_Vector */
966	//	**AdvSIMD modified immediate**	/* AdvSIMD modified immediate */		
967	//	MOVI	//0x0F000400,/*	MOVI	ARM64Op_movi_32_bit_shifted_immediate */
968	//	ORR	//0x0F001400,/*	ORR	ARM64Op_orr_vector_immediate_32_bit */
969	//	MOVI	//0x0F008400,/*	MOVI	ARM64Op_movi_16_bit_shifted_immediate */
970	//	ORR	//0x0F009400,/*	ORR	ARM64Op_orr_vector_immediate_16_bit */
971	//	MOVI	//0x0F00C400,/*	MOVI	ARM64Op_movi_32_bit_shifting_ones */
972	//	MOVI	//0x0F00E400,/*	MOVI	ARM64Op_movi_8_bit */
973	//	FMOV	//0x0F00F400,/*	FMOV	ARM64Op_fmov_vector_immediate_Single_precision
974	//	MVNI	//0x2F000400,/*	MVNI	ARM64Op_mvni_32_bit_shifted_immediate */
975	//	BIC	//0x2F001400,/*	BIC	ARM64Op_bic_vector_immediate_32_bit */
976	//	MVNI	//0x2F008400,/*	MVNI	ARM64Op_mvni_16_bit_shifted_immediate */
977	//	BIC	//0x2F009400,/*	BIC	ARM64Op_bic_vector_immediate_16_bit */
978	//	MVNI	//0x2F00C400,/*	MVNI	ARM64Op_mvni_32_bit_shifting_ones */
979	//	MOVI	//0x2F00E400,/*	MOVI	ARM64Op_movi_64_bit_scalar */
980	//	MOVI	//0x6F00E400,/*	MOVI	ARM64Op_movi_64_bit_vector */
981	//	FMOV	//0x6F00F400,/*	FMOV	ARM64Op_fmov_vector_immediate_Double_precision
982	//	**AdvSIMD shift by immediate**	/* AdvSIMD shift by immediate */		
983	//	SSHR	//0x0F000400,/*	SSHR	ARM64Op_sshr_Vector */
984	//	SSRA	//0x0F001400,/*	SSRA	ARM64Op_ssra_Vector */
985	//	SRSHR	//0x0F002400,/*	SRSHR	ARM64Op_srshr_Vector */
986	//	SRSRA	//0x0F003400,/*	SRSRA	ARM64Op_srsra_Vector */
987	//	SHL	//0x0F005400,/*	SHL	ARM64Op_shl_Vector */
988	//	SQSHL	//0x0F007400,/*	SQSHL	ARM64Op_sqshl_immediate_Vector */
989	//	SHRN	//0x0F008400,/*	SHRN	ARM64Op_shrn */
990	//	SHRN2	//0x0F008400,/*	SHRN2	ARM64Op_shrn2 */
991	//	RSHRN	//0x0F008C00,/*	RSHRN	ARM64Op_rshrn */

1	in_use	Opcode	//BINARY Opcode Opcodecomments
992	//	RSHRN2	//0x0F008C00,/* RSHRN2 ARM64Op_rshrn2 */
993	//	SQSHRN	//0x0F009400,/* SQSHRN ARM64Op_sqshrn_Vector */
994	//	SQSHRN2	//0x0F009400,/* SQSHRN2 ARM64Op_sqshrn2_Vector */
995	//	SQRSHRN	//0x0F009C00,/* SQRSHRN ARM64Op_sqrshrn_Vector */
996	//	SQRSHRN2	//0x0F009C00,/* SQRSHRN2 ARM64Op_sqrshrn2_Vector */
997	//	SSHLL	//0x0F00A400,/* SSHLL ARM64Op_sshll */
998	//	SSHLL2	//0x0F00A400,/* SSHLL2 ARM64Op_sshll2 */
