MNEMO	ARGUMENTS	cyc	byte	flags	ACTION	comment
ADD	A,Rn	1	1	C,OV,AC	A=A+Rn	1
	A,direct	1	2	C,OV,AC	A=A+direct	2
	A,@Ri	1	1	C,OV,AC	A=A+[Ri]	3
	A,#data	1	2	C,OV,AC	A=A+#data	4
ADDC	A,Rn	1	1	C,OV,AC	A=A+Rn+C	1
	A,direct	1	2	C,OV,AC	A=A+direct+C	2
	A,@Ri	1	1	C,OV,AC	A=A+[Ri]+C	3
	A,#data	1	2	C,OV,AC	A=A+#data+C	4
SUBB	A,Rn	1	1	C,OV,AC	A=A-Rn-C	1
	A,direct	1	2	C,OV,AC	A=A-direct-C	2
	A,@Ri	1	1	C,OV,AC	A=A-[Ri]-C	3
	A,#data	1	2	C,OV,AC	A=A-#data-C	4
INC	A	1	1	-	A=A+1	
	Rn	1	1	-	Rn=Rn+1	1
	direct	1	2	-	direct=direct+1	2
	@Ri	1	1	-	[Ri]=[Ri]+1	3
DEC	A	1	1	-	A=A-1	
	Rn	1	1	-	Rn=Rn-1	1
	direct	1	2	-	direct=direct-1	2
	@Ri	1	1	-	[Ri]=[Ri]-1	3
MUL	AB	4	1	C=0, OV	AB=A*B	5
DIV	AB	4	1	C=0, OV	A=AB/A; B=AB%A	5
DA	A	1	1	C	decimal adjust A	6
ANL	A,Rn	1	1	-	A=A&Rn	1
	A,direct	1	2	-	A=A&direct	2
	A,@Ri	1	1	-	A=A&[Ri]	3
	A,#data	1	2	-	A=A&#data	4
	direct,A	1	2	-	direct=direct&A	2
	direct,#data	2	3	-	direct=direct&#data	2,4
ORL	A,Rn	1	1	-	A=A|Rn	1
	A,direct	1	2	-	A=A|direct	2
	A,@Ri	1	1	-	A=A|[Ri]	3
	A,#data	1	2	-	A=A|#data	4
	direct,A	1	2	-	direct=direct|A	2
	direct,#data	2	3	-	direct=direct|#data	2,4
XRL	A,Rn	1	1	-	A=A^Rn	1
	A,direct	1	2	-	A=A^direct	2
	A,@Ri	1	1	-	A=A^[Ri]	3
	A,#data	1	2	-	A=A^#data	4
	direct,A	1	2	-	direct=direct^A	2
	direct,#data	2	3	-	direct=direct^#data	2,4
MOV	A,Rn	1	1	-	A=Rn	1
	A,direct	1	2	-	A=direct	2
	A,@Ri	1	1	-	A=[Ri]	3
	A,#data	1	2	-	A=#data	4
	Rn,A	1	1	-	Rn=A	1
	Rn,direct	2	2	-	Rn=direct	1,2
	Rn,#data	1	2	-	Rn=#data	1,4
	direct,A	1	2	-	direct=A	2
	direct,Rn	2	2	-	direct=Rn	1,2
	direct,direct	2	3	-	direct1=direct2	2
	direct,@Ri	2	2	-	direct=[Ri]	2,3
	direct,#data	2	3	-	direct=#data	2,4
	@Ri,A	1	1	-	[Ri]=A	3
	@Ri,direct	2	2	-	[Ri]=direct	2,3
	@Ri,#data	1	2	-	[Ri]=#data	3,4

