(micro-)instruction overview

hex	*Instruction / stages:*	*0*	*1*	*2*	*3*	*4*
NO Operation						
x00	NOP (No OPeration)	PCO, MAI *point to PC value*	MO, IRI, CE *Load val into IR, count enable*			
LOAD instructions						
x01	LDAi (Load A immediate) *LDAi val*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to val*	MO, AI, CE *val to A*	
x02	LDAm (Load A from memory) *LDAm adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *point to val*	MO, AI, CE *val to A*
x07	LDBi (Load B immediate) *LDBi val*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to val*	MO, BI, CE *val to B*	
x08	LDBm (Load B from memory) *LDBm adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *point to val*	MO, BI, CE *val to B*
STORE instructions						
x10	STAm (STore A to memory) *STAm adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *load adr*	AO, MI, CE *store A*
x11	STBm (STore B to memory) *STBm adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *load adr*	BO, MI, CE *store B*
ARITHMETIC instructions						
x20	ADD (Add B to A) *ADD*	PCO, MAI	MO, IRI, CE	AL3, AL0, EO, AI, CE *add B to A*		
x21	ADDi (Add immediate) *ADDi val*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to val*	MO, BI *load val in B*	AL3, AL0, EO, AI, CE *add to A*
x22	ADDm (Add from memory) *ADDm adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *point to val*	MO, BI *load val in B*
x24	*SUB (Subtract B from A)* *SUB*	PCO, MAI	MO, IRI, CE	*AL2, AL1, EO, AI, CE* *sub B from A*		
x25	*SUBi (Subtract immediate)* *SUBi val*	PCO, MAI	MO, IRI, CE	*PCO, MAI* *point to val*	*MO, BI* *load val in B*	*AL2, AL1, EO, AI, CE* *sub from A*
x26	*SUBm (Subtract from memory)* *SUBm adr*	PCO, MAI	MO, IRI, CE	*PCO, MAI* *point to adr*	*MO, MAI* *point to val*	*MO, BI* *load val in B*
x28	*INCA (Add 1 to register A)* *INCA*	PCO, MAI	MO, IRI, CE	*AO, BI* *A to B*	*ALM, AL3, AL2, EO, AI* *Make A = 1*	AL3, AL0, EO, AI *add B to A*

x29	INCB (Add 1 to register B)	PCO, MAI	MO, IRI, CE	ALM, AL3, AL2, EO, AI	AL3, AL0, EO, AI	AO, BI
	INCB			Make A = 1	add B to A	A to B (so B = 1)

LOGIC instructions

x30	XOR	PCO, MAI	MO, IRI, CE	ALM, AL2, AL1, EO, AI, CE		
	A = A XOR B			*xor B into A*		
x31	XORi (Xor immediate)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, BI	ALM, AL2, AL1, EO, AI, CE
	A = A XOR imm.value			*point to val*	*load val in B*	*xor B into A*
x32	XORm	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, MAI	MO, BI
	XORm adr			*point to adr*	*point to val*	*load val in B*
x34	AND	PCO, MAI	MO, IRI, CE	ALM, AL3, AL1, AL0, EO, AI, CE		
	A = A AND B			*and B into A*		
x35	ANDi (AND immediate)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, BI	ALM, AL3, AL1, AL0, EO, AI, CE
	A = A AND imm.value			*point to val*	*load val in B*	*and B into A*
x36	ANDm	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, MAI	MO, BI
	ANDm adr			*point to adr*	*point to val*	*load val in B*
x38	OR	PCO, MAI	MO, IRI, CE	ALM, AL3, AL2, AL1, EO, AI, CE		
	A = A OR B			*or B into A*		
x39	ORi (OR immediate)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, BI	ALM, AL3, AL2, AL1, EO, AI, CE
	A = A OR imm.value			*point to val*	*load val in B*	*or B into A*
x3A	ORm	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, MAI	MO, BI
	ORm adr			*point to adr*	*point to val*	*load val in B*
x3C	NOTA	PCO, MAI	MO, IRI, CE	ALM, EO, AI		
	A = NOT A			*NOT A to A*		
x3D	NOTB	PCO, MAI	MO, IRI, CE	ALM, AL2, AL0, EO, BI		
	B = NOT B			*NOT B to B*		
x3E	CLRA	PCO, MAI	MO, IRI, CE	ALM, AL1, AL0, EO, AI		
	A = 0			*A = 0*		
x3F	CLRB	PCO, MAI	MO, IRI, CE	ALM, AL1, AL0, EO, BI		
	B = 0			*B = 0*		

JUMP instructions

x40	JMPi (Jump immediate)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, PCI	
	JMPi adr			*point to adr*	*adr to PC*	
x41	JMPm (Jump indirect)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, MAI	MO, PCI
	JMPm adr			*point to adr*	*load adr*	*adr to PC*
x44	JPCi (Jump when carry immediate)	PCO, MAI	MO, IRI, CE	PCO, MAI	CE	MO, PCIc
	JPCi adr			point to adr	bump PC	adr to PC when carry
x45	JPCm (Jump when carry indirect)	PCO, MAI	MO, IRI, CE	PCO, MAI	MO, MAI	CE
	JPCm adr			point to adr	load adr	bump PC

OUTPUT instructions

x50	OUTA (Output A) *OUTA*	PCO, MAI	MO, IRI, CE	AO, OI *A to output*		
x51	OUTB (Output B) *OUTB*	PCO, MAI	MO, IRI, CE	BO, OI *B to output*		
x54	OUTi (Output immediate) *OUTi val*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to val*	MO, OI, CE *val to output*	
x55	OUTm (Output from memory) *OUTi adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr*	MO, MAI *load adr*	MO, OI, CE *val to output*

MOVE instructions

x60	MOVi (Move immediate) *MOVi val, adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to val*	MO, BI, CE *val in B*	PCO, MAI *point to adr*
x61	MOVm (Move mem to mem) *MOVm adr, adr*	PCO, MAI	MO, IRI, CE	PCO, MAI *point to adr1*	MO, MAI *point to val*	MO, BI, CE *val in B*

HALT

xFF	HLT (HaLT program)	PCO, MAI	MO, IRI, CE	HLT		

5 ***6*** ***7***

AL3, AL0, EO, AI, CE
add to A

AL2, AL1, EO, AI, CE
sub from A

Destructive to A register contents!

ALM, AL2, AL1, EO, AI, CE
xor B into A

ALM, AL3, AL1, AL0, EO, AI, CE
and B into A

ALM, AL3, AL2, AL1, EO, AI, CE
or B into A

MO, PCIc
adr to PC when carry

BO, MI, CE
val to mem

PCO, MAI
point to adr2

BO, MI, CE
val to mem