Instruction Set

hex	mnemonic	B/C	hex	mnemonic	B/C	hex	mnemonic	B/C	hex	mnemonic	B/C
00	NOP	1/1	40	JC code addr	2/2*	80	SJMP code addr	2/3	C0	PUSH dir	2/3
01	AJMP code addr	2/3	41	AJMP code addr	2/3	81	AJMP code addr	2/3	C1	AJMP code addr	2/3
02	LJMP code addr	3/3	42	ORL dir, A	2/2	82	ANL C, bit addr	2/2	C2	CLR bit addr	2/2
03	RR A	1/1	43	ORL dir, #data	3/3	83	MOVC A, @A+PC	1/2	C3	CLR C	1/1
04	INC A	1/1	44	ORL A, #data	2/2	84	DIV AB	1/12	C4	SWAP A	1/1
05	INC dir	2/2	45	ORL A, dir	2/2	85	MOV dir, dir	3/3	C5	XCH A, dir	2/2
06	INC @R0	1/1	46	ORL A, @R0	1/1	86	MOV dir, @R0	2/2	C6	XCH A, @R0	1/1
07	INC @R1	1/1	47	ORL A, @R1	1/1	87	MOV dir, @R1	2/2	C7	XCH A, @R1	1/1
08	INC R0	1/1	48	ORL A, R0	1/1	88	MOV dir, R0	2/2	C8	XCH A, R0	1/1
09	INC R1	1/1	49	ORL A, R1	1/1	89	MOV dir, R1	2/2	C9	XCH A, R1	1/1
0A	INC R2	1/1	4A	ORL A, R2	1/1	8A	MOV dir, R2	2/2	CA	XCH A, R2	1/1
0B	INC R3	1/1	4B	ORL A, R3	1/1	8B	MOV dir, R3	2/2	CB	XCH A, R3	1/1
0C	INC R4	1/1	4C	ORL A, R4	1/1	8C	MOV dir, R4	2/2	CC	XCH A, R4	1/1
0D	INC R5	1/1	4D	ORL A, R5	1/1	8D	MOV dir, R5	2/2	CD	XCH A, R5	1/1
0E	INC R6	1/1	4E	ORL A, R6	1/1	8E	MOV dir, R6	2/2	CE	XCH A, R6	1/1
0F	INC R7	1/1	4F	ORL A, R7	1/1	8F	MOV dir, R7	2/2	CF	XCH A, R7	1/1
10	JBC bit addr, code	3/3	50	JNC code addr	2/2*	90	MOV DPTR, #data	3/3	D0	POP dir	2/3
11	ACALL code addr	2/3	51	ACALL code addr	2/3	91	ACALL code addr	2/3	D1	ACALL code addr	2/3
12	LCALL code addr	3/3	52	ANL dir, A	2/1	92	MOV bit addr, C	2/2	D2	SETB bit addr	2/2
13	RRC A	1/1	53	ANL dir, #data	3/3	93	MOVC A, @A+DPTR	1/2	D3	SETB C	1/1
14	DEC A	1/1	54	ANL A, #data	2/2	94	SUBB A, #data	2/2	D4	DA A	1/1
15	DEC dir	2/2	55	ANL A, dir	2/2	95	SUBB A, dir	2/2	D5	DJNZ dir, code addr	3/3*
16	DEC @R0	1/1	56	ANL A, @R0	1/1	96	SUBB A, @R0	1/1	D6	XCHD A, @R0	1/1
17	DEC @R1	1/1	57	ANL A, @R1	1/1	97	SUBB A, @R1	1/1	D7	XCHD A, @R1	1/1
18	DEC R0	1/1	58	ANL A, R0	1/1	98	SUBB A, R0	1/1	D8	DJNZ R0, code addr	2/2*
19	DEC R1	1/1	59	ANL A, R1	1/1	99	SUBB A, R1	1/1	D9	DJNZ R1, code addr	2/2*
1A	DEC R2	1/1	5A	ANL A, R2	1/1	9A	SUBB A, R2	1/1	DA	DJNZ R2, code addr	2/2*
1B	DEC R3	1/1	5B	ANL A, R3	1/1	9B	SUBB A, R3	1/1	DB	DJNZ R3, code addr	2/2*
1C	DEC R4	1/1	5C	ANL A, R4	1/1	9C	SUBB A, R4	1/1	DC	DJNZ R4, code addr	2/2*
1D	DEC R5	1/1	5D	ANL A, R5	1/1	9D	SUBB A, R5	1/1	DD	DJNZ R5, code addr	2/2*
1E	DEC R6	1/1	5E	ANL A, R6	1/1	9E	SUBB A, R6	1/1	DE	DJNZ R6, code addr	2/2*
1F	DEC R7	1/1	5F	ANL A, R7	1/1	9F	SUBB A, R7	1/1	DF	DJNZ R7, code addr	2/2*
20	JB bit addr, code	3/3*	60	JZ code addr	2/2*	A0	ORL C, /bit addr	2/2	E0	MOVX A, @DPTR	1/2
