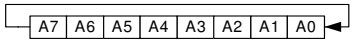
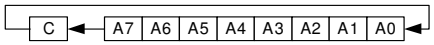
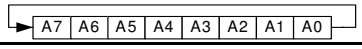
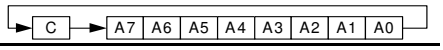
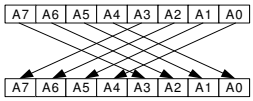
	Mnemonic	Instruction code								Cycles/instruction (83ns/cycle)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Arithmetic instructions	ADD A,Rn	0	0	1	0	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A)+(Rn)
	ADD A,direct	0 a7	0 a6	1 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A)+(direct)
	ADD A,@Ri	0	0	1	0	0	1	1	i	1	2	4	2 or 4	(A)=(A)+((Ri))
	ADD A,#data	0 d7	0 d6	1 d5	0 d4	0 d3	1 d2	0 d1	0 d0	1	2	6		(A)=(A)+#data
	ADDC A,Rn	0	0	1	1	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A)+(Rn)+(C)
	ADDC A,direct	0 a7	0 a6	1 a5	1 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A)+(direct)+(C)
	ADDC A,@Ri	0	0	1	1	0	1	1	i	1	2	4	2 or 4	(A)=(A)+((Ri))+(C)
	ADDC A,#data	0 d7	0 d6	1 d5	1 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)=(A)+#data+(C)
	SUBB A,Rn	1	0	0	1	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A)-(Rn)-(C)
	SUBB A,direct	1 a7	0 a6	0 a5	1 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A)-(direct)-(C)
	SUBB A,@Ri	1	0	0	1	0	1	1	i	1	2	4	2 or 4	(A)=(A)-((Ri))-(C)
	SUBB A,#data	1 d7	0 d6	0 d5	1 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)=(A)-#data-(C)
	INC A	0	0	0	0	0	1	0	0	1	2	4	2 or 4	(A)=(A)+1
	INC Rn	0	0	0	0	1	n2	n1	n0	1	2	4	2 or 4	(Rn)=(Rn)+1
	INC direct	0 a7	0 a6	0 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(direct)=(direct)+1
	INC @Ri	0	0	0	0	0	1	1	i	1	2	4	2 or 4	((Ri))=((Ri))+1
	INC DPTR	1	0	1	0	0	0	1	1	2	4	4	4	(DPTR)=(DPTR)+1
	DEC A	0	0	0	1	0	1	0	0	1	2	4	2 or 4	(A)=(A)-1
	DEC Rn	0	0	0	1	1	n2	n1	n0	1	2	4	2 or 4	(Rn)=(Rn)-1
	DEC direct	0 a7	0 a6	0 a5	1 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(direct)=(direct)-1
	DEC @Ri	0	0	0	1	0	1	1	i	1	2	4	2 or 4	((Ri))=((Ri))-1
	MUL AB	1	0	1	0	0	1	0	0	4	8	8	8	(B),(A)=(A)x(B) B=MSB A=LSB
	DIV AB	1	0	0	0	0	1	0	0	4	8	8	8	(A),(B)=(A)/(B) A=Q B=R
	DA A	1	1	0	1	0	1	0	0	1	2	4	2 or 4	Contents A=BCD , use ADD or ADDC with BCD If flags not modified DA A will adjust result to BCD
Logical instructions	ANL A,Rn	0	1	0	1	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A) AND (Rn)
	ANL A,direct	0 a7	1 a6	0 a5	1 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A) AND (direct)
	ANL A,@Ri	0	1	0	1	0	1	1	i	1	2	4	2 or 4	(A)=(A) AND ((Ri))
	ANL A,#data	0 d7	1 d6	0 d5	1 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)=(A) AND #data
	ANL direct,A	0 a7	1 a6	0 a5	1 a4	0 a3	0 a2	1 a1	0 a0	1	2	6	4	(direct)=(direct) AND (A)
	ANL direct,#data	0 d7	1 d6	0 d5	1 d4	0 d3	0 d2	1 d1	1 d0	2	4	10	6 or 8	(direct)=(direct) AND #data
	ORL A,Rn	0	1	0	0	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A) OR (Rn)
	ORL A,direct	0 a7	1 a6	0 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A) OR (direct)
	ORL A,@Ri	0	1	0	0	0	1	1	i	1	2	4	2 or 4	(A)=(A) OR ((Ri))