999	//	SCVTF	//0x0F00E400,/* SCVTF ARM64Op_scvtf_vector_fixed_point_Vector */
1000	//	FCVTZS	//0x0F00FC00,/* FCVTZS ARM64Op_fcvtzs_vector_fixed_point_Vector */
1001	//	USHR	//0x2F000400,/* USHR ARM64Op_ushr_Vector */
1002	//	USRA	//0x2F001400,/* USRA ARM64Op_usra_Vector */
1003	//	URSHR	//0x2F002400,/* URSHR ARM64Op_urshr_Vector */
1004	//	URSRA	//0x2F003400,/* URSRA ARM64Op_ursra_Vector */
1005	//	SRI	//0x2F004400,/* SRI ARM64Op_sri_Vector */
1006	//	SLI	//0x2F005400,/* SLI ARM64Op_sli_Vector */
1007	//	SQSHLU	//0x2F006400,/* SQSHLU ARM64Op_sqshlu_Vector */
1008	//	UQSHL	//0x2F007400,/* UQSHL ARM64Op_uqshl_immediate_Vector */
1009	//	SQSHRUN	//0x2F008400,/* SQSHRUN ARM64Op_sqshrun_Vector */
1010	//	SQSHRUN2	//0x2F008400,/* SQSHRUN2 ARM64Op_sqshrun2_Vector */
1011	//	SQRSHRUN	//0x2F008C00,/* SQRSHRUN ARM64Op_sqrshrun_Vector */
1012	//	SQRSHRUN2	//0x2F008C00,/* SQRSHRUN2 ARM64Op_sqrshrun2_Vector */
1013	//	UQSHRN	//0x2F009400,/* UQSHRN ARM64Op_uqshrn_Vector */
1014	//	UQRSHRN	//0x2F009C00,/* UQRSHRN ARM64Op_uqrshrn_Vector */
1015	//	UQRSHRN2	//0x2F009C00,/* UQRSHRN2 ARM64Op_uqrshrn2_Vector */
1016	//	USHLL	//0x2F00A400,/* USHLL ARM64Op_ushll */
1017	//	USHLL2	//0x2F00A400,/* USHLL2 ARM64Op_ushll2 */
1018	//	UCVTF	//0x2F00E400,/* UCVTF ARM64Op_ucvtf_vector_fixed_point_Vector */
1019	//	FCVTZU	//0x2F00FC00,/* FCVTZU ARM64Op_fcvtzu_vector_fixed_point_Vector */
1020	//	**AdvSIMD TBL/TBX**	/* AdvSIMD TBL/TBX */
1021	//	TBL	//0x0E000000,/* TBL ARM64Op_tbl_Single_register_table */
1022	//	TBX	//0x0E001000,/* TBX ARM64Op_tbx_Single_register_table */
1023	//	TBL	//0x0E002000,/* TBL ARM64Op_tbl_Two_register_table */
1024	//	TBX	//0x0E003000,/* TBX ARM64Op_tbx_Two_register_table */
1025	//	TBL	//0x0E004000,/* TBL ARM64Op_tbl_Three_register_table */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1026	//	TBX	//0x0E005000,/* TBX	ARM64Op_tbx_Three_register_table */	
1027	//	TBL	//0x0E006000,/* TBL	ARM64Op_tbl_Four_register_table */	
1028	//	TBX	//0x0E007000,/* TBX	ARM64Op_tbx_Four_register_table */	
1029	//	**AdvSIMD ZIP/UZP/TRN**	/* AdvSIMD ZIP/UZP/TRN */		
1030	//	UZP1	//0x0E001800,/* UZP1	ARM64Op_uzp1 */	
1031	//	TRN1	//0x0E002800,/* TRN1	ARM64Op_trn1 */	
1032	//	ZIP1	//0x0E003800,/* ZIP1	ARM64Op_zip1 */	
