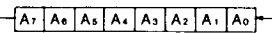
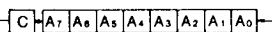
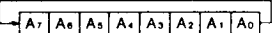
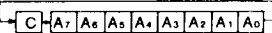
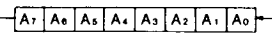
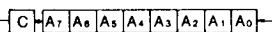
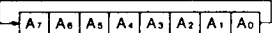
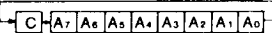


INSTRUCTION SET DETAILS

	Mnemonic	Instruction Code	Hexa- decimal	Byte	Cycle	Explanation
		D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀				
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
Logical operations	MUL AB	1 0 1 0 0 1 0 0	A4	1	4	(B ₁₅ ~ a ₀), (A ₇ ~ a ₀) ← (A) × (B)
	DIV AB	1 0 0 0 0 1 0 0	84	1	4	(A ₁₅ ~ a ₀), (B ₇ ~ a ₀) ← (A)/(B)
	DAA	1 1 0 1 0 1 0 0	D4	1	1	Contents of Accumulator are BCD, IF [(A ₃ ~ a ₀) > 9] or [(AC) = 1] THEN (A ₃ ~ a ₀) ← (A ₃ ~ a ₀) + 6 AND IF [(A ₇ ~ a ₄) > 9] OR [(C) = 1] THEN (A ₇ ~ a ₄) ← (A ₇ ~ a ₄) + 6
	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
	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))
Logical operations	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) ← (A)
Logical operations	RLA	0 0 1 0 0 0 1 1	23	1	1	 The contents of the accumulator are rotated left by one bit.
	RLC A	0 0 1 1 0 0 1 1	33	1	1	 The contents of the accumulator and carry are rotated left by one bit.
	RRA	0 0 0 0 0 0 1 1	03	1	1	 The contents of the accumulator are rotated right by one bit.
	RRC A	0 0 0 1 0 0 1 1	13	1	1	 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 ~ a ₀) ← (A ₇ ~ a ₄) (A ₇ ~ a ₄) ← (A ₃ ~ a ₀)

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code	Hexa- decimal	Byte	Cycle	Explanation
		D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀				
Arithmetic 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) ← (A)
	RLA	0 0 1 0 0 0 1 1	23	1	1	 The contents of the accumulator are rotated left by one bit.
	RLC A	0 0 1 1 0 0 1 1	33	1	1	 The contents of the accumulator and carry are rotated left by one bit.
	RRA	0 0 0 0 0 0 1 1	03	1	1	 The contents of the accumulator are rotated right by one bit.
	RRC A	0 0 0 1 0 0 1 1	13	1	1	 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 ~ a ₀) ← (A ₇ ~ a ₄) (A ₇ ~ a ₄) ← (A ₃ ~ a ₀)

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code	Hexa- decimal	Byte	Cycle	Explanation
		D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀				
Data transfer	MOVA, 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)
	MOVA, @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 a ₆ a ₅ a ₄ a ₃ a ₂ a ₁ a ₀ a ₇ a ₆ a ₅ a ₄ a ₃ a ₂ a ₁ a ₀	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
	MOVDPTR, #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
	MOVCA, @A+DPTR	1 0 0 1 0 0 1 1	93	1	2	(A) ← ((A) + (DPTR))
	MOVCA, @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
	XCHA, Rn	1 1 0 0 1 n ₂ n ₁ n ₀	C8 ~ CF	1	1	(A) ↔ (Rn)
	XCHA, 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	Hexa- decimal	Byte	Cycle	Explanation
		D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀				
Data transfer	XCHA, @Ri	1 1 0 0 0 1 1 1	C6 ~ C7	1	1	(A) ↔ ((Ri))
	XCHDA, @Ri	1 1 0 1 0 1 1 1	D6 ~ D7	1	1	(A ₃ ~ a ₃) ↔ ((Ri ₃ ~ r ₃))
	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) ← (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) ← (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	B2 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 (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 (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 ~ a ₇) (SP) ← (SP) + 1 ((SP)) ← (PC ₁₅ ~ a ₁₅) (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 ~ a ₇) (SP) ← (SP) + 1 ((SP)) ← (PC ₁₅ ~ a ₁₅) (PC) ← addr ₁₅ ~ 0
	RET	0 0 1 0 0 0 1 0	22	1	2	(PC ₁₅ ~ a ₁₅) ← ((SP)) (SP) ← (SP) - 1 (PC ₇ ~ a ₇) ← ((SP)) (SP) ← (SP) - 1
	RETI	0 0 1 1 0 0 1 0	32	1	2	(PC ₁₅ ~ a ₁₅) ← ((SP)) (SP) ← (SP) - 1 (PC ₇ ~ a ₇) ← ((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 ~ a ₁₀) ← 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 ~ 0

