2507/302
MICROCONTROLLER TECHNOLOGY
Oct./Nov. 2021
Time: 3 hours

THE KENYA NATIONAL EXAMINATIONS COUNCIL

DIPLOMA IN AERONAUTICAL ENGINEERING (AVIONICS OPTION)

MODULE III

MICROCONTROLLER TECHNOLOGY

3 hours

INSTRUCTIONS TO CANDIDATES

You should have the following for this examination:
Answer booklet;
Non-programmable scientific calculator;
Drawing instruments;
Intel 8051 Instruction set.
*Answer any **FIVE** of the **EIGHT** questions in the answer booklet provided.*
Maximum marks for each part of a question are as indicated.
Candidates should answer the questions in English.

This paper consists of 12 printed pages.

Candidates should check the question paper to ascertain that all the pages are printed as indicated and that no questions are missing.

Turn over

1. (a) State the function of each of the following with respect to microcontrollers:

   (i) instruction decoder;
   (ii) arithmetic and logic unit;
   (iii) program counter.

   (3 marks)

   (b) Write an assembly language program to add contents of memory locations 52 H and 53 H. (2 marks)

   (c) Explain the function of each of the following in assembly language programming:

   (i) assembler;
   (ii) debugger;
   (iii) linker.

   (6 marks)

   (d) Convert:

   (i) $(152.41)_8$ to decimal;

   (ii) $(11011.110)_2$ to decimal;

   (iii) $(3F.B2)_{16}$ to octal.

   (9 marks)

2. (a) State **three** ways of increasing the processing speed of a Central Processing Unit (CPU). (3 marks)

   (b) Draw the read instruction cycle for a microcontroller. (4 marks)

   (c) A microcontroller is programmed to monitor the pressure level of compressed air system in an aeroplane. The air supply valve should open as long as the pressure level is above 30 psi otherwise the oxygen masks eject and an alarm raised. Draw a flow chart for the control. (7 marks)

   (d) Perform the following arithmetic operations in the given bases:

   (i) $(AF1.B3)_{16} + (FDF.E0)_{16}$;

   (ii) $(47.34)_8 + (26.53)_8$;

   (iii)
$$\begin{array}{r} (1\,0\,1\,0\,0\,0\,1.1\,0\,1)_2 \\ + \underline{(1\,1\,0\,1\,1\,1\,1.0\,1\,1)_2} \\ \underline{\qquad\qquad\qquad} \end{array}$$

   (6 marks)

3. (a) (i) State **three** merits of using stepper motors in positioning control systems.

(ii) Explain each of the following modes of driving unipolar stepper motor:

(I) full-step drive;

(II) half-step drive.

(7 marks)

(b) Figure 1 shows a block diagram of external ROM memory interfacing with 8051 microcontroller.

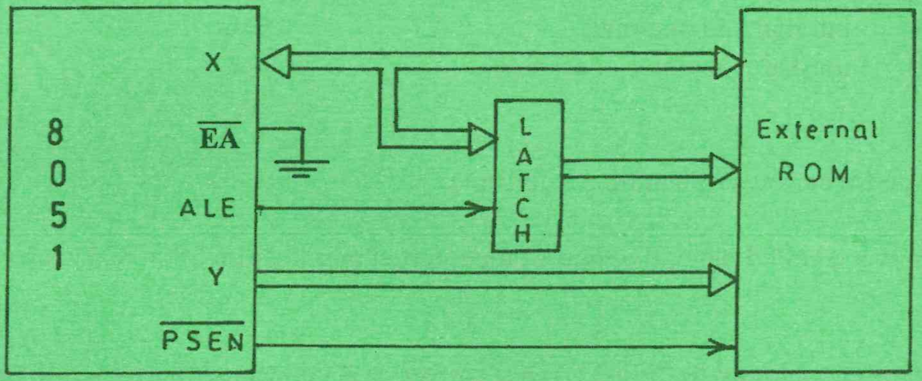

Fig. 1

(i) State the function of each of the following signals:

(I) $\overline{PSEN}$;

(II) ALE.