1033	//	UZP2	//0x0E005800,/* UZP2	ARM64Op_uzp2 */	
1034	//	TRN2	//0x0E006800,/* TRN2	ARM64Op_trn2 */	
1035	//	ZIP2	//0x0E007800,/* ZIP2	ARM64Op_zip2 */	
1036	//	**AdvSIMD EXT**	/* AdvSIMD EXT */		
1037	//	EXT	//0x2E000000,/* EXT	ARM64Op_ext */	
1038	//	**Loads and stores**	/* Loads and stores */		
1039	//	**AdvSIMD load/store multiple structures**	/* AdvSIMD load/store multiple structures */		
1040	//	ST4	//0x0C000000,/* ST4	ARM64Op_st4_multiple_structures_No_offset */	
1041	//	ST1	//0x0C002000,/* ST1	ARM64Op_st1_multiple_structures_Four_registers */	
1042	//	ST3	//0x0C004000,/* ST3	ARM64Op_st3_multiple_structures_No_offset */	
1043	//	ST1	//0x0C006000,/* ST1	ARM64Op_st1_multiple_structures_Three_registers */	
1044	//	ST1	//0x0C007000,/* ST1	ARM64Op_st1_multiple_structures_One_register */	
1045	//	ST2	//0x0C008000,/* ST2	ARM64Op_st2_multiple_structures_No_offset */	
1046	//	ST1	//0x0C00A000,/* ST1	ARM64Op_st1_multiple_structures_Two_registers */	
1047	//	LD4	//0x0C400000,/* LD4	ARM64Op_ld4_multiple_structures_No_offset */	
1048	//	LD1	//0x0C402000,/* LD1	ARM64Op_ld1_multiple_structures_Four_registers */	
1049	//	LD3	//0x0C404000,/* LD3	ARM64Op_ld3_multiple_structures_No_offset */	
1050	//	LD1	//0x0C406000,/* LD1	ARM64Op_ld1_multiple_structures_Three_registers */	
1051	//	LD1	//0x0C407000,/* LD1	ARM64Op_ld1_multiple_structures_One_register */	
1052	//	LD2	//0x0C408000,/* LD2	ARM64Op_ld2_multiple_structures_No_offset */	
1053	//	LD1	//0x0C40A000,/* LD1	ARM64Op_ld1_multiple_structures_Two_registers */	
1054	//	**AdvSIMD load/store multiple structures (pc**	/* AdvSIMD load/store multiple structures (post-indexed) */		
1055	//	ST4	//0x0C800000,/* ST4	ARM64Op_st4_multiple_structures_Register_offsetRm !	
1056	//	ST1	//0x0C802000,/* ST1	ARM64Op_st1_multiple_structures_Four_registers_regi	
1057	//	ST3	//0x0C804000,/* ST3	ARM64Op_st3_multiple_structures_Register_offsetRm !	
1058	//	ST1	//0x0C806000,/* ST1	ARM64Op_st1_multiple_structures_Three_registers_reg	
1059	//	ST1	//0x0C807000,/* ST1	ARM64Op_st1_multiple_structures_One_register_regist	

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1060	//	ST2	//0x0C808000,/*	ST2	ARM64Op_st2_multiple_structures_Register_offsetRm !
