					Binary Encoding											Assembly	S Update	Op.	Op. (continued)
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0	0	1	0	0	Rd			im8								MOVS Rd, #<im8>	N Z	Move	Rd=ExZ(<im8>)
0	1	0	0	0	1	1	0	0	Rm				Rd			MOV Rd, Rm	-		Rd=Rm
0	0	0	1	1	1	0	im3			Rn			Rd			ADDS Rd, Rn, #<im3>	N Z C V	Add	Rd=Rn+ExZ(<im3>)
0	0	0	1	1	0	0	Rm			Rn			Rd			ADDS Rd, Rn, Rm	N Z C V		Rd=Rn+Rm
1	0	1	1	0	0	0	0	0	im7							ADD SP, SP, #<im7>	-		SP=SP+ExZ(<im7>)
0	0	0	1	1	0	1	Rm			Rn			Rd			SUBS Rd, Rn, Rm	N Z C V	Subtract	Rd=Rn+~(Rm)+1
0	0	0	1	1	1	1	im3			Rn			Rd			SUBS Rd, Rn, #<im3>	N Z C V		Rd=Rn+~ExZ(<im3>)+1
1	0	1	1	0	0	0	0	1	im7							SUB SP, SP, #<im7>	-		SP=Sp+~ExZ(<im7>)+1
0	1	0	0	0	0	1	0	1	0	Rm			Rn			CMP Rn, Rm	N Z C V	Compare	x=Rn+~(Rm)+1 (x is unused)
0	1	0	0	0	0	0	0	0	0	Rm			Rd			ANDS Rd, Rm	N Z	Logical	Rd=Rm&Rd
0	1	0	0	0	0	0	0	0	1	Rm			Rd			EORS Rd, Rm	N Z		Rd=Rm⊕Rd
0	1	0	0	0	0	1	1	0	0	Rm			Rd			ORRS Rd, Rm	N Z		Rd=Rm|Rd
0	1	0	0	0	0	1	1	1	1	Rm			Rd			MVNS Rd, Rm	N Z		Rd=~(Rm)
0	1	0	0	0	0	0	0	1	0	Rm			Rd			LSLS Rd, Rd, Rm	N Z C	Rotate	Rd=PadZ(Rd<<Rm)
0	1	0	0	0	0	0	0	1	1	Rm			Rd			LSRS Rd, Rd, Rm	N Z C		Rd=PadZ(Rd>>Rm)
0	1	0	0	0	0	0	1	0	0	Rm			Rd			ASRS Rd, Rd, Rm	N Z C		Rd=PadS(Rd>>Rm)
0	1	0	0	0	0	0	1	1	1	Rm			Rd			RORS Rd, Rd, Rm	N Z C		Rd=PadCircle(Rd>>Rm)
0	1	1	0	0	im5					Rn			Rd			STR Rd, [Rn, #<im5>]	-	Store	Mem[Rn+ExZ(<im5>)]=Rd
0	1	1	0	1	im5					Rn			Rd			LDR Rd, [Rn, #<im5>]	-	Load	Rd=Mem[Rn+ExZ(<im5>)]
1	1	0	1	cond				im8								B<cc> <label>	-	Branch	PC=<cc>?PC+ExS(<im8>):PC+1
1	1	1	0	0	im11											B <label>	-		PC=PC+ExS(<im11>)
0	1	0	0	0	1	0	1	0	0	im6						BL <label>	-		LR=PC+1; PC=PC+ExS(<im6>)
0	1	0	0	0	1	1	1	0	Rm			0	0	0		BX Rm	-		PC=Rm
1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	NOOP	-		Stall for one cycle
0	1	1	1	0	0	0	Rm			Rn			Rd			FADD	-	Floating Point	Rd=Rn+Rm
0	1	1	1	0	0	1	Rm			Rn			Rd			FSUB	-		Rd=Rn+~(Rm)+1
0	1	1	1	0	1	0	Rm			Rn			Rd			FMUL	-		Rd=Rn * Rm
0	1	1	1	0	1	1	Rm			Rn			Rd			FDIV	-		Rd=Rn ÷ Rm
0	1	1	1	1	0	0	0	0	0	Rm			Rn			FCMP	N Z C V		x=Rn-Rm (x is unused)