Value Valu								
	Type R-I							
			-		-			
The content of the								
Compare								
		· ·	'					
Format								
Type R-10		· · · · · · · · · · · · · · · · · · ·						
Format	rd rd rd rd i i i i i i i 0 1 1 1	CPI rd, i		1	Compare Immediate			
The companies of the		Instruction		Cycles	Description			
Type R-R		<u> </u>						
	rs rs rs io io io io io io io io 1 0 0 1	OUT rs, io		1	IO Out			
		Instruction	Operation		Description			
				1				
		<u> </u>						
Instruction Compare								
Total rate Tot		ADD rd, rs		1				
To		ADC rd, rs	rd <= rd + rs + C	1	ADD With Carry			
Type R-P Format		CMP rd, rs	See Notes	1	Compare			
Type R-P Cycles Description Store, Post Increment Store, Post Increme		SUB rd, rs	rd <= rd - rs	1	Subtract			
Second S	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			
To To To To To To To To			Type R-P					
S TS TS TS TS TS TS TS		Instruction	Operation	Cycles	Description			
		SRI rs, p	*p <= rs, p <= p + 1	1	Store, Post Increment			
Type R-P-K	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			
Type R-P-K Format Fo		LRI rd, p	rs <= *p, p <= p + 1	2	Load, Post Increment			
Format	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			
S S S P P R K K K K K K K K K			Type R-P-K					
Type P-I Type P-I Type P-I Type Branch Type Branc				Cycles				
Type P-I			*(p + k) <= rs	1	Store, With Offset			
Instruction Operation Op	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			
1	Type P-I							
Type Branch Type Branch T		Instruction		Cycles	Description			
Instruction	i i i i p p p i i i i i 1 1 0 1	API p, i		1	Add Pair Immediate			
D Q Q Q Q Q Q Q Q Q			Type Branch					
0 0 1 a a a a a a a a a a a a a a a a a	Tormas		Operation					
0 1 0 a a a a a a a a a a a a a a a a a								
0 1 1 a a a a a a a a a a a a a a a a a								
1 0 0 a a a a a a a a a a a a a a a a a								
1								
1								
Type R Total Tota					-			
Instruction Operation Operation Cycles Description	1 1 0 a a a a a a a a 1 1 1 0	BNN a	pc <= pc + a, if !n	1	Branch No Negative			
rd r	Type R							
rd r		Instruction	Operation Operation	Cycles	Description			
rd rd rd rd rd rd rd rd 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 SRA rd rd <= rd >>> 1 Shift Right Arithmetic rd rd rd rd 0 0 0 0 1 1 1 0 1 1 1 1 1 RLC rd {c,rd} <= {rd,c} 1 Rotate Left Through Carry		SLL rd	rd <= rd << 1	1	Shift Left Logical			
rd rd rd rd rd rd 0 0 0 0 0 1 1 0 1 1 1 1 1 1 RLC rd {c,rd} <= {rd,c} 1 Rotate Left Through Carry		SRL rd	rd <= rd >> 1	1	Shift Right Logical			
		SRA rd	rd <= rd >>> 1	1	Shift Right Arithmetic			
		RLC rd	{c,rd} <= {rd,c}	1	Rotate Left Through Carry			
		RRC rd	{rd,c} <= {c,rd}	1	Rotate Right Through Carry			
rd rd rd rd rd 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 NOT rd rd <= !rd 1 NOT	rd rd rd rd 0 0 0 0 1 1 1 1 1 1 1 1 1	NOT rd	rd <= !rd	1	NOT			
rd rd rd rd rd 1 1 1 1 0 0 0 1 1 0 0 0 0 0 0 POP rd rd <= stack 2 POP	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 1 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 1	JNCI, p	PC <= p, if !c	1	Jump Not Carry Indirect			
0 1 1 0 p p p 0 0 0 1 1 1 1 1 1	JZI, p	$PC \le p$, if z	1	Jump Zero Indirect			
1 0 0 0 p p p 0 0 0 0 1 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 1 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 1 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 1 0 0 0 0 0 1 1 1 1 1 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 1 1 0 0 0 1 1 1 1 1 1 1 1	JNC a	pc <= a, if !c	1 or 2	Jump Not Carry			
0 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1	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 1 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 0 1 0 1 1 1 1 1 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 1 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	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	CC a	(pc <= a, stack <= pc) if c	1 or 2	Call Carry			
0 1 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1	CNC a	(pc <= a, stack <= pc) if !c	1 or 2	Call Not Carry			
0 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 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 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	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	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 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 0 0 1 1 1 1 0 0 1 0 1 1 1 1	RNC	pc <= stack if !c	1 or 3	Return Not Carry			
0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 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 1 1 1 1	RNZ	pc <= stack if !z	1 or 3	Return Not Zero			
1 0 1 0 1 1 1 1 2 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 1 1 1	RNN	pc <= stack if !n	1 or 3	Return Not Negative			
0 0 0 0 1 1 1 1 0 0 1 1 1 1 1		stack <= s	1	Push Status Register			
0 0 0 0 1 1 1 0 0 1 1 0 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 0 0	NOP		1	No Operation			
	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 0 1 1 1 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			