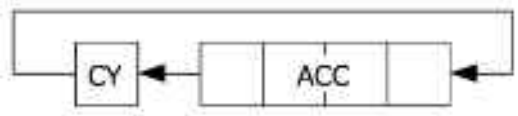
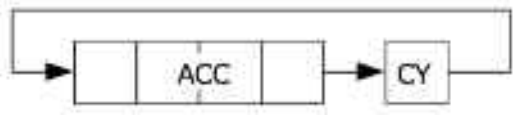
Type	Mne-monic	Assembly Example	Machine Code								Operation Summary	Detailed Description
			OPR (Upper)				OPA (Lower)					
Clear Both	**CLB**	clb	1	1	1	1	0	0	0	0	●PC+1→PCTEMP ●0→ACC ●0→CY ●PCTEMP→PC	Clears both the ACC (accumulator) and CY (carry) to zero.
Clear Carry	**CLC**	clc	1	1	1	1	0	0	0	1	●PC+1→PCTEMP ●0→CY ●PCTEMP→PC	Clears CY to zero.
Complement Carry	**CMC**	cmc	1	1	1	1	0	0	1	1	●PC+1→PCTEMP ●$\overline{CY}$→CY ●PCTEMP→PC	Inverts the value of CY.
Set Carry	**STC**	stc	1	1	1	1	1	0	1	0	●PC+1→PCTEMP ●1→CY ●PCTEMP→PC	Sets CY to 1.
Complement Accumulator	**CMA**	cma	1	1	1	1	0	1	0	0	●PC+1→PCTEMP ●$\overline{ACC}$→ACC ●PCTEMP→PC	Inverts ACC. CY remains unchanged.
Increment Accumulator	**IAC**	iac	1	1	1	1	0	0	1	0	●PC+1→PCTEMP ●ACC+1→ACC,CY ●PCTEMP→PC	Increments ACC. If there is no overflow, CY becomes 0; if an overflow occurs, CY becomes 1.
Decrement Accumulator	**DAC**	dac	1	1	1	1	1	0	0	0	●PC+1→PCTEMP ●ACC-1→ACC,CY ●PCTEMP→PC	Decrements ACC. If there is no borrow, CY becomes 1; if a borrow occurs, CY becomes 0. a3 a2 a1 a0 +) 1 1 1 1 c4 s3 s2 s1 s0 c4→CY, {s3,s2,s1,s0}→ACC
Rotate Left	**RAL**	ral	1	1	1	1	0	1	0	1	●PC+1→PCTEMP ●{ACC[3:0],CY} →{CY, ACC[3:0]} ●PCTEMP→PC	Performs a left rotate of ACC including CY. 
Rotate Right	**RAR**	rar	1	1	1	1	0	1	1	0	●PC+1→PCTEMP ●{CY,ACC[3:0]} →{ACC[3:0],CY} ●PCTEMP→PC	Performs a right rotate of ACC including CY. 
Transmit Carry and Clear	**TCC**	tcc	1	1	1	1	0	1	1	1	●PC+1→PCTEMP ●0→ACC ●CY→ACC[0] ●0→CY ●PCTEMP→PC	Clears ACC, then transfers the CY value to ACC's least significant bit (LSB), and finally clears CY to 0.
Decimal Adjust Accumulator	**DAA**	daa	1	1	1	1	1	0	1	1	●PC+1→PCTEMP ●If (CY|(ACC>9)), ACC+6→ACC ●If Carry, 1→CY ●PCTEMP→PC	Decimal adjust instruction: If CY is 1 or ACC is greater than 9, add 6 to ACC. If this addition generates a carry, set CY to 1; if there's no carry, CY remains unchanged.
Transfer Carry Subtract	**TCS**	tcs	1	1	1	1	1	0	0	1	●PC+1→PC ●If (CY==0),9→ACC ●If (CY==1),10→ACC ●0→CY	If CY = 0, store 9 into ACC; if CY = 1, store 10 into ACC. Then clear CY to 0.
Keyboard Process	**KBP**	kbp	1	1	1	1	1	1	0	0	●PC+1→PCTEMP ●KBP(ACC)→ACC ●PCTEMP→PC	Keyboard scan code conversion instruction: If the value in ACC has exactly one bit set to 1, it is converted to a numerical value. If multiple bits are set to 1, the result is set to 15 (indicating an error). CY remains unchanged. ACC (Before) → ACC (After) 0 0 0 0 → 0 0 0 0 0 0 0 1 → 0 0 0 1 0 0 1 0 → 0 0 1 0 0 1 0 0 → 0 0 1 1 1 0 0 0 → 0 1 0 0 0 0 1 1 → 1 1 1 1 0 1 0 1 → 1 1 1 1 0 1 1 0 → 1 1 1 1 0 1 1 1 → 1 1 1 1 1 0 0 1 → 1 1 1 1 1 0 1 0 → 1 1 1 1 1 0 1 1 → 1 1 1 1 1 1 0 0 → 1 1 1 1 1 1 0 1 → 1 1 1 1 1 1 1 0 → 1 1 1 1 1 1 1 1 → 1 1 1 1
Designate Command Line	**DCL**	dcl	1	1	1	1	1	1	0	1	●PC+1→PCTEMP ●ACC[2:0]→DCL ●PCTEMP→PC	Transfers the lower 3 bits of ACC to the DCL register inside the CPU, thereby specifying how $\overline{CM}$-RAMx (RAM bank selection) signals are output from that point forward.