INSTRUCTION SET DETAILS (CONT.)

	Mnemonic	Instruction Code	Hexa- decimal	Byte	Cycle	Explanation
		D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀				
Program branching	SJMP rel	1 0 0 0 0 0 0 0 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	80 Byte 2	2	2	(PC) ← (PC) + 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 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	60 Byte	2	2	(PC) ← (PC) + 2 IF (A) = 0 THEN (PC) ← (PC) + rel
	JNZ rel	0 1 1 1 0 0 0 0 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	70 Byte 2	2	2	(PC) ← (PC) + 2 IF (A) ≠ 0 THEN (PC) ← (PC) + rel
	JC rel	0 1 0 0 0 0 0 0 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	40 Byte 2	2	2	(PC) ← (PC) + 2 IF (C) ≠ 1 THEN (PC) ← (PC) + rel
	JNC rel	0 1 0 1 0 0 0 0 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	50 Byte 2	2	2	(PC) ← (PC) + 2 IF (C) ≠ 0 THEN (PC) ← (PC) + rel
	JB bit, rel	0 0 1 0 0 0 0 0 b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	20 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF (bit) = 1 THEN (PC) ← (PC) + rel
	JNB bit, rel	0 0 1 1 0 0 0 0 b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	30 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF (bit) = 0 THEN (PC) ← (PC) + rel
	JBC bit, rel	0 0 0 1 0 0 0 0 b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	10 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF (bit) = 1 THEN (bit) ← 0 (PC) ← (PC) + rel
	CJNE A, direct, rel	1 0 1 1 0 1 0 1 a ₇ a ₆ a ₅ a ₄ a ₃ a ₂ a ₁ a ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	B5 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF (direct) < (A) THEN (PC) ← (PC) + 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 d ₇ d ₆ d ₅ d ₄ d ₃ d ₂ d ₁ d ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	B4 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF #data < (A) 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 d ₇ d ₆ d ₅ d ₄ d ₃ d ₂ d ₁ d ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	B8 ~ BF Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF #data < (Rn) THEN (PC) ← (PC) + 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 d ₇ d ₆ d ₅ d ₄ d ₃ d ₂ d ₁ d ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	B6 ~ B7 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 IF #data < ((Ri)) THEN (PC) ← (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 r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	D8 ~ DF Byte 2	2	2	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF (Rn) ≠ 0 THEN (PC) ← (PC) + rel
	DJNZ direct, rel	1 1 0 1 0 1 0 1 a ₇ a ₆ a ₅ a ₄ a ₃ a ₂ a ₁ a ₀ r ₇ r ₆ r ₅ r ₄ r ₃ r ₂ r ₁ r ₀	D5 Byte 2 Byte 3	3	2	(PC) ← (PC) + 3 (direct) ← (direct) - 1 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 SETTINGS¹

INSTRUCTION	FLAG	INSTRUCTION	FLAG
	C OV AC		C OV AC
ADD	X X X	CLR C	X
ADDC	X X X	CPL C	X
SUBB	X X X	ANL C, bit	X
MUL	O X	ANL C, /bit	X
DIV	O X	ORL C, bit	X
DA	X	ORL C, /bit	X
RRC	X	MOV C, bit	X
RLC	X	CJNE	X
SETB C	I		

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.