	Mnemonic	Instruction code								Cycles/instruction				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Logical instructions	ORL A,#data	0 d7	1 d6	0 d5	0 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)=(A) OR #data
	ORL direct,A	0 a7	1 a6	0 a5	0 a4	0 a3	0 a2	1 a1	0 a0	1	2	6	4	(direct)=(direct) OR (A)
	ORL direct,#data	0 d7	1 d6	0 d5	0 d4	0 d3	0 d2	1 d1	1 d0	2	4	10	6 or 8	(direct)=(direct) OR #data
	XRL A,Rn	0	1	1	0	1	n2	n1	n0	1	2	4	2 or 4	(A)=(A) XOR (Rn)
	XRL A,direct	0 a7	1 a6	1 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)=(A) XOR (direct)
	XRL A,@Ri	0	1	1	0	0	1	1	i	1	2	4	2 or 4	(A)=(A) XOR ((Ri))
	XRL A,#data	0 d7	1 d6	1 d5	0 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)=(A) XOR #data
	XRL direct,A	0 a7	1 a6	1 a5	0 a4	0 a3	0 a2	1 a1	0 a0	1	2	6	4	(direct)=(direct) XOR (A)
	XRL direct,#data	0 d7	1 d6	1 d5	0 d4	0 d3	0 d2	1 d1	1 d0	2	4	10	6 or 8	(direct)=(direct) XOR #data
	CLR A	1	1	1	0	0	1	0	0	1	2	4	2 or 4	(A)=0
	CPL A	1	1	1	1	0	1	0	0	1	2	4	2 or 4	ONES COMPLEMENT OF (A)
	RL A	0	0	1	0	0	0	1	1	1	2	4	2 or 4	rotate 1 bit left 
	RLC A	0	0	1	1	0	0	1	1	1	2	4	2 or 4	rotate 1 bit left 
	RR A	0	0	0	0	0	0	1	1	1	2	4	2 or 4	rotate 1 bit right 
	RRC A	0	0	0	1	0	0	1	1	1	2	4	2 or 4	rotate 1 bit right 
	SWAP A	1	1	0	0	0	1	0	0	1	2	4	2 or 4	
Data transfer instructions	MOV A,Rn	1	1	1	0	1	n2	n1	n0	1	2	4	2 or 4	(A)<--(Rn)
	MOV A,direct	1 a7	1 a6	1 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)<--(direct)
	MOV A,@Ri	1	1	1	0	0	1	1	i	1	2	4	2 or 4	(A)<--((Ri))
	MOV A,#data	0 d7	1 d6	1 d5	1 d4	0 d3	1 d2	0 d1	0 d0	1	2	6	4	(A)<--#data
	MOV Rn,A	1	1	1	1	1	n2	n1	n0	1	2	4	2 or 4	(Rn)<--(A)
	MOV Rn,direct	1 a7	0 a6	1 a5	0 a4	1 a3	n2 a2	n1 a1	n0 a0	2	4	8	6	(Rn)<--(direct)
	MOV Rn,#data	0 d7	1 d6	1 d5	1 d4	1 d3	n2 d2	n1 d1	n0 d0	1	2	6	4	(Rn)<--#data
	MOV direct,A	1 a7	1 a6	1 a5	1 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(direct)<--(A)
	MOV direct,Rn	1 a7	0 a6	0 a5	0 a4	1 a3	n2 a2	n1 a1	n0 a0	2	4	8	6	(direct)<--(Rn)
	MOV direct1,direct2	1 2a7 1a7	0 a6 a6	0 a5 a5	0 a4 a4	0 a3 a3	1 a2 a2	0 a1 a1	1 a0 a0	2	4	10	6 or 8	(direct1)<--(direct2)