(ii) Explain the reason for grounding the $\overline{EA}$ pin.

(iii) Identify the I/O ports labelled X and Y, citing their functions.

(7 marks)

(c) Figure 2 shows a circuit diagram of a microcontroller input drive.

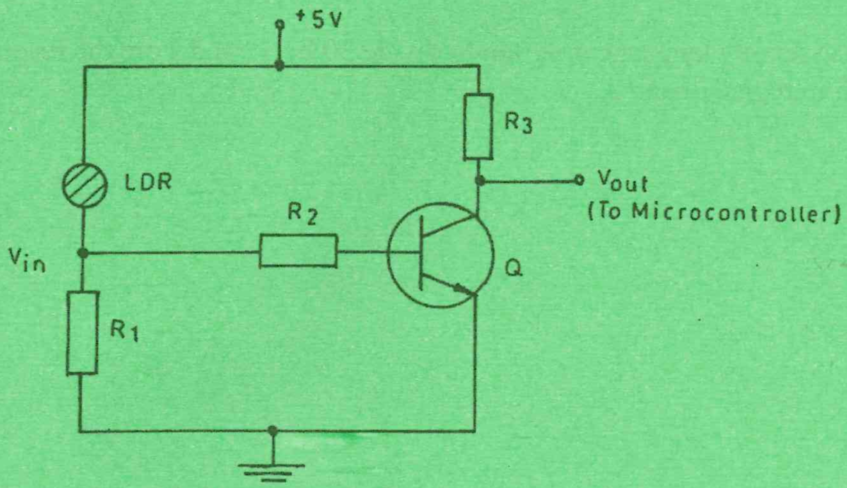

Fig. 2

(i) describe the circuit operation;

(ii) complete the following truth table for the circuit operation.

(6 marks)

V_{in}	V_{out}

4. (a) Differentiate between each of the following types of interrupts:

(i) internal and external;

(ii) maskable and non-maskable.

(4 marks)

(b) State **three** ways of handling interrupts. (3 marks)

(c) Draw a labelled block diagram of the internal architecture of an 8051 microcontroller. (7 marks)

(d) An 8-bit analogue-to-digital (ADC) converter is connected to a $_{+}V_{ref}$ of value 8 V while $_{-}V_{ref}$ is connected to ground. Determine the:

(i) step size;

(ii) data output value when the analogue input voltage is 5 V.

(6 marks)

5. (a) State the special feature function of each of the following 8051 I/O port 3 pins:

(i) P3.1;

(ii) P.3.3;

(iii) P3.5;

(iv) P.3.7.

(4 marks)

(b) Write an assembly language program to divide 30 by 12 and store the result of operation in register bank 2. (6 marks)

(c) Table 1 shows an 8051 assembly language program:

Table 1

```
MOV A, #05H
MOV R0,  # 07 H
MOV R 1, # 55H
ADD A, R0
ADD A, A
ADD A, A
MOV @ R1, A
```

(i) draw the trace table;
(ii) determine the program size in bytes.

(10 marks)

6. (a) State **four** transmission media used to transfer data between the elements of a SCADA system. (4 marks)

(b) With the aid of a labelled block diagram, describe the components of a SCADA system. (8 marks)

(c) A DC motor is controlled by start and stop push button switches. Signal lamps RED and GREEN illuminate when the motor is running or stopped respectively. Using the tags in table 2:

(i) draw a PLC ladder program for the motor control;
(ii) write down the PLC program listing.

(8 marks)

Table 2

Control parameters	Tags
Start	X400
Stop	X401
Motor run	Y430
Green signal lamp	X431
Red signal lamp	Y432

7. (a) Define each of the following as used in process control:

(i) process load;
(ii) process lag.

(2 marks)

(b) Explain the function of each of the following in process control:

(ii) feedback element;
(iii) final control elements;
(iv) error detector.

(6 marks)

(c) Table 3 shows an 8051 assembly language program. The binary value 11010011 is in register A at the beginning of the program. Complete the table to show contents of register A at the end of each instruction execution. (4 marks)

Table 3

Mnemonic	Result of operation
CPL A	
RL A	
RR A	
SWAP A	

(d) Figure 3 shows a ladder diagram program of a programmable logic controller (PLC).

