Opcodes for the ISA of Spiderchip64

Version 0.1

Aurélien Casteilla

14-05-2021

Table 1: Memory operations

Mnemonic	Arguments	Condition 31 - 28	Memory ? 27	load ? 26	size 25 - 24	sign 23	mode 22 - 21	-S suffix 20	(other) 19 - 15	(index) 14 - 10	(base) 9 - 5	register 4 - 0
[cc.]LDW[S]	Rd, offset	CCCC	1	1	11	0	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LDW[S]	Rd, Rx, offset	CCCC	1	1	11	0	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LDW[S]	Rd, Rx, Ry << shift	CCCC	1	1	11	0	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LDW[S]	Rd, Rx, Ry+	CCCC	1	1	11	0	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LDW[S]	Rd, Rx, Ry-	CCCC	1	1	11	0	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LDW[S]	Rd, Rx, +Ry	CCCC	1	1	11	0	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LDW[S]	Rd, Rx, -Ry	CCCC	1	1	11	0	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LWS[S]	Rd, offset	CCCC	1	1	10	1	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LWS[S]	Rd, Rx, offset	CCCC	1	1	10	1	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LWS[S]	Rd, Rx, Ry << shift	CCCC	1	1	10	1	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LWS[S]	Rd, Rx, Ry+	CCCC	1	1	10	1	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LWS[S]	Rd, Rx, Ry-	CCCC	1	1	10	1	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LWS[S]	Rd, Rx, +Ry	CCCC	1	1	10	1	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LWS[S]	Rd, Rx, -Ry	CCCC	1	1	10	1	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LWU[S]	Rd, offset	CCCC	1	1	10	0	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LWU[S]	Rd, Rx, offset	CCCC	1	1	10	0	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LWU[S]	Rd, Rx, Ry << shift	CCCC	1	1	10	0	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LWU[S]	Rd, Rx, Ry+	CCCC	1	1	10	0	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LWU[S]	Rd, Rx, Ry-	CCCC	1	1	10	0	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LWU[S]	Rd, Rx, +Ry	CCCC	1	1	10	0	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LWU[S]	Rd, Rx, -Ry	CCCC	1	1	10	0	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LHS[S]	Rd, offset	CCCC	1	1	01	1	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LHS[S]	Rd, Rx, offset	CCCC	1	1	01	1	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LHS[S]	Rd, Rx, Ry << shift	CCCC	1	1	01	1	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LHS[S]	Rd, Rx, Ry+	CCCC	1	1	01	1	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LHS[S]	Rd, Rx, Ry-	CCCC	1	1	01	1	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LHS[S]	Rd, Rx, +Ry	CCCC	1	1	01	1	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LHS[S]	Rd, Rx, -Ry	CCCC	1	1	01	1	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LHU[S]	Rd, offset	CCCC	1	1	01	0	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LHU[S]	Rd, Rx, offset	CCCC	1	1	01	0	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LHU[S]	Rd, Rx, Ry << shift	CCCC	1	1	01	0	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LHU[S]	Rd, Rx, Ry+	CCCC	1	1	01	0	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LHU[S]	Rd, Rx, Ry-	CCCC	1	1	01	0	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LHU[S]	Rd, Rx, +Ry	CCCC	1	1	01	0	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LHU[S]	Rd, Rx, -Ry	CCCC	1	1	01	0	11	S	11XXX	yyyyy	xxxxx	ddddd

