

	OPCODE	Mnemonic			
0	00000	NOP			
1	00001	AND $R_z \leftarrow R_x \& R_y$	(ALU)	Z	
2	00010	OR $R_z \leftarrow R_x R_y$	(ALU)	Z	
3	00011	EXOR $R_z \leftarrow R_x \oplus R_y$	(ALU)	Z	
4	00100	ADD $R_z \leftarrow R_x + R_y$	(ALU)	Z, C	
5	00101	ANDI $R_z \leftarrow R_x \& \text{ImmData}$	(ALU)	Z	
6	00110	ORI $R_z \leftarrow R_x \text{ImmData}$	(ALU)	Z	
7	00111	EXORI $R_z \leftarrow R_x \oplus \text{ImmData}$	(ALU)	Z	
8	01000	ADDI $R_z \leftarrow R_x + \text{ImmData}$	(ALU)	Z, C	
9	01001	MOVI $R_z \leftarrow R_x + 0$	(ALU)	Z, C	
10	01010	MOVI $R_z \leftarrow 0 + \text{ImmData}$	(ALU)		
11	01011	Load $R_z \leftarrow M[\text{Address}]$			
12	01100	Store $M[\text{Address}] \leftarrow R_x + 0$	(ALU)		
13	01101	JMP Jump to Code Address			
14	01110	JMPZ Jump if Z=1			
15	01111	JMPNZ Jump if Z=0			
16	10000	JMPC C=1			
17	10001	JMPNC C=0			
18	10010	PUSH $\text{Stack} \leftarrow R_x + 0$	(ALU)		
19	10011	POP $R_z \leftarrow \text{Stack}$			
20	10100	IN $R_z \leftarrow \text{Port}[\text{Address}]$			
21	10101	OUT $\text{Port}[\text{Address}] \leftarrow R_x + 0$	(ALU)		
22	10110	Load Indirect $R_z \leftarrow M[R_x]$			
23	10111	Store Indirect $M[R_x] \leftarrow R_x + 0$	(ALU)		
24	11000	SUB $R_z \leftarrow R_x - R_y$	(ALU)	Z, C	
25	11001	Shift Right $R_z \leftarrow R_x \gg R_y$	(ALU)	Z, C, R	
26	11010	Shift Left $R_z \leftarrow R_x \ll R_y$	(ALU)	Z, C, R	
27	11011	JMPZ PC Relative Jump to (PC + Offset)	(ALU)		
28	11100	JMPNZ PC Relative " " " "	(ALU)		
29					
30					
31					

Sr. No.	OPCODE	
0	00000	NOP
1	00001	AND
2	00010	OR
3	00011	EXOR
4	00100	ADD
5	00101	ANDI
6	00110	ORI
7	00111	EXORI
8	01000	ADDI
9	01001	MOVI
10	01010	MOVI
11	01011	Load
12	01100	Store
13	01101	JMP
14	01110	JMPZ
15	01111	JMPNZ
16	10000	JMPC
17	10001	JMPNC
18	10010	PUSH
19	10011	POP
20	10100	IN
21	10101	OUT
22	10110	Load Indirect
23	10111	Store Indirect
24	11000	SUB
25	11001	Shift Right
26	11010	Shift Left
27	11011	JMPZ Relative
28	11100	JMPNZ Relative
29	11101	
30	11110	
31	11111	

How to write in assembler? (Example given below) Comments

NOP;
 AND, R12, R2, R5;
 OR, R3, R15, R6;
 EXOR, R3, R15, R6;
 ADD, R13, R15, R2;
 ANDI, R3, R15, 77;
 ORI, R3, R15, 88;
 EXORI, R3, R15, AB;
 ADDI, R10, R15, FC;
 MOV, R12, R7;
 MOVI, R14, AC;
 LOAD, R10, 81;
 STORE, 83, R10;
 JMP, AB;
 JMPZ, AC;
 JMPNZ, AD;
 JMPC, AE;
 JMPNC, AF;
 PUSH, R12;
 POP, R13;
 IN, R11, F1;
 OUT, F2, R14;
 LOADI, R10, R2;
 STOREI, R10, R2;
 SUB, R12, R13, R14;
 SHIFTR, R12, R13, R14;
 SHIFTL, R12, R13, R14;
 JMPPCRZ, AD;
 JMPPCRNZ, AE;

Means Load R10 <--- M[81H]
 Means LoadI M[83H] <--- R10
 Means JMP to address ABH in the instruction memory
 Means PUSH Stack <--- R12
 Means POP R13 <--- Stack
 Means IN R11 <--- P[F1H]
 Means OUT P[F2H] <--- R14
 Means LoadI R10 <--- M[R2]
 Means StoreI M[R2] <--- R10
 Means Shift Right R12 <--- R13 >> R14