	Mnemonic	Instruction code								Cycles/instruction				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Data transfer instructions	MOV direct,@Ri	1 a7	0 a6	0 a5	0 a4	0 a3	1 a2	1 a1	i a0	2	4	8	6	(direct)<--((Ri))
	MOV direct,#data	0 a7 d7	1 a6 d6	1 a5 d5	1 a4 d4	0 a3 d3	1 a2 d2	0 a1 d1	1 a0 d0	2	4	10	6 or 8	(direct)<--#data
	MOV @Ri,A	1	1	1	1	0	1	1	i	1	2	4	2 or 4	((Ri))<--(A)
	MOV @Ri,direct	1 a7	0 a6	1 a5	0 a4	0 a3	1 a2	1 a1	i a0	2	4	8	6	((Ri))<--(direct)
	MOV @Ri,#data	0 d7	1 d6	1 d5	1 d4	0 d3	1 d2	1 d1	i d0	1	2	6	4	((Ri))<--#data
	MOV DPTR,#data	1 d15 d7	0 d14 d6	0 d13 d5	1 d12 d4	0 d11 d3	0 d10 d2	0 d9 d1	0 d8 d0	2	4	10	6 or 8	(DPTR)<--#data16 equals: (DPH)<--high #data16 (DPL)<--low #data16
	MOVC A,@A+DPTR	1	0	0	1	0	0	1	1	2	4	6	4|6|8	(A)<--((A)+(DPTR))
	MOVC @(DPTR++),A	1	0	1	0	0	1	0	1	**ERROR**	4	4	4 or 6	((DPTR))<--(A) and DPTR+1
	MOVC A,@A+PC	1	0	0	0	0	0	1	1	2	4	6	4|6|8	(A)<--((A)+(PC+1))
	MOVX A,@Ri	1	1	1	0	0	0	1	i	2	4	6	4 or 6	(A)<--((Ri))
	MOVX A,@DPTR	1	1	1	0	0	0	0	0	2	4	6	4 or 6	(A)<--((DPTR))
	MOVX @Ri,A	1	1	1	1	0	0	1	i	2	4	6	4 or 6	((Ri))<--(A)
	MOVX @DPTR,A	1	1	1	1	0	0	0	0	2	4	6	4 or 6	((DPTR))<--(A)
	PUSH direct	1 a7	1 a6	0 a5	0 a4	0 a3	0 a2	0 a1	0 a0	2	4	8	6	(SP)=(SP)+1 ((SP))<--(direct)
	POP direct	1 a7	1 a6	0 a5	1 a4	0 a3	0 a2	0 a1	0 a0	2	4	8	6	(direct)<--((SP)) (SP)=(SP)-1
	XCH A,Rn	1	1	0	0	1	n2	n1	n0	1	2	4	2 or 4	(A)<>(Rn)
	XCH A,direct	1 a7	1 a6	0 a5	0 a4	0 a3	1 a2	0 a1	1 a0	1	2	6	4	(A)<>(direct)
	XCH A,@Ri	1	1	0	0	0	1	1	i	1	2	4	2 or 4	(A)<>((Ri))
	XCHD A,@Ri	1	1	0	1	0	1	1	i	1	2	4	2 or 4	(A)<>((Ri)) (only low nibble)
Boolean manipulation instructions	CLR C	1	1	0	0	0	0	1	1	1	2	4	2 or 4	(C)=0
	CLR bit	1 b7	1 b6	0 b5	0 b4	0 b3	0 b2	1 b1	0 b0	1	2	6	4	(bit)=0
	SETB C	1	1	0	1	0	0	1	1	1	2	4	2 or 4	(C)=1
	SETB bit	1 b7	1 b6	0 b5	1 b4	0 b3	0 b2	1 b1	0 b0	1	2	6	4	(bit)=1
	CPL C	1	0	1	1	0	0	1	1	1	2	4	2 or 4	complement carry flag
	CPL bit	1 b7	0 b6	1 b5	1 b4	0 b3	0 b2	1 b1	0 b0	1	2	6	4	complement (bit)
	ANL C,bit	1 b7	0 b6	0 b5	0 b4	0 b3	0 b2	1 b1	0 b0	2	4	8	6	(C)=(C) AND (bit)
	ANL C,/bit	1 b7	0 b6	1 b5	1 b4	0 b3	0 b2	0 b1	0 b0	2	4	8	6	(C)=(C) AND NOT(BIT)
	ORL C,bit	0 b7	1 b6	1 b5	1 b4	0 b3	0 b2	1 b1	0 b0	2	4	8	6	(C)=(C) OR (bit)
	ORL C,/BIT	1 b7	0 b6	1 b5	0 b4	0 b3	0 b2	0 b1	0 b0	2	4	8	6	(C)=(C) OR NOT (bit)
	MOV C,bit	1 b7	0 b6	1 b5	0 b4	0 b3	0 b2	1 b1	0 b0	1	2	6	4	(C)=(bit)