Mnemonic	Arguments	Condition	Memory ?	load ?	size	sign	mode	-S suffix	(other)	(index)	(base)	register
[cc.]LBS[S]	Rd, offset	CCCC	1	1	00	1	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LBS[S]	Rd, Rx, offset	CCCC	1	1	00	1	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LBS[S]	Rd, Rx, Ry << shift	CCCC	1	1	00	1	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LBS[S]	Rd, Rx, Ry+	CCCC	1	1	00	1	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LBS[S]	Rd, Rx, Ry-	CCCC	1	1	00	1	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LBS[S]	Rd, Rx, +Ry	CCCC	1	1	00	1	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LBS[S]	Rd, Rx, -Ry	CCCC	1	1	00	1	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LBU[S]	Rd, offset	CCCC	1	1	00	0	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LBU[S]	Rd, Rx, offset	CCCC	1	1	00	0	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LBU[S]	Rd, Rx, Ry << shift	CCCC	1	1	00	0	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LBU[S]	Rd, Rx, Ry+	CCCC	1	1	00	0	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LBU[S]	Rd, Rx, Ry-	CCCC	1	1	00	0	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LBU[S]	Rd, Rx, +Ry	CCCC	1	1	00	0	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LBU[S]	Rd, Rx, -Ry	CCCC	1	1	00	0	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]STD[S]	Rd, offset	CCCC	1	0	11	0	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]STD[S]	Rd, Rx, offset	CCCC	1	0	11	0	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]STD[S]	Rd, Rx, Ry << shift	CCCC	1	0	11	0	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]STD[S]	Rd, Rx, Ry+	CCCC	1	0	11	0	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]STD[S]	Rd, Rx, Ry-	CCCC	1	0	11	0	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]STD[S]	Rd, Rx, +Ry	CCCC	1	0	11	0	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]STD[S]	Rd, Rx, -Ry	CCCC	1	0	11	0	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]STW[S]	Rd, offset	CCCC	1	0	10	X	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]STW[S]	Rd, Rx, offset	CCCC	1	0	10	X	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]STW[S]	Rd, Rx, Ry << shift	CCCC	1	0	10	X	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]STW[S]	Rd, Rx, Ry+	CCCC	1	0	10	X	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]STW[S]	Rd, Rx, Ry-	CCCC	1	0	10	X	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]STW[S]	Rd, Rx, +Ry	CCCC	1	0	10	X	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]STW[S]	Rd, Rx, -Ry	CCCC	1	0	10	X	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]STH[S]	Rd, offset	CCCC	1	0	01	X	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]STH[S]	Rd, Rx, offset	CCCC	1	0	01	X	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]STH[S]	Rd, Rx, Ry << shift	CCCC	1	0	01	X	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]STH[S]	Rd, Rx, Ry+	CCCC	1	0	01	X	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]STH[S]	Rd, Rx, Ry-	CCCC	1	0	01	X	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]STH[S]	Rd, Rx, +Ry	CCCC	1	0	01	X	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]STH[S]	Rd, Rx, -Ry	CCCC	1	0	01	X	11	S	11XXX	yyyyy	xxxxx	ddddd

Mnemonic	Arguments	Condition	Memory ?	load ?	size	sign	mode	-S suffix	(other)	(index)	(base)	register
[cc.]STB[S]	Rd, offset	CCCC	1	0	00	X	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]STB[S]	Rd, Rx, offset	CCCC	1	0	00	X	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]STB[S]	Rd, Rx, Ry << shift	CCCC	1	0	00	X	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]STB[S]	Rd, Rx, Ry+	CCCC	1	0	00	X	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]STB[S]	Rd, Rx, Ry-	CCCC	1	0	00	X	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]STB[S]	Rd, Rx, +Ry	CCCC	1	0	00	X	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]STB[S]	Rd, Rx, -Ry	CCCC	1	0	00	X	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]LLD[S]	Rd, offset	CCCC	1	1	11	1	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]LLD[S]	Rd, Rx, offset	CCCC	1	1	11	1	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]LLD[S]	Rd, Rx, Ry << shift	CCCC	1	1	11	1	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]LLD[S]	Rd, Rx, Ry+	CCCC	1	1	11	1	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]LLD[S]	Rd, Rx, Ry-	CCCC	1	1	11	1	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]LLD[S]	Rd, Rx, +Ry	CCCC	1	1	11	1	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]LLD[S]	Rd, Rx, Ry+	CCCC	1	1	11	1	11	S	11XXX	yyyyy	xxxxx	ddddd
[cc.]SCD[S]	Rd, offset	CCCC	1	0	11	1	00	S	rrrrr	rrrrr	rrrrr	ddddd
[cc.]SCD[S]	Rd, Rx, offset	CCCC	1	0	11	1	01	S	rrrrr	rrrrr	xxxxx	ddddd
[cc.]SCD[S]	Rd, Rx, Ry << shift	CCCC	1	0	11	1	10	S	sssss	yyyyy	xxxxx	ddddd
[cc.]SCD[S]	Rd, Rx, Ry+	CCCC	1	0	11	1	11	S	00XXX	yyyyy	xxxxx	ddddd
[cc.]SCD[S]	Rd, Rx, Ry-	CCCC	1	0	11	1	11	S	01XXX	yyyyy	xxxxx	ddddd
[cc.]SCD[S]	Rd, Rx, +Ry	CCCC	1	0	11	1	11	S	10XXX	yyyyy	xxxxx	ddddd
[cc.]SCD[S]	Rd, Rx, -Ry	CCCC	1	0	11	1	11	S	11XXX	yyyyy	xxxxx	ddddd

