INSTRUCTION SET DETAILS

	Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa-decimal	Byte	Cycle	Explanation
Arithmetic operations	ADD A, Rn	0 0 1 0 1 n_2 n_1 n_0	28 ~ 2F	1	1	(A) ← (A) + (Rn)
	ADD A, direct	0 0 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	25 Byte 2	2	1	(A) ← (A) + (direct)
	ADD A, @Ri	0 0 1 0 0 1 1 1	26 ~ 27	1	1	(A) ← (A) + ((Ri))
	ADD A, #data	0 0 1 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	24 Byte 2	2	1	(A) ← (A) + #data
	ADDC A, Rn	0 0 1 1 1 n_2 n_1 n_0	38 ~ 3F	1	1	(A) ← (A) + (C) + (Rn)
	ADDC A, direct	0 0 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	35 Byte 2	2	1	(A) ← (A) + (C) + (direct)
	ADDC A, @Ri	0 0 1 1 0 1 1 1	36 ~ 37	1	1	(A) ← (A) + (C) + ((Ri))
	ADDC A, #data	0 0 1 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	34 Byte 2	2	1	(A) ← (A) + (C) + #data
	SUBB A, Rn	1 0 0 1 1 n_2 n_1 n_0	98 ~ 9F	1	1	(A) ← (A) − ((C) + (Rn))
	SUBB A, direct	1 0 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	95 Byte 2	2	1	(A) ← (A) − ((C) + (direct))
	SUBB A, @Ri	1 0 0 1 0 1 1 1	96 ~ 97	1	1	(A) ← (A) − ((C) + ((Ri)))
	SUBB A, #data	1 0 0 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	94 Byte 2	2	1	(A) ← (A) − ((C) + #data)
	INC A	0 0 0 0 0 1 0 0	04	1	1	(A) ← (A) + 1
	INC Rn	0 0 0 0 1 n_2 n_1 n_0	08 ~ 0F	1	1	(Rn) ← (Rn) + 1
	INC direct	0 0 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	05 Byte 2	2	1	(direct) ← (direct) + 1
	INC @Ri	0 0 0 0 0 1 1 1	06 ~ 07	1	1	((Ri)) ← ((Ri)) + 1
	INC DPTR	1 0 1 0 0 0 1 1	A3	1	2	(DPTR) ← (DPTR) + 1
	DEC A	0 0 0 1 0 1 0 0	14	1	1	(A) ← (A) − 1
	DEC Rn	0 0 0 1 1 n_2 n_1 n_0	18 ~ 1F	1	1	(Rn) ← (Rn) − 1
	DEC direct	0 0 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	15	2	1	(direct) ← (direct) − 1
	DEC @Ri	0 0 0 1 0 1 1 1	16 ~ 17	1	1	((Ri) ← ((Ri)) − 1
	MUL AB	1 0 1 0 0 1 0 0	A4	1	4	$(B_{15} \sim 8)$, $(A_7 \sim 0)$ ← (A) × (B)
	DIV AB	1 0 0 0 0 1 0 0	84	1	4	$(A_{15} \sim 8)$, $(B_7 \sim 0)$ ← (A)/(B)
	DA A	1 1 0 1 0 1 0 0	D4	1	1	Contents of Accumulator are BCD, IF [[$(A_3 \sim 0) > 9$] or [(AC) = 1]] THEN $(A_3 \sim 0)$ ← $(A_3 \sim 0) + 6$ AND IF [[$(A_7 \sim 4) > 9$] OR [(C) = 1]] THEN $(A_7 \sim 4)$ ← $(A_7 \sim 4) + 6$
Logical operations	ANL A, Rn	0 1 0 1 1 n_2 n_1 n_0	58 ~ 5F	1	1	(A) ← (A) AND (Rn)
	ANL A, direct	0 1 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	55 Byte 2	2	1	(A) ← (A) AND (direct)
	ANL A, @Ri	0 1 0 1 0 1 1 1	56 ~ 57	1	1	(A) ~ (A) AND ((Ri))
	ANL A, #data	0 1 0 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	54 Byte 2	2	1	(A) ← (A) AND #data