21	AJMP code addr	2/3	61	AJMP code addr	2/3	A1	AJMP code addr	2/3	E1	AJMP code addr	2/3
22	RET	1/3	62	XRL dir, A	2/2	A2	MOV C, bit addr	2/2	E2	MOVX A, @R0	1/2
23	RL A	1/1	63	XRL dir, #data	3/3	A3	INC DPTR	1/1	E3	MOVX A, @R1	1/2
24	ADD A, #data	2/2	64	XRL A, #data	2/2	A4	MUL AB	1/1	E4	CLR A	1/1
25	ADD A, dir	2/2	65	XRL A, dir	2/2	A5	n/a (reserved)	1/1	E5	MOV A, dir	2/2
26	ADD A, @R0	1/1	66	XRL A, @R0	1/1	A6	MOV @R0, dir	2/2**	E6	MOV A, @R0	1/1
27	ADD A, @R1	1/1	67	XRL A, @R1	1/1	A7	MOV @R1, dir	2/2**	E7	MOV A, @R1	1/1
28	ADD A, R0	1/1	68	XRL A, R0	1/1	A8	MOV R0, dir	2/2**	E8	MOV A, R0	1/1
29	ADD A, R1	1/1	69	XRL A, R1	1/1	A9	MOV R1, dir	2/2**	E9	MOV A, R1	1/1
2A	ADD A, R2	1/1	6A	XRL A, R2	1/1	AA	MOV R2, dir	2/2**	EA	MOV A, R2	1/1
2B	ADD A, R3	1/1	6B	XRL A, R3	1/1	AB	MOV R3, dir	2/2**	EB	MOV A, R3	1/1
2C	ADD A, R4	1/1	6C	XRL A, R4	1/1	AC	MOV R4, dir	2/2**	EC	MOV A, R4	1/1
2D	ADD A, R5	1/1	6D	XRL A, R5	1/1	AD	MOV R5, dir	2/2**	ED	MOV A, R5	1/1
2E	ADD A, R6	1/1	6E	XRL A, R6	1/1	AE	MOV R6, dir	2/2**	EE	MOV A, R6	1/1
2F	ADD A, R7	1/1	6F	XRL A, R7	1/1	AF	MOV R7, dir	2/2**	EF	MOV A, R7	1/1
30	JNB bit addr, code	3/3*	70	JNZ code addr	2/2*	B0	ANL C, /bit addr	2/2	F0	MOVX @DPTR, A	1/1
31	ACALL code addr	2/3	71	ACALL code addr	2/3	B1	ACALL code addr	2/3	F1	ACALL code addr	2/3
32	RETI	1/3	72	ORL C, bit addr	2/2	B2	CPL bit addr	2/2	F2	MOVX @R0, A	1/1
33	RLC A	1/1	73	JMP @A+DPTR	1/2	B3	CPL C	1/1	F3	MOVX @R1, A	1/1
34	ADDC A, #data	2/2	74	MOV A, #data	2/2	B4	CJNE A, #data, code	3/3*	F4	CPL A	1/1
35	ADDC A, dir	2/2	75	MOV dir, #data	3/3	B5	CJNE A, dir, code	3/3*	F5	MOV dir, A	2/2
36	ADDC A, @R0	1/1	76	MOV @R0, #data	2/2	B6	CJNE @R0, #data, code	3/3*	F6	MOV @R0, A	1/1
37	ADDC A, @R1	1/1	77	MOV @R1, #data	2/2	B7	CJNE @R1, #data, code	3/3*	F7	MOV @R1, A	1/1
38	ADDC A, R0	1/1	78	MOV R0, #data	2/2	B8	CJNE R0, #data, code	3/3*	F8	MOV R0, A	1/1
39	ADDC A, R1	1/1	79	MOV R1, #data	2/2	B9	CJNE R1, #data, code	3/3*	F9	MOV R1, A	1/1
3A	ADDC A, R2	1/1	7A	MOV R2, #data	2/2	BA	CJNE R2, #data, code	3/3*	FA	MOV R2, A	1/1
3B	ADDC A, R3	1/1	7B	MOV R3, #data	2/2	BB	CJNE R3, #data, code	3/3*	FB	MOV R3, A	1/1
3C	ADDC A, R4	1/1	7C	MOV R4, #data	2/2	BC	CJNE R4, #data, code	3/3*	FC	MOV R4, A	1/1
3D	ADDC A, R5	1/1	7D	MOV R5, #data	2/2	BD	CJNE R5, #data, code	3/3*	FD	MOV R5, A	1/1
3E	ADDC A, R6	1/1	7E	MOV R6, #data	2/2	BE	CJNE R6, #data, code	3/3*	FE	MOV R6, A	1/1
3F	ADDC A, R7	1/1	7F	MOV R7, #data	2/2	BF	CJNE R7, #data, code	3/3*	FF	MOV R7, A	1/1