MNEMO	ARGUMENTS	cyc	byte	flags	ACTION	comment
CLR	A	1	1	-	A=0	
CPL	A	1	1	-	A=A^#0FFh	
RL	A	1	1	-	Ai+1=Ai; A0=A7	
RR	A	1	1	-	Ai=Ai+1; A7=A0	
RLC	A	1	1	C	Ai+1=Ai; C=A7; A0=C	
RRC	A	1	1	C	Ai=Ai+1; C=A0; A7=C	
SWAP	A	1	1	-	A3:0=A7:4; A7:4=A3:0	
XCH	A,Rn	1	1	-	Rn=A; A=Rn	1
XCH	A,direct	1	2		direct=A; A=direct	2
XCH	A,@Ri	1	1		[Ri]=A; A=[Ri]	3
XCHD	A,@Ri	1	1		[Ri]3:0=A3:0; A3:0=[Ri]3:0	3
MOVC	A,@A+DPTR	2	1	-	A=code[DPTR+A]	7,8
	A,@A+PC	2	1	-	A=code[PC+A]	7,9
MOVX	A,@Ri	2	1	-	A=xdata[Ri]	3,10
	A,@DPTR	2	1	-	A=xdata[DPTR]	8,10
	@Ri,A	2	1	-	xdata[Ri]=A	3,10
	@DPTR,A	2	1	-	xdata[DPTR]=A	8,10
PUSH	direct	2	2	-	SP=SP+1, [SP]=direct	2,11
POP	direct	2	2	-	direct=[SP], SP=SP-1	2,11
MOV	DPTR,#data	2	3	-	DPTR=#data (16-bit)	8
INC	DPTR	2	1	-	DPTR=DPTR+1 (16-bit)	8
CLR	C	1	1	C=0	C=0	
	bit	1	2	-	bit=0	12
SETB	C	1	1	C=1	C=1	
	bit	1	2	-	bit=1	12
CPL	C	1	1	C	C=C^1	
	bit	1	2	-	bit=bit^1	12
ANL	C,bit	2	2	C	C=C&bit	12
	C,/bit	2	2	C	C=C&(bit^1)	12
ORL	C,bit	2	2	C	C=C|bit	12
	C,/bit	2	2	C	C=C|(bit^1)	12
MOV	C,bit	1	2	C	C=bit	12
	bit,C	2	2	-	bit=C	12
JC	rel	2	2	-	if C=1 PC=PC+rel	9,13
JNC	rel	2	2	-	if C=0 PC=PC+rel	9,13
JB	bit,rel	2	3	-	if bit=1 PC=PC+rel	12,9,13
JBC	bit,rel	2	3	-	if bit=1 PC=PC+rel; bit=0	12,9,13
JNB	bit,rel	2	3	-	if bit=0 PC=PC+rel	12,9,13
JZ	rel	2	2	-	if A=0 PC=PC+rel	9,13
JNZ	rel	2	2	-	if A>0 PC=PC+rel	9,13
DJNZ	Rn,rel	2	2	-	if --Rn=0 PC=PC+rel	9,13,14
DJNZ	direct,rel	2	3	-	if --direct=0 PC=PC+rel	9,13,14
CJNE	A,direct,rel	2	3	C	if A<direct C=1 else C=0 *	9,13,15
CJNE	A,#data,rel	2	3	C	if A<#data C=1 else C=0 *	9,13,15
CJNE	Rn,#data,rel	2	3	C	if Rn<#data C=1 else C=0 *	9,13,15
CJNE	@Ri,#data,rel	2	3	C	if [Ri]<#data C=1 else C=0 *	9,13,15
SJMP	rel	2	2	-	PC=PC+rel	9,13
AJMP	addr11	2	2	-	PC10:0=addr11	9,16
LJMP	addr16	2	3	-	PC=addr16	9
JMP	@A+DPTR	2	1	-	PC=A+DPTR	8,9
ACALL	addr11	2	2	-	[++SP]=PC, PC10:0=addr11	11,9,16,17
LCALL	addr16	2	3		[++SP]=PC, PC=addr16	11,9,17
RET		2	1		PC=[SP--]	11,9,18
RETI		2	1		PC=[SP--]	11,9,18

* CJNE jump description see note 15

8051 instruction set cheatsheet by wek
(C)2005 www.efton.sk

Notes: Operations: **&** - AND, **|** - OR, **^** - exclusive OR, ***** - multiplication, **/** - integer division, **%** - modulo division; **";"** means simultaneous operations, **","** means consecutive operations.

1. **Rn** - registers, n=0..7, as four banks (internal RAM:00h-07h, 08h-0Fh, 10h-17h, 18h-1Fh), depending on bits RS0 and RS1 in PSW (SFR:0D0h).
2. **direct** - directly addressed memory = lower internal RAM area (00h-7Fh) and SFR area (80h-0FF).
3. **[Ri]** - inderectly adressed memory in the lower (00h-7Fh) and upper (80h-0FFh) internal RAM; pointed by content of R0 or R1 register (i=0..1)(see note 1).
4. **#data** - 1 byte of immediate data.
5. **AB** - 16 bit value created by concatenation of content of B (SFR: 0F0h) as HSB and ACC (SFR: 0E0h) as LSB.
6. decimal adjust for BCD arithmetics, to be used after ADD, ADDC and SUBB: if (A3:0>9 or AC=1) A=A+6 (plus setting but not clearing C), if (A7:4>9 or C=1) A=A+60h.
7. **code** - code memory, accessed (read only) by code fetches or MOVC; integrated on chip, and/or external, activated by /PSEN signal; read twice per instruction cycle.
8. **DPTR** - 16-bit data pointer, concatenation of DPH (SFR: 82h) as MSB and DPL (SFR: 83h) as LSB.
9. **PC** - 16-bit program counter, not accessible from program explicitly. Instructions which access it implicitly use an incremented value, pointing to the following instruction.
10. **xdata** - external data memory, accessed by MOVX instructions, activated by /RD and /WR signals. When accessed using [Ri] (see note 3), the upper address is given by current P2.
11. **SP** - stack pointer (SFR: 81h), indirectly adresses internal RAM in the same way as Ri (see note 3) . Incremented before push, decremented after pop (explicit or implicit in call/ret).
12. **bit** - bit address. Bit addresses 0aaaabbb refer to bit bbb of internal RAM 0100aaaa (20h...2Fh); 1aaaabbb refer to bit bbb of SFR 1aaaa000 (80h, 88h, 90h, 99h,...,F0h, F8h).
13. **rel** - relative jump; the 8-bit parameter added to PC (pointing to the followin instruction - see note 9) is treated as a signed integer (-128...127).
14. **DJNZ** - decrement parameter, and if not zero, relative jump (see note 13).
15. **CJNE** - compare parameters, and if not equal, relative jump (see note 13). Simultaneously set carry flag: if the parameter1 < parameter2, C=1, else C=0.
16. **addr11** - jump within a 2kB page. Jump address is given by concatenation of bits 15:11 of PC (see note 9), bits 7:5 of opcode and bits 7:0 of parameter.
17. **CALL** - SP=SP+1, [SP]=PC7:0, SP=SP+1, [SP]=PC15:8, PC=new address
18. **RET (RETI)** - PC15:8=[SP], SP=SP-1, PC7:0=[SP], SP=SP-1