Inst.	Cycle	A1				A2				A3				M1				M2				X1				X2				X3				A1	
		a11	*a12*	*a12clk1*	*a12clk2*	*a21*	*a22*	*a22clk1*	*a22clk2*	*a31*	*a32*	*a32clk1*	*a32clk2*	*m11*	*m12*	*m12clk1*	*m12clk2*	*m21*	*m22*	*m22clk1*	*m22clk2*	*x11*	*x12*	*x12clk1*	*x12clk2*	*x21*	*x22*	*x22clk1*	*x22clk2*	*x31*	*x32*	*x32clk1*	*x32clk2*	*a11*	*a12*
ALL [1]		**data = pl**				**data = pm**				**data = ph cm_rom = 1 if io: cm_ram = ram_bank**			**pc++**	**cm_rom = 0 cm_ram = 0**	**ada = 0 adb = 0xF adc = 0**		**if sc: opr = data**	**cm_rom = 1 if io: cm_ram = ram_bank**			**if sc: opa = data**	**cm_rom = 0 cm_ram = 0**	**acc_out = acc cy_out = cy**			**if ope: alu.init() [2]**		**if not io: tmp = data**	**if io: tmp = data**						
NOP	**1/1**			sc = 1																															
JCN	**1/2**			sc = 1																															
	2/2			sc = 0 cond = jcn_cond(opa)													pm = data				pl = data														
FIM	**1/2**			sc = 1																															
	2/2			sc = 0													regp[opa][7:4] = data				regp[opa][3:0] = data														
SRC	**1/1**			sc = 1																						data = regp[opa][7:4] cm_rom = 1 cm_ram = ram_bank				data = regp[opa][3:0] cm_rom = 0 cm_ram = 0					
FIN	**1/2**			sc = 1																						data = regp[0][7:4]			pm = data	data = regp[0][3:0]			pl = data		
	2/2			sc = 0													regp[opa][7:4] = data				regp[opa][3:0] = data														
JIN	**1/1**			sc = 1																						data = regp[opa][7:4]			pm = data	data = regp[opa][3:0]			pl = data		
JUN	**1/2**			sc = 1																															
	2/2			sc = 0													pm = data				pl = data					data = opa			ph = data						
JMS	**1/2**			sc = 1																															
	2/2			sc = 0													pm = data				sp-- pl = data					data = opa			ph = data						
INC	**1/1**			sc = 1																						data = reg[opa]		adc = 1		data = add			reg[opa] = data		
ISZ	**1/2**			sc = 1																						data = reg[opa]		adc = 1		data = add			reg[opa] = data		
	2/2			sc = 0 cond = jcn_cond(opa)													pm = data				pl = data														
ADD	**1/1**			sc = 1																						data = reg[opa]		ada = acc adc = cy		acc = add cy = co					
SUB	**1/1**			sc = 1																						data = reg[opa]		ada = acc adb = ~adb adc = ~cy		acc = add cy = co					
LD	**1/1**			sc = 1																						data = reg[opa]				acc = add					
XCH	**1/1**			sc = 1																						data = reg[opa]				acc = add data = acc_out			reg[opa] = data		
BBL	**1/1**			sc = 1																	sp--				sp--	data = opa			sp--	acc = add					
LDM	**1/1**			sc = 1																						data = opa				acc = add					
WRM	**1/1**			sc = 1																						data = acc_out									
WMP	**1/1**			sc = 1																						data = acc_out									
WRR	**1/1**			sc = 1																						data = acc_out									
WR0	**1/1**			sc = 1																						data = acc_out									
WR1	**1/1**			sc = 1																						data = acc_out									
WR2	**1/1**			sc = 1																						data = acc_out									
WR3	**1/1**			sc = 1																						data = acc_out									
SBM	**1/1**			sc = 1																									ada = acc adb = ~adb adc = ~cy					acc = add cy = co
RDM	**1/1**			sc = 1																														acc = add	
RDR	**1/1**			sc = 1																														acc = add	
ADM	**1/1**			sc = 1																									ada = acc adc = cy					acc = add cy = co
RD0	**1/1**			sc = 1																														acc = add	
RD1	**1/1**			sc = 1																														acc = add	
RD2	**1/1**			sc = 1																														acc = add	
RD3	**1/1**			sc = 1																														acc = add	
CLB	**1/1**			sc = 1																															
CLC	**1/1**			sc = 1																															
IAC	**1/1**			sc = 1																															
CMC	**1/1**			sc = 1																															
CMA	**1/1**			sc = 1																															
RAL	**1/1**			sc = 1																															
RAR	**1/1**			sc = 1																															
TCC	**1/1**			sc = 1																															
DAC	**1/1**			sc = 1																															
TCS	**1/1**			sc = 1																															
STC	**1/1**			sc = 1																															
DAA	**1/1**			sc = 1																															
KBP	**1/1**			sc = 1																															
DCL	**1/1**			sc = 1																						ram_bank = dcl(acc_out)									

[1] For all instructions

[2] A complex initialization routine sets the bus with the proper value for tmp for all ope instructions.