Instr	Description	Action	Examples	Clock	Class	OpCode
	Transfer Instructions					
mov *d*,*s*	8bit Copy Reg-Reg Copies the source register *s* to the destination register *d*. *d* = a/b/c/d/m1/m2/x/y *s* = a/b/c/d/m1/m2/x/y	*d* <- *s*	mov a,b mov c,x mov m1,a	8	MOV-8	00*dddsss*
mov *d*,*s*	16bit Copy Reg-Reg Copies the source register *s* to the destination register *d*. *d* = xy/pc *s* = m/xy/j	*d* <- *s*	mov xy,m mov pc,j	12	MOV-16	1010*dss*0
ldi *d*,*i*	8bit Load Immediate Loads a constant to the destination register *d*. *d* = a/b *i* = -16..15	*d* <- *i*	ldi a,5 ldi a,101b ldi b,7h	8	SETAB	01*diiiii*
ldi m,*a*	16bit Load Immediate Loads a constant to the m register. *a* = 0000h..FFFFh	m <- *i*	ldi m,23EAh ldi m,AF37h	24	GOTO	11000000 *aaaaaaaa* *aaaaaaaa*
ldr *d*	8bit Load Direct SRAM-Reg Loads the destination register *d* from SRAM pointed at by the m register. *d* = a/b/c/d	*d* <- (m)	ldr a ldr c	12	LOAD	100100*dd*
str *s*	8bit Store Reg-SRAM Store the contents of the source register *s* to SRAM pointed at by the m register. *s* = a/b/c/d	(m) <- *s*	str b str d	12	STORE	100110*ss*
	ALU Instructions					
add	ALU Add Adds the contents of registers b and c storing the result in register a.	a <- b + c	add	8	ALU	10000000
and	ALU Bitwise AND Performs a bitwise AND on the contents of registers b and c storing the result in register a.	a <- b & c	and	8	ALU	10000010
clr *d*	ALU Clear Clears the destination register *d*. *d* = a/d	*d* <- 0	clr a clr d	8	ALU	1000d111
eor	ALU Bitwise XOR Performs a bitwise XOR on the contents of registers b and c storing the result in register a.	a <- b ^ c	eor	8	ALU	10000100
inc	ALU Increment Adds one to the contents of register b storing the result in register a.	a <- b + 1	inc	8	ALU	10000001
not	ALU Bitwise NOT Performs a bitwise NOT on the contents of register b storing the result in register a.	a <- ~b	not	8	ALU	10000101
orr	ALU Bitwise OR Performs a bitwise OR on the contents of registers b and c storing the result in register a.	a <- b | c	orr	8	ALU	10000011
rol	ALU Bitwise Rotate Left Performs a cyclic left shift (rotate) on the contents of register b storing the result in register a.	a <- <<b	rol	8	ALU	10000110
	Misc Instructions					
ixy	16bit XY Increment Increments the XY register.	xy <- xy + 1	ixy	14	INC-XY	10110000
hlt	Halt Clears the pc register and halts execution.	pc = 0 -stop-	hlt	12	MOV-16	10101110
	Control Flow Instructions					
b *a*	Unconditional Branch Loads the address *a* into the j register then unconditionally branches to that address. *a* = 0000h..FFFFh	j = *a* pc = j	b F4E9h	24	GOTO	11100110 *aaaaaaaa* *aaaaaaaa*
bl *a*	Unconditional Branch and Link Loads the address *a* into the j register, stores the next instruction location in the xy register then unconditionally branches to the that address. *a* = 0000h..FFFFh	j = *a* xy = pc pc = j	call f9h	24	GOTO	11100111 *aaaaaaaa* *aaaaaaaa*
bcc *a*	Conditional Branch Loads the address *a* into the j register then branches to that address if condition *c* is met. *a* = 0000h..FFFFh *c* = al - always (Z==0/Z==1) eq - equal (Z==0) ne - nonequal (Z==1) cc - carry clear (Cy==0) mi - minus (S==1)	j = *a* (pc = j)	bal 6A25h beq A25Eh bne 25E0h bcc 5E0Bh bmi E0B9h	24	GOTO	111*cccc*0 *aaaaaaaa* *aaaaaaaa*
blcc *a*	Conditional Branch and Link Loads the address *a* into the j register, stores the next instruction location in the xy register then branches to that address if condition *c* is met. *a* = 0000h..FFFFh *c* = al - always (Z==0/Z==1) eq - equal (Z==0) ne - nonequal (Z==1) cc - carry clear (Cy==0) mi - minus (S==1)	j = *a* xy = pc (pc = j)	blal 6A25h bleq A25Eh blne 25E0h blcc 5E0Bh blmi E0B9h	24	GOTO	111*cccc*1 *aaaaaaaa* *aaaaaaaa*
bx	Branch Indirect (Return) Unconditionally branches to the address stored in the xy register.	pc = xy	bx	12	MOV-16	10101010