1061	//	ST1	//0x0C80A000,/*	ST1	ARM64Op_st1_multiple_structures_Two_registers_regi
1062	//	ST4	//0x0C9F0000,/*	ST4	ARM64Op_st4_multiple_structures_Immediate_offset */
1063	//	ST1	//0x0C9F2000,/*	ST1	ARM64Op_st1_multiple_structures_Four_registers_imm
1064	//	ST3	//0x0C9F4000,/*	ST3	ARM64Op_st3_multiple_structures_Immediate_offset */
1065	//	ST1	//0x0C9F6000,/*	ST1	ARM64Op_st1_multiple_structures_Three_registers_im
1066	//	ST1	//0x0C9F7000,/*	ST1	ARM64Op_st1_multiple_structures_One_register_imme
1067	//	ST2	//0x0C9F8000,/*	ST2	ARM64Op_st2_multiple_structures_Immediate_offset */
1068	//	ST1	//0x0C9FA000,/*	ST1	ARM64Op_st1_multiple_structures_Two_registers_imm
1069	//	LD4	//0x0CC00000,/*	LD4	ARM64Op_ld4_multiple_structures_Register_offsetRm
1070	//	LD1	//0x0CC02000,/*	LD1	ARM64Op_ld1_multiple_structures_Four_registers_regi
1071	//	LD3	//0x0CC04000,/*	LD3	ARM64Op_ld3_multiple_structures_Register_offsetRm
1072	//	LD1	//0x0CC06000,/*	LD1	ARM64Op_ld1_multiple_structures_Three_registers_re
1073	//	LD1	//0x0CC07000,/*	LD1	ARM64Op_ld1_multiple_structures_One_register_regis
1074	//	LD2	//0x0CC08000,/*	LD2	ARM64Op_ld2_multiple_structures_Register_offsetRm
1075	//	LD1	//0x0CC0A000,/*	LD1	ARM64Op_ld1_multiple_structures_Two_registers_regi
1076	//	LD4	//0x0CDF0000,/*	LD4	ARM64Op_ld4_multiple_structures_Immediate_offset */
1077	//	LD1	//0x0CDF2000,/*	LD1	ARM64Op_ld1_multiple_structures_Four_registers_imn
1078	//	LD3	//0x0CDF4000,/*	LD3	ARM64Op_ld3_multiple_structures_Immediate_offset */
1079	//	LD1	//0x0CDF6000,/*	LD1	ARM64Op_ld1_multiple_structures_Three_registers_im
1080	//	LD1	//0x0CDF7000,/*	LD1	ARM64Op_ld1_multiple_structures_One_register_imme
1081	//	LD2	//0x0CDF8000,/*	LD2	ARM64Op_ld2_multiple_structures_Immediate_offset */
1082	//	LD1	//0x0CDFA000,/*	LD1	ARM64Op_ld1_multiple_structures_Two_registers_imn
1083	//	**AdvSIMD load/store single structure**	/* AdvSIMD load/store single structure */		
1084	//	ST1	//0x0D000000,/*	ST1	ARM64Op_st1_single_structure_8_bit */
1085	//	ST3	//0x0D002000,/*	ST3	ARM64Op_st3_single_structure_8_bit */
1086	//	ST1	//0x0D004000,/*	ST1	ARM64Op_st1_single_structure_16_bit */
1087	//	ST3	//0x0D006000,/*	ST3	ARM64Op_st3_single_structure_16_bit */
1088	//	ST1	//0x0D008000,/*	ST1	ARM64Op_st1_single_structure_32_bit */
1089	//	ST1	//0x0D008400,/*	ST1	ARM64Op_st1_single_structure_64_bit */
1090	//	ST3	//0x0D00A000,/*	ST3	ARM64Op_st3_single_structure_32_bit */
1091	//	ST3	//0x0D00A400,/*	ST3	ARM64Op_st3_single_structure_64_bit */
1092	//	ST2	//0x0D200000,/*	ST2	ARM64Op_st2_single_structure_8_bit */
1093	//	ST4	//0x0D202000,/*	ST4	ARM64Op_st4_single_structure_8_bit */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1094	//	ST2	//0x0D204000,/*	ST2	ARM64Op_st2_single_structure_16_bit */