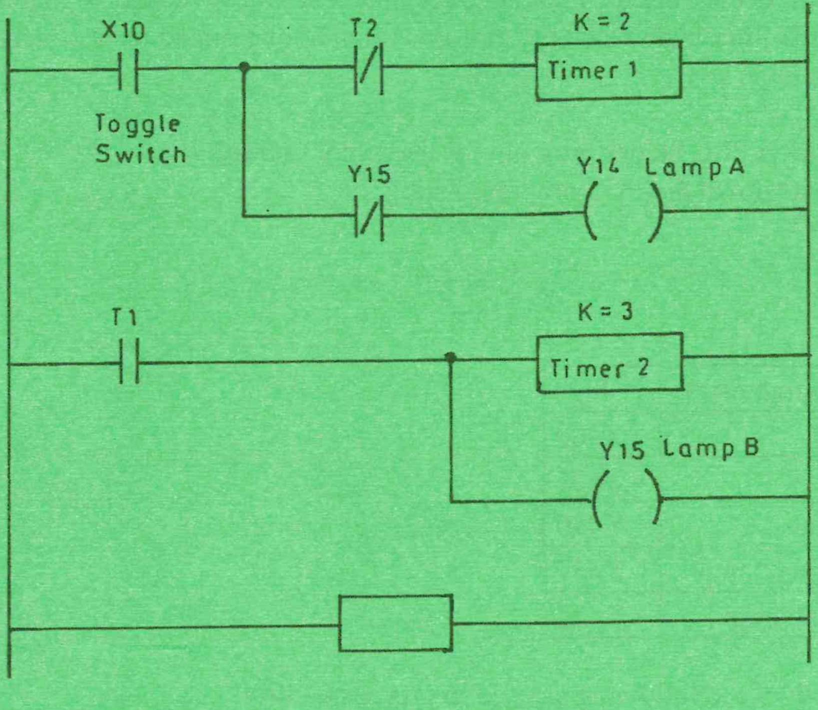

Fig. 3

(i) explain the PLC program operation;
(ii) write the program instruction listing.

(8 marks)

8. (a) State **three** applications of timer circuits in microcontrollers. (3 marks)

(b) Figure 4 shows a block diagram of an 8051 microcontroller timer circuit.

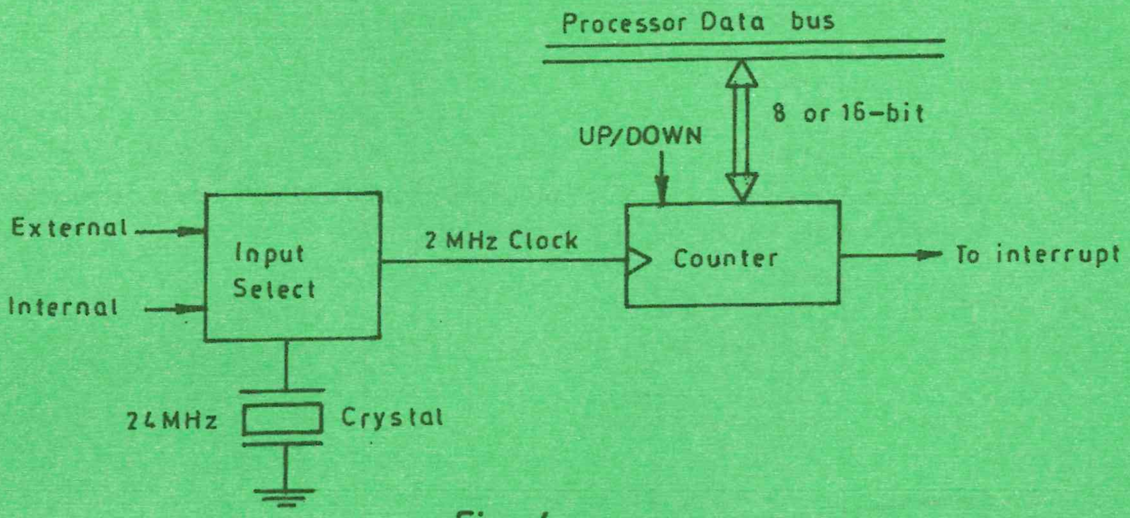

Fig. 4

(i) describe its operation;
(ii) determine the number of instructions executed per second.

