Type R-I

Format	Instruction	Operation	Cycles	Description
rd rd rd rd i i i i i i i i 0 0 0 1	LDI rd, i	rd <= i	1	Load Immediate
rd rd rd rd i i i i i i i i 0 0 1 0	ANI rd, i	rd <= rd & i	1	AND Immediate
rd rd rd rd i i i i i i i i 0 0 1 1	ORI rd, i	rd <= rd | i	1	OR Immediate
rd rd rd rd i i i i i i i i 0 1 0 0	XOI rd, i	rd <= rd ^ i	1	XOR Immediate
rd rd rd rd i i i i i i i i 0 1 0 1	ADI rd, i	rd <= rd + i	1	ADD Immediate
rd rd rd rd i i i i i i i i 0 1 1 0	ACI rd, i	rd <= rd + i + C	1	ADD Immediate With Carry
rd rd rd rd i i i i i i i i 0 1 1 1	CPI rd, i	See Notes	1	Compare Immediate

Type R-IO

Format	Instruction	Operation	Cycles	Description
rd rd rd rd io io io io io io io io 1 0 0 0	IN rd, io	rd <= *io	2	IO In
rs rs rs rs io io io io io io io io 1 0 0 1	OUT rs, io	*io <= rs	1	IO Out

Type R-R

Format	Instruction	Operation	Cycles	Description
rd rd rd rd rs rs rs rs 0 0 0 1 1 0 1 0	MOV rd, rs	rd <= rs	1	Move
rd rd rd rd rs rs rs rs 0 0 1 0 1 0 1 0	AND rd, rs	rd <= rd & rs	1	AND
rd rd rd rd rs rs rs rs 0 0 1 1 1 0 1 0	OR rd, rs	rd <= rd | rs	1	OR
rd rd rd rd rs rs rs rs 0 1 0 0 1 0 1 0	XOR rd, rs	rd <= rd ^ rs	1	XOR
rd rd rd rd rs rs rs rs 0 1 0 1 1 0 1 0	ADD rd, rs	rd <= rd + rs	1	ADD
rd rd rd rd rs rs rs rs 0 1 1 0 1 0 1 0	ADC rd, rs	rd <= rd + rs + C	1	ADD With Carry
rd rd rd rd rs rs rs rs 0 1 1 1 1 0 1 0	CMP rd, rs	See Notes	1	Compare
rd rd rd rd rs rs rs rs 1 0 0 0 1 0 1 0	SUB rd, rs	rd <= rd + ~rs + 1	1	Subtract
rd rd rd rd rs rs rs rs 1 0 0 1 1 0 1 0	SBB rd, rs	rd <= rd + ~rs + C	1	Subtract With Borrow

Type R-P

Format	Instruction	Operation	Cycles	Description
rs rs rs rs p p p 0 1 1 0 0 1 0 1 0	SRI rs, p	*p <= rs, p <= p + 1	1	Store, Post Increment
rs rs rs rs p p p 0 1 1 0 1 1 0 1 0	SRD rs, p	*p <= rs, p <= p - 1	1	Store, Post Decrement
rd rd rd rd p p p 0 1 1 1 0 1 0 1 0	LRI rd, p	rs <= *p, p <= p + 1	2	Load, Post Increment
rd rd rd rd p p p 0 1 1 1 1 1 0 1 0	LRD rd, p	rs <= *p, p <= p - 1	2	Load, Post Decrement

Type R-P-K

Format	Instruction	Operation	Cycles	Description
rs rs rs rs p p p k k k k k 1 0 1 1	STR rs, p, k	*(p + k) <= rs	1	Store, With Offset
rd rd rd rd p p p k k k k k 1 1 0 0	LDR rd, p, k	rd <= *(p + k)	2	Load, With Offset

Type P-I

Format	Instruction	Operation	Cycles	Description
i i i i p p p i i i i i 1 1 0 1	API p, i	p <= p + i	1	Add Pair Immediate

Type Branch

Format	Instruction	Operation	Cycles	Description
0 0 0 a a a a a a a a a 1 1 1 0	BR	pc <= pc + a	1	Branch
0 0 1 a a a a a a a a a 1 1 1 0	BC a	pc <= pc + a, if c	1	Branch Carry
0 1 0 a a a a a a a a a 1 1 1 0	BNC a	pc <= pc + a, if !c	1	Branch No Carry
0 1 1 a a a a a a a a a 1 1 1 0	BZ a	pc <= pc + a, if z	1	Branch Zero
1 0 0 a a a a a a a a a 1 1 1 0	BNZ a	pc <= pc + a, if !z	1	Branch No Zero
1 0 1 a a a a a a a a a 1 1 1 0	BN a	pc <= pc + a, if n	1	Branch Negative
1 1 0 a a a a a a a a a 1 1 1 0	BNN a	pc <= pc + a, if !n	1	Branch No Negative