1095	//	ST4	//0x0D206000,/*	ST4	ARM64Op_st4_single_structure_16_bit */
1096	//	ST2	//0x0D208000,/*	ST2	ARM64Op_st2_single_structure_32_bit */
1097	//	ST2	//0x0D208400,/*	ST2	ARM64Op_st2_single_structure_64_bit */
1098	//	ST4	//0x0D20A000,/*	ST4	ARM64Op_st4_single_structure_32_bit */
1099	//	ST4	//0x0D20A400,/*	ST4	ARM64Op_st4_single_structure_64_bit */
1100	//	LD1	//0x0D400000,/*	LD1	ARM64Op_ld1_single_structure_8_bit */
1101	//	LD3	//0x0D402000,/*	LD3	ARM64Op_ld3_single_structure_8_bit */
1102	//	LD1	//0x0D404000,/*	LD1	ARM64Op_ld1_single_structure_16_bit */
1103	//	LD3	//0x0D406000,/*	LD3	ARM64Op_ld3_single_structure_16_bit */
1104	//	LD1	//0x0D408000,/*	LD1	ARM64Op_ld1_single_structure_32_bit */
1105	//	LD1	//0x0D408400,/*	LD1	ARM64Op_ld1_single_structure_64_bit */
1106	//	LD3	//0x0D40A000,/*	LD3	ARM64Op_ld3_single_structure_32_bit */
1107	//	LD3	//0x0D40A400,/*	LD3	ARM64Op_ld3_single_structure_64_bit */
1108	//	LD1R	//0x0D40C000,/*	LD1R	ARM64Op_ld1r_No_offset */
1109	//	LD3R	//0x0D40E000,/*	LD3R	ARM64Op_ld3r_No_offset */
1110	//	LD2	//0x0D600000,/*	LD2	ARM64Op_ld2_single_structure_8_bit */
1111	//	LD4	//0x0D602000,/*	LD4	ARM64Op_ld4_single_structure_8_bit */
1112	//	LD2	//0x0D604000,/*	LD2	ARM64Op_ld2_single_structure_16_bit */
1113	//	LD4	//0x0D606000,/*	LD4	ARM64Op_ld4_single_structure_16_bit */
1114	//	LD2	//0x0D608000,/*	LD2	ARM64Op_ld2_single_structure_32_bit */
1115	//	LD2	//0x0D608400,/*	LD2	ARM64Op_ld2_single_structure_64_bit */
1116	//	LD4	//0x0D60A000,/*	LD4	ARM64Op_ld4_single_structure_32_bit */
1117	//	LD4	//0x0D60A400,/*	LD4	ARM64Op_ld4_single_structure_64_bit */
1118	//	LD2R	//0x0D60C000,/*	LD2R	ARM64Op_ld2r_No_offset */
1119	//	LD4R	//0x0D60E000,/*	LD4R	ARM64Op_ld4r_No_offset */
1120	//	**AdvSIMD load/store single structure (post-**	/* AdvSIMD load/store single structure (post-indexed) */		
1121	//	ST1	//0x0D800000,/*	ST1	ARM64Op_st1_single_structure_8_bit_register_offsetRr
1122	//	ST3	//0x0D802000,/*	ST3	ARM64Op_st3_single_structure_8_bit_register_offsetRr
1123	//	ST1	//0x0D804000,/*	ST1	ARM64Op_st1_single_structure_16_bit_register_offsetF
1124	//	ST3	//0x0D806000,/*	ST3	ARM64Op_st3_single_structure_16_bit_register_offsetF
1125	//	ST1	//0x0D808000,/*	ST1	ARM64Op_st1_single_structure_32_bit_register_offsetF
1126	//	ST1	//0x0D808400,/*	ST1	ARM64Op_st1_single_structure_64_bit_register_offsetF
1127	//	ST3	//0x0D80A000,/*	ST3	ARM64Op_st3_single_structure_32_bit_register_offsetF

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1128	//	ST3	//0x0D80A400,/*	ST3	ARM64Op_st3_single_structure_64_bit_register_offsetF