(7 marks)

(c) (i) An interrupt occurs when the next instruction to run is "ADD A, @RO". State the registers that may need to be protected by the interrupt service routine.

(ii) State the functions of each of the following 8051 microcontroller instructions:

(I) LCALL;
(II) RET I.

(5 marks)

(d) Figure 5 shows a memory map of 8051 internal RAM.

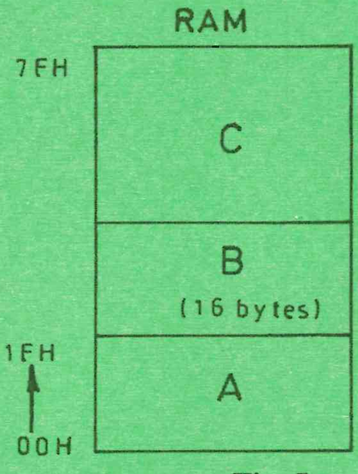

Fig. 5

(i) identify the memories labelled A, B and C;
(ii) state the address range for memories B and C.

(5 marks)

Appendix A: Instruction Set of 8051 Microcontroller

Mnemonics, Arranged Alphabetically

MNEMONIC	DESCRIPTION	BYTES	CYCLES	FLAGS
ACALL addr11	PC + 2 → (SP); addr11 → PC	2	2	
ADD A, direct	A + (direct) → A	2	1	C OV AC
ADD A, @Ri	A + (Ri) → A	1	1	C OV AC
ADD A, #data	A + #data → A	2	1	C OV AC
ADD A, Rn	A + Rn → A	1	1	C OV AC
ADDC A, direct	A + (direct) + C → A	2	1	C OV AC
ADDC A, @Ri	A + (Ri) + C → A	1	1	C OV AC
ADDC A, #data	A + #data + C → A	2	1	C OV AC
ADDC A, Rn	A + Rn + C → A	1	1	C OV AC
AJMP addr11	Addr11 → PC	2	2	
ANL A, direct	A AND (direct) → A	2	1	
ANL A, @Ri	A AND (Ri) → A	1	1	
ANL A, #data	A AND #data → A	2	1	
ANL A, Rn	A AND Rn → A	1	1	
ANL direct, A	(direct) AND A → (direct)	2	1	
ANL direct, #data	(direct) AND #data → (direct)	3	2	
ANL C, bit	C AND bit → C	2	2	C
ANL C, $\overline{\text{bit}}$	C AND $\overline{\text{bit}}$ → C	2	2	C
CJNE A, direct, rel	[A <> (direct)]: PC + 3 + rel → PC	3	2	C
CJNE A, #data, rel	[A <> data]: PC + 3 + rel → PC	3	2	C
CJNE @Ri, #data, rel	[(Ri) <> data]: PC + 3 + rel → PC	3	2	C
CJNE Rn, #data, rel	[Rn <> data]: PC + 3 + rel → PC	3	2	C
CLR A	0 → A	1	1	
CLR bit	0 → bit	2	1	
CLR C	0 → C	1	1	0
CPL A	$\overline{\text{A}}$ → A	1	1	
CPL bit	$\overline{\text{bit}}$ → bit	2	1	
CPL C	$\overline{\text{C}}$ → C	1	1	C
DA A	Abin → Adec	1	1	C
DEC A	A – 1 → A	1	1	
DEC direct	(direct) – 1 → (direct)	2	1	
DEC @Ri	(Ri) – 1 →(Ri)	1	1	
DEC Rn	Rn – 1 → Rn	1	1	
DIV AB	A/B → AB	1	4	0 OV
DJNZ direct, rel	[(direct) – 1 <> 00]: PC + 3 + rel → PC	3	2	
DJNZ Rn, rel	[Rn – 1 <> 00]: PC + 2 + rel → PC	2	2	
INC A	A + 1 → A	1	1	
INC direct	(direct) + 1 → (direct)	2	1	
INC DPTR	DPTR + 1 → DPTR	1	2	
INC @Ri	(Ri) + 1 → (Ri)	1	1	
INC Rn	Rn + 1→ Rn	1	1	
JB bit, rel	[b=1]: PC + 3 + rel → PC	3	2	
JBC bit, rel	[b=1]: PC + 3 + rel → PC; 0 → bit	3	2	
JC rel	[C=1]: PC + 2 + rel → PC	2	2	
JMP @A + DPTR	DPTR + A → PC	1	2	
JNB bit, rel	[b=0]: PC + 3 + rel → PC	3	2	
JNC rel	[C=0]: PC + 2 + rel → PC	2	2	
JNZ rel	[A>00]: PC + 2 + rel → PC	2	2	
JZ rel	[A=00]: PC + 2 + rel → PC	2	2	
LCALL addr16	PC + 3 → (SP); addr16 → PC	3	2	
LJMP addr16	Addr16 → PC	3	2	
MOV A, direct	(direct) → A	2	1	
MOV A, @Ri	(Ri) → A	1	1	
MOV A, #data	#data → A	2	1	
MOV A, Rn	Rn → A	1	1	
MOV direct, A	A → (direct)	2	1	
MOV direct, direct	(direct) → (direct)	3	2	
MOV direct, @Ri	(Ri) → (direct)	2	2	
MOV direct, #data	#data → (direct)	3	2	
MOV direct, Rn	Rn → (direct)	2	2	
MOV bit, C	C → bit	2	2	C
MOV C, bit	bit → C	2	1	
MOV @Ri, A	A → (Ri)	1	1	
MOV @Ri, direct	(direct) → (Ri)	2	2	