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa-decimal	Byte	Cycle	Explanation
Logical operations	ANL direct, A	0 1 0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	52 Byte 2	2	1	(direct) ← (direct) AND (A)
	ANL direct, #data	0 1 0 1 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	53 Byte 2 Byte 3	3	2	(direct) ← (direct) AND #data
	ORL A, Rn	0 1 0 0 1 n_2 n_1 n_0	48 ~ 4F	1	1	(A) ← (A) OR (Rn)
	ORL A, direct	0 1 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	45 Byte 2	2	1	(A) ← (A) OR (direct)
	ORL A, @Ri	0 1 0 0 0 1 1 1	46 ~ 47	1	1	(A) ← (A) OR ((Ri))
	ORL A, #data	0 1 0 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	44 Byte 2	2	1	(A) ← (A) OR #data
	ORL direct, A	0 1 0 0 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	42 Byte 2	2	1	(direct) ← (direct) OR (A)
	ORL direct, #data	0 1 0 0 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	43 Byte 2 Byte 3	3	2	(direct) ← (direct) OR #data
	XRL A, Rn	0 1 1 0 1 n_2 n_1 n_0	68 ~ 6F	1	1	(A) ← (A) XOR (Rn)
	XRL A, direct	0 1 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	65 Byte 2	2	1	(A) ← (A) XOR (direct)
	XRL A, @Ri	0 1 1 0 0 1 1 1	66 ~ 67	1	1	(A) ← (A) XOR ((Ri))
	XRL A, #data	0 1 1 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	64 Byte 2	2	1	(A) ← (A) XOR #data
	XRL direct, A	0 1 1 0 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	62 Byte 2	2	1	(direct) ← (direct) XOR (A)
	XRL direct, #data	0 1 1 0 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	63 Byte 2 Byte 3	3	2	(direct) ← (direct) XOR #data
	CLR A	1 1 1 0 0 1 0 0	E4	1	1	(A) ← 0
	CPL A	1 1 1 1 0 1 0 0	F4	1	1	(A) ← $(\bar{A})$
	RL A	0 0 1 0 0 0 1 1	23	1	1	A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 The contents of the accumulator are rotated left by one bit.
	RLC A	0 0 1 1 0 0 1 1	33	1	1	C A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 The contents of the accumulator and carry are rotated left by one bit.
	RR A	0 0 0 0 0 0 1 1	03	1	1	A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 The contents of the accumulator are rotated right by one bit.
	RRC A	0 0 0 1 0 0 1 1	13	1	1	C A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 The contents of the accumulator and carry are rotated right by one bit.
	SWAP A	1 1 0 0 0 1 0 0	C4	1	1	$(A_3 \sim 0) \rightleftarrows (A_7 \sim 4)$

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa-decimal	Byte	Cycle	Explanation
Data transfer	MOV A, Rn	1 1 1 0 1 n_2 n_1 n_0	E8 ~ EF	1	1	(A) ← (Rn)
	MOV A, direct	1 1 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	E5 Byte 2	2	1	(A) ← (direct)
	MOV A, @Ri	1 1 1 0 0 1 1 1	E6 ~ E7	1	1	(A) ← ((Ri))
	MOV A, #data	0 1 1 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	74 Byte 2	2	1	(A) ← #data
	MOV Rn, A	1 1 1 1 1 n_2 n_1 n_0	F8 ~ FF	1	1	(Rn) ← (A)
	MOV Rn, direct	1 0 1 0 1 n_2 n_1 n_0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	A8 ~ AF Byte 2	2	2	(Rn) ← (direct)
	MOV Rn, #data	0 1 1 1 1 n_2 n_1 n_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	78 ~ 7F Byte 2	2	1	(Rn) ← #data
	MOV direct, A	1 1 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	F5 Byte 2	2	1	(direct) ← (A)
	MOV direct, Rn	1 0 0 0 1 n_2 n_1 n_0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	88 ~ 8F Byte 2	2	2	(direct) ← (Rn)
	MOV direct 1, direct 2	1 0 0 0 0 1 0 1 a_7^2 a_6^2 a_5^2 a_4^2 a_3^2 a_2^2 a_1^2 a_0^2 a_7^1 a_6^1 a_5^1 a_4^1 a_3^1 a_2^1 a_1^1 a_0^1	85 Byte 2 Byte 3	3	2	(direct 1) ← (direct 2)
	MOV direct, @Ri	1 0 0 0 0 1 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	86 ~ 87 Byte 2	2	2	(direct) ← ((Ri))
	MOV direct, #data	0 1 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	75 Byte 2 Byte 3	3	2	(direct) ← #data
	MOV @Ri, A	1 1 1 1 0 1 1 1	F6 ~ F7	1	1	((Ri)) ← A
	MOV @Ri, direct	1 0 1 0 0 1 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	A6 ~ A7 Byte 2	2	2	((Ri)) ← (direct)
	MOV @Ri, #data	0 1 1 1 0 1 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	76 ~ 77 Byte 2	2	1	((Ri)) ← #data
	MOV DPTR, #data 16	1 0 0 1 0 0 0 0 d_{15} d_{14} d_{13} d_{12} d_{11} d_{10} d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	90 Byte 2 Byte 3	3	2	(DPTR) ← #data$_{16}$
	MOVC A, @A+ DPTR	1 0 0 1 0 0 1 1	93	1	2	(A) ← ((A) + (DPTR))
	MOVC A, @A+PC	1 0 0 0 0 0 1 1	83	1	2	(PC) ← (PC) + 1, (A) ← ((A) + (PC))
	MOVX A, @Ri	1 1 1 0 0 0 1 1	E2 ~ E3	1	2	(A) ← ((Ri)) External RAM
	MOVX A, @DPTR	1 1 1 0 0 0 0 0	E0	1	2	(A) ← ((DPTR)) External RAM
	MOVX @Ri, A	1 1 1 1 0 0 1 1	F2 ~ F3	1	2	((Ri)) ← (A) External RAM
	MOVX @DPTR, A	1 1 1 1 0 0 0 0	F0	1	2	((DPTR)) ← (A) External RAM
	PUSH direct	1 1 0 0 0 0 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	C0 Byte 2	2	2	(SP) ← (SP) + 1 ((SP)) ← (direct)
	POP direct	1 1 0 1 0 0 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	D0 Byte 2	2	2	(direct) ← ((SP)) (SP) ← (SP) − 1
	XCH A, Rn	1 1 0 0 1 n_2 n_1 n_0	C8 ~ CF	1	1	(A) ⇄ (Rn)
	XCH A, direct	1 1 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	C5 Byte 2	2	1	(A) ⇄ (direct)