1129	//	ST1	//0x0D9F0000,/*	ST1	ARM64Op_st1_single_structure_8_bit_immediate_offse
1130	//	ST3	//0x0D9F2000,/*	ST3	ARM64Op_st3_single_structure_8_bit_immediate_offse
1131	//	ST1	//0x0D9F4000,/*	ST1	ARM64Op_st1_single_structure_16_bit_immediate_offs
1132	//	ST3	//0x0D9F6000,/*	ST3	ARM64Op_st3_single_structure_16_bit_immediate_offs
1133	//	ST1	//0x0D9F8000,/*	ST1	ARM64Op_st1_single_structure_32_bit_immediate_offs
1134	//	ST1	//0x0D9F8400,/*	ST1	ARM64Op_st1_single_structure_64_bit_immediate_offs
1135	//	ST3	//0x0D9FA000,/*	ST3	ARM64Op_st3_single_structure_32_bit_immediate_offs
1136	//	ST3	//0x0D9FA400,/*	ST3	ARM64Op_st3_single_structure_64_bit_immediate_offs
1137	//	ST2	//0x0DA00000,/*	ST2	ARM64Op_st2_single_structure_8_bit_register_offsetRı
1138	//	ST4	//0x0DA02000,/*	ST4	ARM64Op_st4_single_structure_8_bit_register_offsetRı
1139	//	ST2	//0x0DA04000,/*	ST2	ARM64Op_st2_single_structure_16_bit_register_offsetF
1140	//	ST4	//0x0DA06000,/*	ST4	ARM64Op_st4_single_structure_16_bit_register_offsetF
1141	//	ST2	//0x0DA08000,/*	ST2	ARM64Op_st2_single_structure_32_bit_register_offsetF
1142	//	ST2	//0x0DA08400,/*	ST2	ARM64Op_st2_single_structure_64_bit_register_offsetF
1143	//	ST4	//0x0DA0A000,/*	ST4	ARM64Op_st4_single_structure_32_bit_register_offsetF
1144	//	ST4	//0x0DA0A400,/*	ST4	ARM64Op_st4_single_structure_64_bit_register_offsetF
1145	//	ST2	//0x0DBF0000,/*	ST2	ARM64Op_st2_single_structure_8_bit_immediate_offse
1146	//	ST4	//0x0DBF2000,/*	ST4	ARM64Op_st4_single_structure_8_bit_immediate_offse
1147	//	ST2	//0x0DBF4000,/*	ST2	ARM64Op_st2_single_structure_16_bit_immediate_offs
1148	//	ST4	//0x0DBF6000,/*	ST4	ARM64Op_st4_single_structure_16_bit_immediate_offs
1149	//	ST2	//0x0DBF8000,/*	ST2	ARM64Op_st2_single_structure_32_bit_immediate_offs
1150	//	ST2	//0x0DBF8400,/*	ST2	ARM64Op_st2_single_structure_64_bit_immediate_offs
1151	//	ST4	//0x0DBFA000,/*	ST4	ARM64Op_st4_single_structure_32_bit_immediate_offs
1152	//	ST4	//0x0DBFA400,/*	ST4	ARM64Op_st4_single_structure_64_bit_immediate_offs
1153	//	LD1	//0x0DC00000,/*	LD1	ARM64Op_ld1_single_structure_8_bit_register_offsetRı
1154	//	LD3	//0x0DC02000,/*	LD3	ARM64Op_ld3_single_structure_8_bit_register_offsetRı
1155	//	LD1	//0x0DC04000,/*	LD1	ARM64Op_ld1_single_structure_16_bit_register_offsetF
1156	//	LD3	//0x0DC06000,/*	LD3	ARM64Op_ld3_single_structure_16_bit_register_offsetF
1157	//	LD1	//0x0DC08000,/*	LD1	ARM64Op_ld1_single_structure_32_bit_register_offsetF
1158	//	LD1	//0x0DC08400,/*	LD1	ARM64Op_ld1_single_structure_64_bit_register_offsetF
1159	//	LD3	//0x0DC0A000,/*	LD3	ARM64Op_ld3_single_structure_32_bit_register_offsetI
1160	//	LD3	//0x0DC0A400,/*	LD3	ARM64Op_ld3_single_structure_64_bit_register_offsetI
