15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction
0	0	0	1	0	0	opcode			B/W	As		source				Single-operand arithmetic
0	0	0	1	0	0	0	0	0	B/W	As		source				**RRC** Rotate right through carry
0	0	0	1	0	0	0	0	1	0	As		source				**SWPB** Swap bytes
0	0	0	1	0	0	0	1	0	B/W	As		source				**RRA** Rotate right arithmetic
0	0	0	1	0	0	0	1	1	0	As		source				**SXT** Sign extend byte to word
0	0	0	1	0	0	1	0	0	B/W	As		source				**PUSH** Push value onto stack
0	0	0	1	0	0	1	0	1	0	As		source				**CALL** Subroutine call; push PC and move source to PC
0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	**RETI** Return from interrupt; pop SR then pop PC
0	0	1	condition			10-bit signed offset										Conditional jump; PC = PC + 2×offset
0	0	1	0	0	0	10-bit signed offset										**JNE**/**JNZ** Jump if not equal/zero
0	0	1	0	0	1	10-bit signed offset										**JEQ**/**JZ** Jump if equal/zero
0	0	1	0	1	0	10-bit signed offset										**JNC**/**JLO** Jump if no carry/lower
0	0	1	0	1	1	10-bit signed offset										**JC**/**JHS** Jump if carry/higher or same
0	0	1	1	0	0	10-bit signed offset										**JN** Jump if negative
0	0	1	1	0	1	10-bit signed offset										**JGE** Jump if greater or equal (N == V)
0	0	1	1	1	0	10-bit signed offset										**JL** Jump if less (N != V)
0	0	1	1	1	1	10-bit signed offset										**JMP** Jump (unconditionally)
opcode				source				Ad	B/W	As		destination				Two-operand arithmetic
0	1	0	0	source				Ad	B/W	As		destination				**MOV** Move source to destination
0	1	0	1	source				Ad	B/W	As		destination				**ADD** Add source to destination
0	1	1	0	source				Ad	B/W	As		destination				**ADDC** Add w/carry: dst += (src+C)
0	1	1	1	source				Ad	B/W	As		destination				**SUBC** Subtract w/ carry: dst -= (src+C)
1	0	0	0	source				Ad	B/W	As		destination				**SUB** Subtract; dst -= src
1	0	0	1	source				Ad	B/W	As		destination				**CMP** Compare; (dst-src); discard result
1	0	1	0	source				Ad	B/W	As		destination				**DADD** Decimal (BCD) addition: dst += src
1	0	1	1	source				Ad	B/W	As		destination				**BIT** Test bits; (dst & src); discard result
1	1	0	0	source				Ad	B/W	As		destination				**BIC** Bit clear; dest &= ~src
1	1	0	1	source				Ad	B/W	As		destination				**BIS** "Bit set" - logical OR; dst |= src
1	1	1	0	source				Ad	B/W	As		destination				**XOR** Bitwise XOR; dst ^= src
1	1	1	1	source				Ad	B/W	As		destination				**AND** Bitwise AND; dst &= src

As	src	Syntax	Description
0	*n*	**R*n***	Register direct. The operand is the contents of R*n*.
1	*n*	***x*(R*n*)**	Indexed. The operand is in memory at address R*n*+*x*.
2	*n*	**@R*n***	Register indirect. The operand is in memory at the address held in R*n*.
3	*n*	**@R*n*+**	Indirect autoincrement. As above, then the register is incremented by 1 or 2.
Addressing modes using R0 (PC)			
3	0	**#*x***	Immediate. @PC+ The operand is the next word in the instruction stream.
Addressing modes using R2 (SP) and R3 , special-case decoding			
1	2	**&*LABEL***	Absolute. The operand is in memory at address *x*.
2	2	**#4**	Constant. The operand is the constant 4.
3	2	**#8**	Constant. The operand is the constant 8.
0	3	**#0**	Constant. The operand is the constant 0.
1	3	**#1**	Constant. The operand is the constant 1. There is no index word.
2	3	**#2**	Constant. The operand is the constant 2.
3	3	**#-1**	Constant. The operand is the constant -1.