Opcode	Mnemonic	Macro operation		Micro operations	NB	Step	Control actions
	(fetch)	[IR] ← [MS(PC)]	1	[MAR] ← [PC]		1	E_{PC}, C_{MAR}
			2	[IR(0:7)] ← [MS(MAR)]		2	R, E_{MS}, C_{IR}
		[PC] ← [PC] + 1	3	[ALU(Q)] ← [PC]		3	E_{PC}
			4	[ALU(F)] ← 01 ("Q+1")		3	F_1=0 & F_2=1
			5	[ALUreg] ← [ALU]		3	C_{ALUR}
			6	[PC] ← [ALUreg]		4	E_{ALUR}, C_{PC}
000	CLEAR	[D0] ← 0	1-4	Same as INC1, with F=00 (“Zero”) - but NOTE step 1 is not applicable & shouldn't be included...	• Opcodes of CLEAR, INC1, ADD# and DEC1 match ALU(F) for neatness. • Opcodes 000-011 are arithmetic instructions.		
001	INC1	[D0] ← [D0] + 1	1	[ALU(Q)] ← [D0]	Can’t use P input as need ALU to do Q+1 to increment PC in fetch	1	E_{D0}
			2	[ALU(F)] ← 01 ("Q+1")		1	F_1=0 & F_2=1
			3	[ALUreg] ← [ALU]		1	C_{ALUR}
			4	[D0] ← [ALUreg]		2	E_{ALUR}, C_{D0}
010	ADD# v	[D0] ← [D0] + v	1	[ALU(P)] ← [D0]	Need ALU(P) = D0 as well as ALU(Q) = main bus for ADD#	1	No control actions required
			2	[ALU(Q)] ← [IR(0:4)]		1	E_{IR}
			3	[ALU(F)] ← 10 ("Q+P")		1	F_1=1 & F_2=0
			4	[ALUreg] ← [ALU]	This is why ALUreg is necessary: otherwise ALU would simultaneously feed and be fed by main bus.	1	C_{ALUR}
			5	[D0] ← [ALUreg]		2	E_{ALUR}, C_{D0}
011	DEC1	[D0] ← [D0] - 1	1-4	Same as INC1, with F=11 (“Q-1”)			
100	JMP loc	[PC] ← loc	1	[PC] ← [IR(0:4)]		1	E_{IR}, C_{PC}
101	BUZ loc / BNZ loc / BZC loc / BNE loc	If Z is not 0 then [PC] ← loc	1	If Z is not 0 then [PC] ← [IR(0:4)]		1	If Z is not 0 then E_{IR}, C_{PC}
110	LOAD loc	[D0] ← [MS(loc)]	1	[MAR] ← [IR(0:4)]		1	E_{IR}, C_{MAR}
			2	[D0] ← [MS(MAR)]		2	R, E_{MS}, C_{D0}
111	STORE loc	[MS(loc])] ← [D0]	1	[MAR] ← [IR(0:4)]		1	E_{IR}, C_{MAR}
			2	[MS(MAR)] ← [D0]		2	E_{D0}, W

F_1	F_2	ALU output	NB
0	0	Zero	00=”0”
0	1	Q+1	01=”+1”
1	0	Q+P	
1	1	Q-1	11=”-1”

• NB macro operations don’t mention register widths, MAR, or operand encoding. Micro operations add this detail.

Control action timing assumptions:

can enable an output somewhere and clock it in somewhere else in one step.

enable inputs are level triggered

clock inputs are edge triggered: they are activated on a negative-going transition (from high to low).

ALU can be given inputs (via enabling some outputs), calculate, and CALUR can be clocked in same step. However, EALUR must be enabled in the next step.

Therefore clock actions are always the last within a step.

- Main store is quick enough to be given an address in one step, then in next step, R, EMS and output can be used.

E_{MS}
E_{IR}
E_{PC}
E_{D0}
E_{ALUR}

C_{MAR}
C_{IR}
C_{PC}
C_{D0}
C_{ALUR}

F_1
F_2

R
W

Z
C_{CU}