MNEMONIC	DESCRIPTION	BYTES	CYCLES	FLAGS
MOV Rn, #data	#data → Rn	2	1	
MOVC A, @A+DPTR	(A+DPTR) → A	1	2	
MOVC A, @A+PC	(A+PC) → A	1	2	
MOVX A, @DPTR	(DPTR)^ → A	1	2	
MOVX A, @Ri	(Ri)^ → A	1	2	
MOVX @DPTR, A	A → (DPTR)^	1	2	
MOVX @Ri, A	A →(Ri)^	1	2	
NOP	PC + 1 → PC	1	1	
MUL AB	A x B → AB	1	4	0 OV
ORL A, direct	A OR (direct) → A	2	1	
ORL A, @Ri	A OR (Ri) → A	1	1	
ORL A, #data	A OR #data → A	2	1	
ORL A, Rn	A OR Rn → A	1	1	
ORL direct, A	(direct) OR A → (direct)	2	1	
ORL direct, #data	(direct) OR #data → (direct)	3	2	
ORL C, bit	C OR bit → C	2	2	C
ORL C, $\overline{bit}$	C OR $\overline{bit}$ → C	2	2	C
POP direct	(SP) → (direct)	2	2	
PUSH direct	(direct) → (SP)	2	2	
RET	(SP) → PC	1	2	
RETI	(SP) → PC; EI	1	2	
RL A	A0←A7←A6..←A1←A0	1	1	
RLC A	C←A7←A6..←A0←C	1	1	C
RR A	A0→A7→A6..→A1→A0	1	1	
RRC A	C→A7→A6..→A0→C	1	1	C
SETB bit	1 → bit	2	1	
SETB C	1 → C	1	1	1
SJMP rel	PC + 2 + rel→ PC	2	2	
SUBB A, direct	A – (direct)–C → A	2	1	C OV AC
SUBB A, @Ri	A – (Ri)–C → A	1	1	C OV AC
SUBB A, #data	A – #data–C → A	2	1	C OV AC
SUBB A, Rn	A – Rn–C → A	1	1	C OV AC
SWAP A	Alsn ↔ Amsn	1	1	
XCH A, direct	A ↔ (direct)	2	1	
XCH A, @Ri	A ↔ (Ri)	1	1	
XCH A, Rn	A ↔ Rn	1	1	
XCHD A, @Ri	Alsn ↔ (Ri)lsn	1	1	
XRL A, direct	A XOR (direct) → A	2	1	
XRL A, @Ri	A XOR (Ri) → A	1	1	
XRL A #data	A XOR #data → A	2	1	
XRL A, Rn	A XOR Rn → A	1	1	
XRL direct, A	(direct) XOR A → (direct)	2	1	
XRL direct, #data	(direct) XOR #data → (direct)	3	2	