Notes :

- "cc." mean any condition and are associated with the bits CCCC (bit 31 to bit 28). If there are not any condition, CCCC bits are 0000 and the instruction is always executed.
- "X" mean either 0 or 1
- "s" are the bits used for shifted operands
- "x" are the bits used for the base register
- "y" are the bits used for the index register
- "r" bits are bits used to specify an offset
- if the suffix S is added, the S bit is 1. Otherwise, it's 0.

Table 2: Register operations

Mnemonic	Arguments	Condition 31 - 28	Memory ? 27	operation 26 - 21	-S suffix 20	mode 19	(other) 18 - 15	(source2) 14 - 10	(source1) 9 - 5	destination 4 - 0
[cc.]ADD[S]	Rd, Rx, #imm	CCCC	0	000011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADD[S]	Rd, Rx, Ry << shift	CCCC	0	000011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUB[S]	Rd, Rx, #imm	CCCC	0	000111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUB[S]	Rd, Rx, Ry << shift	CCCC	0	000111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]AND[S]	Rd, Rx, #imm	CCCC	0	100011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]AND[S]	Rd, Rx, Ry << shift	CCCC	0	100011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ORR[S]	Rd, Rx, #imm	CCCC	0	100111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ORR[S]	Rd, Rx, Ry << shift	CCCC	0	100111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]EOR[S]	Rd, Rx, #imm	CCCC	0	101011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]EOR[S]	Rd, Rx, Ry << shift	CCCC	0	101011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASL[S]	Rd, Rx, #imm	CCCC	0	001011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASL[S]	Rd, Rx, Ry << shift	CCCC	0	001011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]LSR[S]	Rd, Rx, #imm	CCCC	0	010011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]LSR[S]	Rd, Rx, Ry << shift	CCCC	0	010011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASR[S]	Rd, Rx, #imm	CCCC	0	001111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASR[S]	Rd, Rx, Ry << shift	CCCC	0	001111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ROL[S]	Rd, Rx, #imm	CCCC	0	011011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROL[S]	Rd, Rx, Ry << shift	CCCC	0	011011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ROR[S]	Rd, Rx, #imm	CCCC	0	010111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROR[S]	Rd, Rx, Ry << shift	CCCC	0	010111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]MUL[S]	Rd, Rx, #imm	CCCC	0	101111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]MUL[S]	Rd, Rx, Ry << shift	CCCC	0	101111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ADC[S]	Rd, Rx, #imm	CCCC	0	011100	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADC[S]	Rd, Rx, Ry << shift	CCCC	0	011100	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SBC[S]	Rd, Rx, #imm	CCCC	0	011101	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SBC[S]	Rd, Rx, Ry << shift	CCCC	0	011101	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RLC[S]	Rd, Rx, #imm	CCCC	0	011111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RLC[S]	Rd, Rx, Ry << shift	CCCC	0	011111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RRC[S]	Rd, Rx, #imm	CCCC	0	011110	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RRC[S]	Rd, Rx, Ry << shift	CCCC	0	011110	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ADDD[S]	Rd, Rx, #imm	CCCC	0	000011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADDD[S]	Rd, Rx, Ry << shift	CCCC	0	000011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ADDW[S]	Rd, Rx, #imm	CCCC	0	000010	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADDW[S]	Rd, Rx, Ry << shift	CCCC	0	000010	S	1	ssss	yyyyy	xxxxx	ddddd