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa-decimal	Byte	Cycle	Explanation
Data transfer	XCH A, @Ri	1 1 0 0 0 1 1 1	C6 ~ C7	1	1	(A) ⇄ ((Ri))
	XCHD A, @Ri	1 1 0 1 0 1 1 1	D6 ~ D7	1	1	($A_{3\sim0}$) ⇄ (($Ri_{3\sim0}$))
Boolean variable manipulation	CLR C	1 1 0 0 0 0 1 1	C3	1	1	(C) ← 0
	CLR bit	1 1 0 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	C2 Byte 2	2	1	(bit) ← 0
	SETB C	1 1 0 1 0 0 1 1	D3	1	1	(C) ← 1
	SETB bit	1 1 0 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	D2 Byte 2	2	1	(bit) ← 1
	CPL C	1 0 1 1 0 0 1 1	B3	1	1	(C) ← ($\overline{C}$)
	CPL bit	1 0 1 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	B2 Byte 2	2	1	(bit) ← ($\overline{bit}$)
	ANL C, bit	1 0 0 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	82 Byte 2	2	2	(C) ← (C) AND (bit)
	ANL C,/bit	1 0 1 1 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	B0 Byte 2	2	2	(C) ← (C) AND ($\overline{bit}$)
	ORL C, bit	0 1 1 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	72 Byte 2	2	2	(C) ← (C) OR (bit)
	ORL C,/bit	1 0 1 0 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	A0 Byte 2	2	2	(C) ← (C) OR ($\overline{bit}$)
	MOV C, bit	1 0 1 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	A2 Byte 2	2	1	(C) ← (bit)
	MOV bit, C	1 0 0 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	92 Byte 2	2	2	(bit) ← (C)
Program branching	ACALL addr 11	a_{10} a_9 a_8 1 0 0 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	Byte 1 Byte 2	2	2	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← ($PC_{7\sim0}$) (SP) ← (SP) + 1 ((SP)) ← ($PC_{15\sim8}$) (PC) ← page address
	LCALL addr 16	0 0 0 1 0 0 1 0 a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	12 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← ($PC_{7\sim0}$) (SP) ← (SP) + 1 ((SP)) ← ($PC_{15\sim8}$) (PC) ← $addr_{15\sim0}$
	RET	0 0 1 0 0 0 1 0	22	1	2	($PC_{15\sim8}$) ← ((SP)) (SP) ← (SP) − 1 ($PC_{7\sim0}$) ← ((SP)) (SP) ← (SP) − 1
	RETI	0 0 1 1 0 0 1 0	32	1	2	($PC_{15\sim8}$) ← ((SP)) (SP) ← (SP) − 1 ($PC_{7\sim0}$) ← ((SP)) (SP) ← (SP) − 1
	AJMP addr 11	a_{10} a_9 a_8 0 0 0 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	Byte 1 Byte 2	2	2	(PC) ← (PC) + 2 ($PC_{10\sim0}$) ← page address
	LJMP addr 16	0 0 0 0 0 0 1 0 a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	02 Byte 2 Byte 3	3	2	(PC) ← $addr_{15\sim0}$