ACRONYMS

addr11	Page address of 11 bits, which is in the same 2K page as the address of the following instruction.
addr16	Address for any location in the 64K memory space.
bit	The address of a bit in the internal RAM bit address area or a bit in an SFR.
C	The carry flag.
#data	An 8-bit binary number from 00 to FFh.
#data16	A 16-bit binary number from 0000 to FFFFh.
direct	An internal RAM address or an SFR byte address.
lsn	Least significant nibble
msn	Most significant nibble.
rel	Number that is added to the address of the next instruction to form an address +127d to –128d from the address of the next instruction.
Rn	Any of registers R0 to R7 of the current register bank.
@Ri	Indirect address using the contents of R0 or R1.
[ ]:	IF the condition inside the brackets is *true*, THEN the action listed will occur; ELSE go to the next instruction.
^	EXTERNAL memory location.
( )	Contents of the location inside the parentheses.

Note that flags affected each instruction are shown where appropriate; any operations which affect the PSW address may also affect the flags.

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	**NOP**	
01	2	**AJMP**	code addr
02	3	**LJMP**	code addr
03	1	**RR**	A
04	1	**INC**	A
05	2	**INC**	data addr
06	1	**INC**	@R0
07	1	**INC**	@R1
08	1	**INC**	R0
09	1	**INC**	R1
0A	1	**INC**	R2
0B	1	**INC**	R3
0C	1	**INC**	R4
0D	1	**INC**	R5
0E	1	**INC**	R6
0F	1	**INC**	R7
10	3	**JBC**	bit addr, code addr
11	2	**ACALL**	code addr
12	3	**LCALL**	code addr
13	1	**RRC**	A
14	1	**DEC**	A
15	2	**DEC**	data addr
16	1	**DEC**	@R0
17	1	**DEC**	@R1
18	1	**DEC**	R0
19	1	**DEC**	R1
1A	1	**DEC**	R2
1B	1	**DEC**	R3
1C	1	**DEC**	R4
1D	1	**DEC**	R5
1E	1	**DEC**	R6
1F	1	**DEC**	R7
20	3	**JB**	bit addr, code addr
21	2	**AJMP**	code addr
22	1	**RET**	
23	1	**RL**	A
24	2	**ADD**	A,#data
25	2	**ADD**	A,data addr
26	1	**ADD**	A,@R0
27	1	**ADD**	A,@R1
28	1	**ADD**	A,R0
29	1	**ADD**	A,R1
2A	1	**ADD**	A,R2
2B	1	**ADD**	A,R3
2C	1	**ADD**	A,R4
2D	1	**ADD**	A,R5
2E	1	**ADD**	A,R6
2F	1	**ADD**	A,R7
30	3	**JNB**	bit addr, code addr
31	2	**ACALL**	code addr
32	1	**RETI**	
33	1	**RLC**	A
34	2	**ADDC**	A,#data
35	2	**ADDC**	A,data addr
36	1	**ADDC**	A,@R0
37	1	**ADDC**	A,@R1
38	1	**ADDC**	A,R0
39	1	**ADDC**	A,R1
3A	1	**ADDC**	A,R2
3B	1	**ADDC**	A,R3
3C	1	**ADDC**	A,R4
3D	1	**ADDC**	A,R5
3E	1	**ADDC**	A,R6
3F	1	**ADDC**	A,R7
40	2	**JC**	code addr
41	2	**AJMP**	code addr
42	2	**ORL**	data addr,A
43	3	**ORL**	data addr,#data
44	2	**ORL**	A,#data
45	2	**ORL**	A,data addr
46	1	**ORL**	A,@R0
47	1	**ORL**	A,@R1
48	1	**ORL**	A,R0
49	1	**ORL**	A,R1
4A	1	**ORL**	A,R2
4B	1	**ORL**	A,R3
4C	1	**ORL**	A,R4
4D	1	**ORL**	A,R5
4E	1	**ORL**	A,R6
4F	1	**ORL**	A,R7
50	2	**JNC**	code addr
51	2	**ACALL**	code addr
52	2	**ANL**	data addr,A
53	3	**ANL**	data addr,#data
54	2	**ANL**	A,#data
55	2	**ANL**	A,data addr
56	1	**ANL**	A,@R0
57	1	**ANL**	A,@R1
58	1	**ANL**	A,R0
59	1	**ANL**	A,R1
5A	1	**ANL**	A,R2
5B	1	**ANL**	A,R3
5C	1	**ANL**	A,R4
5D	1	**ANL**	A,R5
5E	1	**ANL**	A,R6
5F	1	**ANL**	A,R7
60	2	**JZ**	code addr
61	2	**AJMP**	code addr