Figure 1: ***Instruction Set of 8051*****. 'B' ... number of bytes, 'C' ... number of instruction cycles.**

* All conditional jump instructions take one clock cycle more if jump is taken.

e.g. JZ takes 3 clock cycles if jump is taken, 2 clock cycles if not.

** These instructions take one clock cycle more with Generic "fast_cpu" set to 0.

Pipeline-Stalls (Instruction takes one additional clock cycle):

Due to internal instruction pipelining the execution of one instruction is not finished (results written back) when the execution of the next instruction starts. When the second instruction requires the results of the instruction executed before, an additional wait cycle is inserted. As a result the second instruction requires one additional clock cycle to execute.

Ri-Stall: When a write to Register R0 or R1 is in progress and the follwing instruction uses indirect Addressing-mode (@Ri) or accesses external ram (XRAM) one additional clock cycle is required.

Example 1:
```
MOV R0,#BUFFER
MOV A,@R0
```

Example 2:
```
MOV R0,#BUFFER
MOVX A,@R0
```

PSW_Stall: When Register Bank is switched (Bits RS0, RS1 of PSW are written) and the following instruction accesses an register (R0 – R7), one additional clock cycle is required.

Example 1:
```
SETB RS0    ; switch to register bank 1
MOV A,@R0
```

Example 2:
```
MOV PSW,#018H     ; switch to register bank 3
INC R0
```

ACC_Stall: When a write to Accu (ACC) is in progress and the following instruction is JMP @A+DPTR this instruction requires one additional clock cycle.

Example 1:
```
CLR A
JMP @A+DPTR
```

SP_Stall: When a write to Stackpointer (SP) is in progress and the following instruction is a subroutine call (ACALL, LCALL) this instruction requires one additional clock cycle.

Example 1:
```
MOV SP,#STACK
LCALL SUB1
```

MOVX_Stall: When two subsequent MOVX instructions are executed the second instruction takes one additional clock cycle.

Example 1:
```
MOVX A,@DPTR
MOVX @R0,A
```