Type R

Format	Instruction	Operation	Cycles	Description
rd rd rd rd 0 0 0 0 1 0 1 0 1 1 1 1	SLL rd	rd <= rd << 1	1	Shift Left Logical
rd rd rd rd 0 0 0 0 1 0 1 1 1 1 1 1	SRL rd	rd <= rd >> 1	1	Shift Right Logical
rd rd rd rd 0 0 0 0 1 1 0 0 1 1 1 1	SRA rd	rd <= rd >>> 1	1	Shift Right Arithmetic
rd rd rd rd 0 0 0 0 1 1 0 1 1 1 1 1	RLC rd	{c,rd} <= {rd,c}	1	Rotate Left Through Carry
rd rd rd rd 0 0 0 0 1 1 1 0 1 1 1 1	RRC rd	{rd,c} <= {c,rd}	1	Rotate Right Through Carry
rd rd rd rd 0 0 0 0 1 1 1 1 1 1 1 1	NOT rd	rd <= ~rd	1	NOT
rd rd rd rd 1 1 1 0 0 0 1 0 0 0 0 0	POP rd	rd <= stack	2	POP

Type P

Format	Instruction	Operation	Cycles	Description
0 0 0 0 p p p 0 0 0 0 1 1 1 1 1	JMPI, p	PC <= p	1	Jump Indirect
0 0 1 0 p p p 0 0 0 0 1 1 1 1 1	JCI, p	PC <= p, if c	1	Jump Carry Indirect
0 1 0 0 p p p 0 0 0 0 1 1 1 1 1	JNCI, p	PC <= p, if !c	1	Jump Not Carry Indirect
0 1 1 0 p p p 0 0 0 0 1 1 1 1 1	JZI, p	PC <= p, if z	1	Jump Zero Indirect
1 0 0 0 p p p 0 0 0 0 1 1 1 1 1	JNZI, p	PC <= p, if !z	1	Jump Not Zero Indirect
1 0 1 0 p p p 0 0 0 0 1 1 1 1 1	JNI, p	PC <= p, if n	1	Jump Negative Indirect
1 1 0 0 p p p 0 0 0 0 1 1 1 1 1	JNNI, p	PC <= p, if !n	1	Jump Not Negative Indirect

Type A

Format	Instruction	Operation	Cycles	Description
0 0 0 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JMP a	pc <= a	2	Jump
0 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JC a	pc <= a, if c	1 or 2	Jump Carry
0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JNC a	pc <= a, if !c	1 or 2	Jump Not Carry
0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JZ a	pc <= a, if z	1 or 2	Jump Zero
1 0 0 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JNZ a	pc <= a, if !z	1 or 2	Jump Not Zero
1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JN a	pc <= a, if n	1 or 2	Jump Negative
1 1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 a a a a a a a a a a a a a a a a	JNN a	pc <= a, if !n	1 or 2	Jump Not Negative
0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CALL a	pc <= a, stack <= pc	2	Call
0 0 1 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CC a	(pc <= a, stack <= pc) if c	1 or 2	Call Carry
0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CNC a	(pc <= a, stack <= pc) if !c	1 or 2	Call Not Carry
0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CZ a	(pc <= a, stack <= pc) if z	1 or 2	Call Zero
1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CNZ a	(pc <= a, stack <= pc) if !z	1 or 2	Call Not Zero
1 0 1 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CN a	(pc <= a, stack <= pc) if n	1 or 2	Call Negative
1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 a a a a a a a a a a a a a a a a	CNN a	(pc <= a, stack <= pc) if !n	1 or 2	Call Not Negative

Type No Args

Format	Instruction	Operation	Cycles	Description
0 0 0 0 1 1 1 0 0 1 0 0 1 1 1 1	RET	pc <= stack	1 or 3	Return
0 0 1 0 1 1 1 0 0 1 0 0 1 1 1 1	RC	pc <= stack if c	1 or 3	Return Carry
0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1	RNC	pc <= stack if !c	1 or 3	Return Not Carry
0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1	RZ	pc <= stack if z	1 or 3	Return Zero
1 0 0 0 1 1 1 0 0 1 0 0 1 1 1 1	RNZ	pc <= stack if !z	1 or 3	Return Not Zero
1 0 1 0 1 1 1 0 0 1 0 0 1 1 1 1	RN	pc <= stack if n	1 or 3	Return Negative
1 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1	RNN	pc <= stack if !n	1 or 3	Return Not Negative
0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 1	PUS	stack <= s	1	Push Status Register
0 0 0 0 1 1 1 0 0 1 1 0 1 1 1 1	POS	s <= stack	2	Pop Status Register
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	NOP		1	No Operation
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	HLT		1	Halt

Type M

Format	Instruction	Operation	Cycles	Description
0 0 0 0 m m m m 0 1 1 1 1 1 1 1	SSR	s <= s | m	1	Set Status Register
0 0 0 0 m m m m 1 0 0 0 1 1 1 1	CSR	s <= s & m	1	Clear Status Register

Type P-P

Format	Instruction	Operation	Cycles	Description
pd pd pd 0 ps ps ps 0 1 0 0 1 1 1 1 1	MVP pd, ps	pd <= ps	1	Move Register Pair