8051 OpCodes en Hexadecinal.

62	2	**XRL**	data addr,A
63	3	**XRL**	data addr,#data
64	2	**XRL**	A,#data
65	2	**XRL**	A,data addr
66	1	**XRL**	A,@R0
67	1	**XRL**	A,@R1
68	1	**XRL**	A,R0
69	1	**XRL**	A,R1
6A	1	**XRL**	A,R2
6B	1	**XRL**	A,R3
6C	1	**XRL**	A,R4
6D	1	**XRL**	A,R5
6E	1	**XRL**	A,R6
6F	1	**XRL**	A,R7
70	2	**JNZ**	code addr
71	2	**ACALL**	code addr
72	2	**ORL**	C,bit addr
73	1	**JMP**	@A+DPTR
74	2	**MOV**	A,#data
75	3	**MOV**	data addr,#data
76	2	**MOV**	@R0,#data
77	2	**MOV**	@R1,#data
78	2	**MOV**	R0,#data
79	2	**MOV**	R1,#data
7A	2	**MOV**	R2,#data
7B	2	**MOV**	R3,#data
7C	2	**MOV**	R4,#data
7D	2	**MOV**	R5,#data
7E	2	**MOV**	R6,#data
7F	2	**MOV**	R7,#data
80	2	**SJMP**	code addr
81	2	**AJMP**	code addr
82	2	**ANL**	C,bit addr
83	1	**MOVC**	A,@A+PC
84	1	**DIV**	AB
85	3	**MOV**	data addr, data addr
86	2	**MOV**	data addr,@R0
87	2	**MOV**	data addr,@R1
88	2	**MOV**	data addr,R0
89	2	**MOV**	data addr,R1
8A	2	**MOV**	data addr,R2
8B	2	**MOV**	data addr,R3
8C	2	**MOV**	data addr,R4
8D	2	**MOV**	data addr,R5
8E	2	**MOV**	data addr,R6
8F	2	**MOV**	data addr,R7
90	3	**MOV**	DPTR,#data
91	2	**ACALL**	code addr
92	2	**MOV**	bit addr,C
93	1	**MOVC**	A,@A+DPTR
94	2	**SUBB**	A,#data