Mnemonic	Arguments	Condition	Memory ?	operation	-S suffix	mode	(other)	(source2)	(source1)	destination
[cc.]ADDH[S]	Rd, Rx, #imm	CCCC	0	000001	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADDH[S]	Rd, Rx, Ry << shift	CCCC	0	000001	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ADDB[S]	Rd, Rx, #imm	CCCC	0	000000	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ADDB[S]	Rd, Rx, Ry << shift	CCCC	0	000000	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUBD[S]	Rd, Rx, #imm	CCCC	0	000111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUBD[S]	Rd, Rx, Ry << shift	CCCC	0	000111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUBW[S]	Rd, Rx, #imm	CCCC	0	000110	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUBW[S]	Rd, Rx, Ry << shift	CCCC	0	000110	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUBH[S]	Rd, Rx, #imm	CCCC	0	000101	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUBH[S]	Rd, Rx, Ry << shift	CCCC	0	000101	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUBB[S]	Rd, Rx, #imm	CCCC	0	000100	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUBB[S]	Rd, Rx, Ry << shift	CCCC	0	000100	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASLD[S]	Rd, Rx, #imm	CCCC	0	001011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASLD[S]	Rd, Rx, Ry << shift	CCCC	0	001011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASLW[S]	Rd, Rx, #imm	CCCC	0	001010	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASLW[S]	Rd, Rx, Ry << shift	CCCC	0	001010	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASLH[S]	Rd, Rx, #imm	CCCC	0	001001	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASLH[S]	Rd, Rx, Ry << shift	CCCC	0	001001	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASLB[S]	Rd, Rx, #imm	CCCC	0	001000	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASLB[S]	Rd, Rx, Ry << shift	CCCC	0	001000	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASRD[S]	Rd, Rx, #imm	CCCC	0	001111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASRD[S]	Rd, Rx, Ry << shift	CCCC	0	001111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASRW[S]	Rd, Rx, #imm	CCCC	0	001110	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASRW[S]	Rd, Rx, Ry << shift	CCCC	0	001110	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASRH[S]	Rd, Rx, #imm	CCCC	0	001101	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASRH[S]	Rd, Rx, Ry << shift	CCCC	0	001101	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ASRB[S]	Rd, Rx, #imm	CCCC	0	001100	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ASRB[S]	Rd, Rx, Ry << shift	CCCC	0	001100	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]LSRD[S]	Rd, Rx, #imm	CCCC	0	010011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]LSRD[S]	Rd, Rx, Ry << shift	CCCC	0	010011	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]LSRW[S]	Rd, Rx, #imm	CCCC	0	010010	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]LSRW[S]	Rd, Rx, Ry << shift	CCCC	0	010010	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]LSRH[S]	Rd, Rx, #imm	CCCC	0	010001	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]LSRH[S]	Rd, Rx, Ry << shift	CCCC	0	010001	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]LSRB[S]	Rd, Rx, #imm	CCCC	0	010000	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]LSRB[S]	Rd, Rx, Ry << shift	CCCC	0	010000	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ROLD[S]	Rd, Rx, #imm	CCCC	0	011011	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROLD[S]	Rd, Rx, Ry << shift	CCCC	0	011011	S	1	ssss	yyyyy	xxxxx	ddddd