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Hexa-decimal	Byte	Cycle	Explanation
Program branching	SJMP rel	1	0	0	0	0	0	0	0	80	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2			(PC) ← (PC) + rel
	JMP @A+DPTR	0	1	1	1	0	0	1	1	73	1	2	(PC) ← (A) + (DPTR)
	JZ rel	0	1	1	0	0	0	0	0	60	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte			IF (A) = 0 THEN (PC) ← (PC) + rel
	JNZ rel	0	1	1	1	0	0	0	0	70	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2			IF (A) ≠ 0 THEN (PC) ← (PC) + rel
	JC rel	0	1	0	0	0	0	0	0	40	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2			IF (C) ≠ 1 THEN (PC) ← (PC) + rel
	JNC rel	0	1	0	1	0	0	0	0	50	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2			IF (C) ≠ 0 THEN (PC) ← (PC) + rel
	JB bit, rel	0	0	1	0	0	0	0	0	20	3	2	(PC) ← (PC) + 3
		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2			IF (bit) = 1 THEN (PC) ← (PC) + rel
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			
	JNB bit, rel	0	0	1	1	0	0	0	0	30	3	2	(PC) ← (PC) + 3
		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2			IF (bit) = 0 THEN (PC) ← (PC) + rel
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			
	JBC bit, rel	0	0	0	1	0	0	0	0	10	3	2	(PC) ← (PC) + 3
		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2			IF (bit) = 1 THEN (bit) ← 0
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			(PC) ← (PC) + rel
	CJNE A, direct, rel	1	0	1	1	0	1	0	1	B5	3	2	(PC) ← (PC) + 3
		a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2			IF (direct) < (A) THEN (PC) ← (PC)
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			+ rel and (C) ← 0
													IF (direct) > (A) THEN (PC) ← (PC)
													+ rel and (C) ← 1
	CJNE A, #data, rel	1	0	1	1	0	1	0	0	B4	3	2	(PC) ← (PC) + 3
		d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2			IF #data < (A)
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			THEN (PC) ← (PC) + rel and
													(C) ← 0
													IF #data > (A) THEN (PC) ← (PC) +
													rel and (C) ← 1
	CJNE Rn, #data, rel	1	0	1	1	1	n_2	n_1	n_0	B8 ~ BF	3	2	(PC) ← (PC) + 3
		d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2			IF #data < (Rn) THEN (PC) ← (PC)
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			+ rel and (C) ← 0
													IF #data > (Rn) THEN (PC) ← (PC)
													+ rel and (C) ← 1
	CJNE @Ri, #data, rel	1	0	1	1	0	1	1	1	B6 ~ B7	3	2	(PC) ← (PC) + 3
		d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2			IF #data < ((Ri)) THEN (PC) ←
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			(PC) + rel and (C) ← 0
													IF #data > ((Ri)) THEN (PC) ←
													(PC) + rel and (C) ← 1
	DJNZ Rn, rel	1	1	0	1	1	n_2	n_1	n_0	D8 ~ DF	2	2	(PC) ← (PC) + 2
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2			(Rn) ← (Rn) − 1
													IF (Rn) ≠ 0 THEN (PC) ← (PC) + rel
	DJNZ direct, rel	1	1	0	1	0	1	0	1	D5	3	2	(PC) ← (PC) + 3
		a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2			(direct) ← (direct) − 1
		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3			IF (direct) ≠ 0 THEN (PC) ← (PC)
													+ rel
	NOP	0	0	0	0	0	0	0	0	00	1	1	(PC) ← (PC) + 1

NOTES ON THE INSTRUCTION SET AND THE ADDRESSING MODES

Rn – Register R7-R0 of the currently selected Register Bank.

direct – 8-bit internal data location's address. This could be an Internal Data RAM location (0 – 127) or a SFR [i.e., I/O port, control register, status register, etc. (128 – 255)].

@Ri – 8-bit internal data RAM location (0 – 255) addressed indirectly through register R1 or R0.

#data – 8-bit constant included in instruction.

#data 16 – 16-bit constant included in instruction.

addr 16 – 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 – 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel – Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit – Direct Addressed bit in Internal Data RAM or Special Function Register.

INSTRUCTIONS THAT AFFECT FLAG SETTINGS[1]

INSTRUCTION	FLAG C	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	O	X	
DIV	O	X	
DA	X		
RRC	X		
RLC	X		
SETB C	I		

INSTRUCTION	FLAG C	OV	AC
CLR C	O		
CPL C	X		
ANL C, bit	X		
ANL C,/bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		

[1] Note that operations on SFR byte address 208 or bit addresses 208-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.