95	2	**SUBB**	A,data addr
96	1	**SUBB**	A,@R0
97	1	**SUBB**	A,@R1
98	1	**SUBB**	A,R0
99	1	**SUBB**	A,R1
9A	1	**SUBB**	A,R2
9B	1	**SUBB**	A,R3
9C	1	**SUBB**	A,R4
9D	1	**SUBB**	A,R5
9E	1	**SUBB**	A,R6
9F	1	**SUBB**	A,R7
A0	2	**ORL**	C,/bit addr
A1	2	**AJMP**	code addr
A2	2	**MOV**	C,bit addr
A3	1	**INC**	DPTR
A4	1	**MUL**	AB
A5		**reserved**	
A6	2	**MOV**	@R0,data addr
A7	2	**MOV**	@R1,data addr
A8	2	**MOV**	R0,data addr
A9	2	**MOV**	R1,data addr
AA	2	**MOV**	R2,data addr
AB	2	**MOV**	R3,data addr
AC	2	**MOV**	R4,data addr
AD	2	**MOV**	R5,data addr
AE	2	**MOV**	R6,data addr
AF	2	**MOV**	R7,data addr
B0	2	**ANL**	C,/bit addr
B1	2	**ACALL**	code addr
B2	2	**CPL**	bit addr
B3	1	**CPL**	C
B4	3	**CJNE**	A,#data,code addr
B5	3	**CJNE**	A,data addr,code addr
B6	3	**CJNE**	@R0,#data,code addr
B7	3	**CJNE**	@R1,#data,code addr
B8	3	**CJNE**	R0,#data,code addr
B9	3	**CJNE**	R1,#data,code addr
BA	3	**CJNE**	R2,#data,code addr
BB	3	**CJNE**	R3,#data,code addr
BC	3	**CJNE**	R4,#data,code addr
BD	3	**CJNE**	R5,#data,code addr
BE	3	**CJNE**	R6,#data,code addr
BF	3	**CJNE**	R7,#data,code addr
C0	2	**PUSH**	data addr
C1	2	**AJMP**	code addr
C2	2	**CLR**	bit addr
C3	1	**CLR**	C
C4	1	**SWAP**	A
C5	2	**XCH**	A,data addr
C6	1	**XCH**	A,@R0
C7	1	**XCH**	A,@R1

8051 OpCodes en Hexadecinal.

C8	1	**XCH**	A,R0
C9	1	**XCH**	A,R1
CA	1	**XCH**	A,R2
CB	1	**XCH**	A,R3
CC	1	**XCH**	A,R4
CD	1	**XCH**	A,R5
CE	1	**XCH**	A,R6
CF	1	**XCH**	A,R7
D0	2	**POP**	data addr
D1	2	**ACALL**	code addr
D2	2	**SETB**	bit addr
D3	1	**SETB**	C
D4	1	**DA**	A
D5	3	**DJNZ**	data addr,code addr
D6	1	**XCHD**	A,@R0
D7	1	**XCHD**	A,@R1
D8	2	**DJNZ**	R0,code addr
D9	2	**DJNZ**	R1,code addr
DA	2	**DJNZ**	R2,code addr
DB	2	**DJNZ**	R3,code addr
DC	2	**DJNZ**	R4,code addr
DD	2	**DJNZ**	R5,code addr
DE	2	**DJNZ**	R6,code addr
DF	2	**DJNZ**	R7,code addr
E0	1	**MOVX**	A,@DPTR
E1	2	**AJMP**	code addr
E2	1	**MOVX**	A,@R0
E3	1	**MOVX**	A,@R1
E4	1	**CLR**	A
E5	2	**MOV**	A,data addr
E6	1	**MOV**	A,@R0
E7	1	**MOV**	A,@R1
E8	1	**MOV**	A,R0
E9	1	**MOV**	A,R1
EA	1	**MOV**	A,R2
EB	1	**MOV**	A,R3
EC	1	**MOV**	A,R4
ED	1	**MOV**	A,R5
EE	1	**MOV**	A,R6
EF	1	**MOV**	A,R7
F0	1	**MOVX**	@DPTR,A
F1	2	**ACALL**	code addr
F2	1	**MOVX**	@R0,A
F3	1	**MOVX**	@R1,A
F4	1	**CPL**	A
F5	2	**MOV**	data addr,A
F6	1	**MOV**	@R0,A
F7	1	**MOV**	@R1,A
F8	1	**MOV**	R0,A
F9	1	**MOV**	R1,A
FA	1	**MOV**	R2,A
FB	1	**MOV**	R3,A
FC	1	**MOV**	R4,A
FD	1	**MOV**	R5,A
FE	1	**MOV**	R6,A
FF	1	**MOV**	R7,A

Instruction Opcodes in Hexadecimal Order

8051 OpCodes en Hexadecinal.

THIS IS THE LAST PRINTED PAGE.