Instruction	OpCode	Oper 1	Oper 2	Description	Sequence	Clock Cycles
Transfer Operations						
"Fetch"				Fetch next instruction	M[PC] → <inst dec> PC + 1 → PC	1
PSH #dd	0x11	ACL		Push direct data to TOS	M[PC] → ACL PC + 1 → PC SP - 1 → SP ACL → M[SP]	2
PSA	0x12			Push AC to TOS	SP - 1 → SP ACL → M[SP]	1
POP	0x14			Pop TOS	M[SP] → ACL SP + 1 → SP	1
LDM $hhhh	0x21	DRH	DRL	Load from data memory to TOS	M[PC] → DRH PC + 1 → PC M[PC] → DRL PC + 1 → PC M[DR] → ACL SP - 1 → SP ACL → M[SP] PC → PC	4
LDI	0x22			Increment DR, load data mem to TOS	DR + 1 → DR M[DR] → ACL SP – 1 → SP ACL → M[SP]	2
STM $hhhh	0x31	DRH	DRL	Store TOS to data memory	M[PC] → DRH PC + 1 → PC M[PC] → DRL PC + 1 → PC M[SP] → ACL SP + 1 → SP ACL → M[DR] PC → PC	4

STI	0x32			Increment DR, store TOS to data mem	DR + 1 → DR M[SP] → ACL SP + 1 → SP ACL → M[DR]	2
Math Operations						
ADD	0x41			Adds top two stack values, push sum to TOS	M[SP] → temp SP + 1 → SP M[SP] → ACL SP + 1 → SP AC + temp → AC SP - 1 → SP ACL → M[SP] PC → PC	4
SUB	0x42			Subtracts top two stack values, push diff to TOS	M[SP] → temp SP + 1 → SP M[SP] → ACL SP + 1 → SP AC - temp → AC SP - 1 → SP ACL → M[SP] PC → PC	4
NEG	0x44			Negates TOS	M[SP] → ACL SP + 1 → SP 0 → temp temp - AC → AC SP - 1 → SP ACL → M[SP]	3
LSR	0x48			Logical shift the TOS right by one bit	M[SP] → ACL SP + 1 → SP AC >> 1 → AC SP - 1 → SP ACL → M[SP] PC → PC	3

LSL	0x4A			Logical shift the TOS left by one bit	M[SP] → ACL SP + 1 → SP AC << 1 → AC SP - 1 → SP ACL → M[SP] PC → PC	3
Logic Operations						
AND #dd	0x51	ACL		ANDs TOS with direct data, push result to TOS	M[PC] → temp PC + 1 → PC M[SP] → ACL SP + 1 → SP AC & temp → AC SP - 1 → SP ACL → M[SP] PC → PC	4
ORR #dd	0x52	ACL		ORs TOS with direct data, push result to TOS	M[PC] → temp PC + 1 → PC M[SP] → ACL SP + 1 → SP AC | temp → AC SP - 1 → SP ACL → M[SP] PC → PC	4
XOR #dd	0x54	ACL		XORs TOS with direct data, push result to TOS	M[PC] → temp PC + 1 → PC M[SP] → ACL SP + 1 → SP AC ^ temp → AC SP - 1 → SP ACL → M[SP] PC → PC	4

INV	0x58			Inverts TOS, replace TOS with result	M[SP] → temp SP + 1 → SP invert temp → AC SP - 1 → SP ACL → M[SP] PC → PC	3
Compare / Branch Operations						
CPE #dd	0x61	ACL		Compare if TOS is equal to direct data (TOS will be '0' if equal)	M[PC] → temp PC + 1 → PC M[SP] → ACL SP + 1 → SP SP - 1 → SP ACL → M[SP] AC xor temp → AC SP - 1 → SP ACL → M[SP] PC → PC	5
CNE #dd	0x62	ACL		Compare if TOS is not equal to direct data (TOS will be not '0' if not equal)	M[PC] → temp PC + 1 → PC M[SP] → ACL SP + 1 → SP SP - 1 → SP ACL → M[SP] AC xor temp → AC invert AC -> AC SP - 1 → SP ACL → M[SP]	5

BRZ &label	0x71	PCH	PCL	Branch if TOS is zero, stack is popped	M[PC] - > DRH PC + 1 → PC M[PC] → DRL PC + 1 → PC M[SP] → AC SP + 1 → SP If AC = 0 DR → PC Else PC → PC	4
BRN &label	0x72	PCH	PCL	Branch if TOS is not zero, stack is popped	M[PC] - > DRH PC + 1 → PC M[PC] → DRL PC + 1 → PC M[SP] → AC SP + 1 → SP If AC != 0 DR → PC Else PC → PC	4
BRU &label	0x74	PCH	PCL	Branch unconditionally	M[PC] - > DRH PC + 1 → PC M[PC] → DRL PC + 1 → PC DR → PC PC → PC	3
I/O Operations						
INP	0x81			Input IR to TOS	IR → AC SP - 1 → SP ACL → M[SP] PC → PC	2
OUT	0x82			Output TOS to OR, stack is popped	M[SP] → ACL SP + 1 → SP AC → OR PC → PC	2

SER	0x84			Input SR to TOS	SR → AC SP - 1 → SP ACL → M[SP] PC → PC	2
PRT	0x88			Print TOS to PR, stack is popped	M[SP] → ACL SP + 1 → SP AC → PR PC → PC	2
Special Operations						
NOP	0x90			No operation	PC → PC PC → PC	1
CLS	0x92			Clear the stack	<mem top> → SP PC → PC	1
END	0x98			End of program (aka halt)	PC – 1 → PC PC → PC	1
RST	0x9F			Reset CPU	0 → AC 0 → OUT 0 → PRT <data start> → DR <mem top> → SP <mem bot> → PC	3