	Mnemonic	Instruction code								Cycles/instruction				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
B. m. i.	MOV bit,C	1	0	0	1	0	0	1	0	2	4	8	6	(bit)=(C)
		b7	b6	b5	b4	b3	b2	b1	b0					
Program branching	ACALL addr11	a10	a9	a8	1	0	0	0	1	2	4	8	6 or 8	call subroutine (2KByte page)
		a7	a6	a5	a4	a3	a2	a1	a0					save return address on stack
	LCALL addr16	0	0	0	1	0	0	1	0	2	4	10	8	call subroutine (64KByte range)
		a15	a14	a13	a12	a11	a10	a9	a8					save return address on stack
		a7	a6	a5	a4	a3	a2	a1	a0					
	RET	0	0	1	0	0	0	1	0	2	4	4	4 or 6	pop return address and jump
	RETI	0	0	1	1	0	0	1	0	2	4	4	4 or 6	pop return address and jump
														restore interrupt scanning
	AJMP addr11	a10	a9	a8	0	0	0	0	1	2	4	8	6 or 8	jump to address (2KByte page)
		a7	a6	a5	a4	a3	a2	a1	a0					
	LJMP addr16	0	0	0	0	0	0	1	0	2	4	10	8	jump to address (64KByte range)
		a15	a14	a13	a12	a11	a10	a9	a8					
		a7	a6	a5	a4	a3	a2	a1	a0					
	SJMP rel	1	0	0	0	0	0	0	0	2	4	8	6 or 8	jump to address (127(8) byte range)
		r7	r6	r5	r4	r3	r2	r1	r0					
	JMP @A+DPTR	0	1	1	1	0	0	1	1	2	4	4	4 or 6	(PC)<--(A)+(DPTR)
	JZ rel	0	1	1	0	0	0	0	0	2	4	8	6 or 8	If (A)=0 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	JNZ rel	0	1	1	1	0	0	0	0	2	4	8	6 or 8	If (A) ≠0 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	JC rel	0	1	0	0	0	0	0	0	2	4	8	6 or 8	If (C)=1 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	JNC rel	0	1	0	1	0	0	0	0	2	4	8	6 or 8	If (C)=0 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	JB bit,rel	0	0	1	0	0	0	0	0	2	4	10	6 or 8	If (bit)=1 jump rel
		b7	b6	b5	b4	b3	b2	b1	b0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	JNB bit,rel	0	0	1	1	0	0	0	0	2	4	10	6 or 8	If (bit)=0 jump rel
		b7	b6	b5	b4	b3	b2	b1	b0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	JBC bit,rel	0	0	0	1	0	0	0	0	2	4	10	6 or 8	If (bit)=1 jump rel and clear (bit)
		b7	b6	b5	b4	b3	b2	b1	b0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	CJNE A,direct,rel	1	0	1	1	0	1	0	1	2	4	10	6 or 8	If (A)≠(direct) jump rel
		a7	a6	a5	a4	a3	a2	a1	a0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	CJNE A,#data,rel	1	0	1	1	0	1	0	0	2	4	10	6 or 8	If (A)≠ #data jump rel
		d7	d6	d5	d4	d3	d2	d1	d0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	CJNE Rn,#data,rel	1	0	1	1	1	n2	n1	n0	2	4	10	6 or 8	If (Rn)≠ #data jump rel
		d7	d6	d5	d4	d3	d2	d1	d0					
		r7	r6	r5	r4	r3	r2	r1	r0					
	CJNE @Ri,#data,rel	1	0	1	1	0	1	1	i	2	4	10	6 or 8	If ((Ri))≠ #data jump rel
		d7	d6	d5	d4	d3	d2	d1	d0					
		r7	r6	r5	r4	r3	r2	r1	r0					

	Mnemonic	Instruction code								Cycles/instruction				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Program branching	DJNZ Rn,rel	1	1	0	1	1	n2	n1	n0	2	4	8	6 or 8	(Rn)=(Rn)-1
		r7	r6	r5	r4	r3	r2	r1	r0					If (Rn)≠ 0 jump rel
	DJNZ direct,rel	1	1	0	1	0	1	0	1	2	4	10	6 or 8	(direct)=(direct)-1
		a7	a6	a5	a4	a3	a2	a1	a0					If (direct)≠ 0 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	NOP	0	0	0	0	0	0	0	0	1	2	4	2 or 4	Just waist time

Notes on instruction set and the adressing modes

Rn	Register R7-R0 of currently selected register bank (RS0 and RS1 in PSW SFR)
direct	8 bit GPR address (00h-7fh and 80h-ffh SFR register space)
@Ri	Ri is a 8 bit pointer to indirect addressable GPR's (00h-ffh) (i=0 or i=1)
#data	8 bit constant included in instruction (range 00h-ffh)
#data16	16 bit constant included in instruction (range 0000h-ffffh)
addr16	16 bit destination address (range 64KByte)
addr11	11 bit destination address (range within current 2Kbyte page)
rel	8 bit two's complement jump offset (relative to first byte next instruction)
bit	8 bit address of a direct adressable bit

Instructions that affec flag settings

Instruction	Flag		
ADD	x	x	x
ADDC	x	x	x
SUBB	x	x	x
MUL	O	x	
DIV	O	x	
DA A	x		
RRC	x		
RLC	x		
SETB C	1		
CLR C	0		
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		