Mnemonic	Arguments	Condition	Memory ?	operation	-S suffix	mode	(other)	(source2)	(source1)	destination
[cc.]ROLW[S]	Rd, Rx, #imm	CCCC	0	011010	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROLW[S]	Rd, Rx, Ry << shift	CCCC	0	011010	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ROLH[S]	Rd, Rx, #imm	CCCC	0	011001	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROLH[S]	Rd, Rx, Ry << shift	CCCC	0	011001	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]ROLB[S]	Rd, Rx, #imm	CCCC	0	011000	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]ROLB[S]	Rd, Rx, Ry << shift	CCCC	0	011000	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RORD[S]	Rd, Rx, #imm	CCCC	0	010111	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RORD[S]	Rd, Rx, Ry << shift	CCCC	0	010111	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RORW[S]	Rd, Rx, #imm	CCCC	0	010110	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RORW[S]	Rd, Rx, Ry << shift	CCCC	0	010110	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RORH[S]	Rd, Rx, #imm	CCCC	0	010101	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RORH[S]	Rd, Rx, Ry << shift	CCCC	0	010101	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]RORB[S]	Rd, Rx, #imm	CCCC	0	010100	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]RORB[S]	Rd, Rx, Ry << shift	CCCC	0	010100	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUMW[S]	Rd, Rx, #imm	CCCC	0	101110	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUMW[S]	Rd, Rx, Ry << shift	CCCC	0	101110	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUMH[S]	Rd, Rx, #imm	CCCC	0	101101	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUMH[S]	Rd, Rx, Ry << shift	CCCC	0	101101	S	1	ssss	yyyyy	xxxxx	ddddd
[cc.]SUMB[S]	Rd, Rx, #imm	CCCC	0	101100	S	0	iiii	iiiii	xxxxx	ddddd
[cc.]SUMB[S]	Rd, Rx, Ry << shift	CCCC	0	101100	S	1	ssss	yyyyy	xxxxx	ddddd
BAL	Rx, offset	0000	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BEQ	Rx, offset	0101	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BNE	Rx, offset	0100	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BMI	Rx, offset	1111	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BPL	Rx, offset	1110	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BVS	Rx, offset	1101	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BVC	Rx, offset	1100	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BCS	Rx, offset	0011	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BHQ	Rx, offset	0011	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BCC	Rx, offset	0010	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BLO	Rx, offset	0010	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BHI	Rx, offset	1011	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BLQ	Rx, offset	1010	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BGT	Rx, offset	0111	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BLE	Rx, offset	0110	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BGE	Rx, offset	1001	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
BLT	Rx, offset	1000	0	110000	0	X	rrrr	rrrrr	rrrrr	xxxxx
[cc.]BLK	Rx, offset	CCCC	0	110001	0	X	rrrr	rrrrr	rrrrr	xxxxx

Mnemonic	Arguments	Condition	Memory ?	operation	-S suffix	mode	(other)	(source2)	(source1)	destination
[cc.]TCR	Rd	CCCC	0	111010	0	X	XXXX	XXXXX	XXXXX	ddddd
[cc.]TRC	Rd	CCCC	0	111011	0	X	XXXX	XXXXX	XXXXX	ddddd
[cc.]TSR	Rd	CCCC	0	111100	0	X	XXXX	XXXXX	XXXXX	ddddd
[cc.]TRS	Rd	CCCC	0	111101	0	X	XXXX	XXXXX	XXXXX	ddddd
[cc.]RTI		CCCC	0	111110	0	X	XXXX	XXXXX	XXXXX	XXXXX
[cc.]SYS		CCCC	0	111111	0	X	XXXX	XXXXX	XXXXX	XXXXX
[cc.]ILG		CCCC	0	100000	0	X	XXXX	XXXXX	XXXXX	XXXXX
[cc.]ILG		CCCC	-	--ANY-	-	-	-NON	VALID	---OP	CODE-

Notes :

- "cc." mean any condition and are associated with the bits CCCC (bit 31 to bit 28). If there are not any condition, CCCC bits are 0000 and the instruction is always executed.
- "X" mean either 0 or 1
- "s" are the bits used for shifted operands
- "i" are the bits used for the immediate operand
- "x" are the bits used for the first operand register
- "y" are the bits used for the second operand register
- "r" bits are bits used to specify an offset
- if the suffix S is added, the S bit is 1. Otherwise, it's 0.

Table 3: Conditionnal prefix codes

Conditionnal Prefix	Associated code (binary)	Associated code (hexadecimal)
AL.	0000	0
EQ.	0101	5
NE.	0100	4
MI.	1111	F
PL.	1110	E
VS.	1101	D
VC.	1100	C
CS.	0011	3
HQ.	0011	3
CC.	0010	2
LO.	0010	2
HI.	1011	B
LQ.	1010	A
GT.	0111	7
LE.	0110	6
GE.	1001	9
LT.	1000	8