1161	//	LD1R	//0x0DC0C000,/*	LD1R	ARM64Op_ld1r_Register_offsetRm != 11111 */

1	in_use	Opcode	//BINARY	Opcode	Opcodecomments
1162	//	LD3R	//0x0DC0E000,/*	LD3R	ARM64Op_ld3r_Register_offsetRm != 11111 */
1163	//	LD1	//0x0DDF0000,/*	LD1	ARM64Op_ld1_single_structure_8_bit_immediate_offse
1164	//	LD3	//0x0DDF2000,/*	LD3	ARM64Op_ld3_single_structure_8_bit_immediate_offse
1165	//	LD1	//0x0DDF4000,/*	LD1	ARM64Op_ld1_single_structure_16_bit_immediate_offs
1166	//	LD3	//0x0DDF6000,/*	LD3	ARM64Op_ld3_single_structure_16_bit_immediate_offs
1167	//	LD1	//0x0DDF8000,/*	LD1	ARM64Op_ld1_single_structure_32_bit_immediate_offs
1168	//	LD1	//0x0DDF8400,/*	LD1	ARM64Op_ld1_single_structure_64_bit_immediate_offs
1169	//	LD3	//0x0DDFA000,/*	LD3	ARM64Op_ld3_single_structure_32_bit_immediate_offs
1170	//	LD3	//0x0DDFA400,/*	LD3	ARM64Op_ld3_single_structure_64_bit_immediate_offs
1171	//	LD1R	//0x0DDFC000,/*	LD1R	ARM64Op_ld1r_Immediate_offset */
1172	//	LD3R	//0x0DDFE000,/*	LD3R	ARM64Op_ld3r_Immediate_offset */
1173	//	LD2	//0x0DE00000,/*	LD2	ARM64Op_ld2_single_structure_8_bit_register_offsetRı
1174	//	LD4	//0x0DE02000,/*	LD4	ARM64Op_ld4_single_structure_8_bit_register_offsetRı
1175	//	LD2	//0x0DE04000,/*	LD2	ARM64Op_ld2_single_structure_16_bit_register_offsetF
1176	//	LD4	//0x0DE06000,/*	LD4	ARM64Op_ld4_single_structure_16_bit_register_offsetF
1177	//	LD2	//0x0DE08000,/*	LD2	ARM64Op_ld2_single_structure_32_bit_register_offsetF
1178	//	LD2	//0x0DE08400,/*	LD2	ARM64Op_ld2_single_structure_64_bit_register_offsetF
1179	//	LD4	//0x0DE0A000,/*	LD4	ARM64Op_ld4_single_structure_32_bit_register_offsetF
1180	//	LD4	//0x0DE0A400,/*	LD4	ARM64Op_ld4_single_structure_64_bit_register_offsetF
1181	//	LD2R	//0x0DE0C000,/*	LD2R	ARM64Op_ld2r_Register_offsetRm != 11111 */
1182	//	LD4R	//0x0DE0E000,/*	LD4R	ARM64Op_ld4r_Register_offsetRm != 11111 */
1183	//	LD2	//0x0DFF0000,/*	LD2	ARM64Op_ld2_single_structure_8_bit_immediate_offse
1184	//	LD4	//0x0DFF2000,/*	LD4	ARM64Op_ld4_single_structure_8_bit_immediate_offse
1185	//	LD2	//0x0DFF4000,/*	LD2	ARM64Op_ld2_single_structure_16_bit_immediate_offs
1186	//	LD4	//0x0DFF6000,/*	LD4	ARM64Op_ld4_single_structure_16_bit_immediate_offs
1187	//	LD2	//0x0DFF8000,/*	LD2	ARM64Op_ld2_single_structure_32_bit_immediate_offs
1188	//	LD2	//0x0DFF8400,/*	LD2	ARM64Op_ld2_single_structure_64_bit_immediate_offs
1189	//	LD4	//0x0DFFA000,/*	LD4	ARM64Op_ld4_single_structure_32_bit_immediate_offs
1190	//	LD4	//0x0DFFA400,/*	LD4	ARM64Op_ld4_single_structure_64_bit_immediate_offs
1191	//	LD2R	//0x0DFFC000,/*	LD2R	ARM64Op_ld2r_Immediate_offset */
1192	//	LD4R	//0x0DFFE000,/*	LD4R	ARM64Op_ld4r_Immediate_offset */