Type	Mne-monic	Assembly Example	Machine Code OPR (Upper)				Machine Code OPA (Lower)				Operation Summary	Detailed Description
No Operation	**NOP**	`nop`	0	0	0	0	0	0	0	0	●PC+1→PCTEMP ●PCTEMP→PC	No operation performed.
Load Data to Accumulator	**LDM**	`ldm 0xa` `ldm 10` `ldm LABEL`	1	1	0	1	D				●PC+1→PCTEMP ●D→ACC ●PCTEMP→PC	Loads 4 bit immediate value (OPA field) into ACC. CY remains unchanged.
Load Index Register to Accumulator	**LD**	`ld 3` `ld 15`	1	0	1	0	n				●PC+1→PCTEMP ●Rn→ACC ●PCTEMP→PC	Loads 4 bit content of index register Rn into ACC. CY unchanged.
Exchange Index Register and Accumulator	**XCH**	`xch 3` `xch 15`	1	0	1	1	n				●PC+1→PCTEMP ●ACC→ACBR ●Rn→ACC ●ACBR→Rn ●PCTEMP→PC	Exchanges contents of ACC and Rn. CY unchanged.
Add Index Register to Accumulator with Carry	**ADD**	`add 3` `add 15`	1	0	0	0	n				●PC+1→PCTEMP ●ACC+Rn+CY→ACC,CY ●PCTEMP→PC	Performs 4-bit addition of ACC, Rn, and CY. Result to ACC; carry-out to CY. `a3 a2 a1 a0` `cy` `+) r3 r2 r1 r0` `c4 s3 s2 s1 s0` `c4→CY, {s3,s2,s1,s0}→ACC`
Subtract Index Register from Accumulator with Borrow	**SUB**	`sub 3` `sub 15`	1	0	0	1	n				●PC+1→PCTEMP ●ACC+$\overline{Rn}$+$\overline{CY}$→ACC,CY ●PCTEMP→PC	The contents of ACC, the bitwise inversion of the 4-bit value in index register Rn (as specified by number n), and the inverted CY bit are added together. The result is stored in ACC, and the carry-out is stored in CY. Note: If the CY value prior to calculation is 0, subtraction is performed without borrow. If the CY value prior to calculation is 1, subtraction is performed with borrow. After calculation, CY = 0 indicates a borrow occurred, and CY = 1 indicates no borrow occurred. Keep in mind that the interpretation of the CY bit is inverted after execution with respect to the borrow status. `a3 a2 a1` $\overline{a0}$ $\overline{cy}$ `+)` $\overline{r3}$ $\overline{r2}$ $\overline{r1}$ $\overline{r0}$ `c4 s3 s2 s1 s0` `c4→CY, {s3,s2,s1,s0}→ACC`
Increment Index Register	**INC**	`inc 8` `inc 12`	0	1	1	0	n				●PC+1→PCTEMP ●Rn+1→Rn ●PCTEMP→PC	The 4 bit content of index register Rn, designated by number n, is incremented by 1. If the original value is 15, it wraps around to 0. The CY (carry) bit remains unchanged.
Branch Back and Load Data to Accumulator	**BBL**	`bbl 0xa` `bbl 12`	1	1	0	0	D				●PC+1→PCTEMP ●SP-1→SP ●(Stack)→PC ●D→ACC	This instruction performs a return from a subroutine. The stack pointer (SP) is decremented by one, and the return address is retrieved from the stack and loaded into the program counter (PC). The PC now points to the instruction following the subroutine call (JMS). Additionally, the 4-bit value D from the instruction's OPA field is transferred into the accumulator (ACC).
Jump indirect	**JIN**	`jin 3p` `jin 3<` `These are identical.`	0	0	1	1	m	1			●PC+1→PCTEMP ●PCTEMP[11:8]→PC[11:8] ●RmP[7:4]→PC[7:4] ●RmP[3:0]→PC[3:0]	After the PC is incremented (+1) to point to the next instruction, the lower 8 bits of the PC are overwritten with the 8-bit contents of index register pair RmP, designated by number m. Note: If PC[7:0] was 0xFF before executing the JIN instruction, incrementing the PC causes a carry into PC[11:8] (page number), which is then incremented. The lower 8 bits are overwritten afterward. If the JIN instruction resides at the last address of a page (PC[7:0] == 0xFF), the jump target ends up inside the next page.
Send Register Control	**SRC**	`src 5p` `src 5<` `src 10` `These are identical.`	0	0	1	0	m	1			●PC+1→PCTEMP ●RmP[7:4]→DB @X2 ●RmP[3:0]→DB @X3 ●PCTEMP→PC	The 8-bit contents of index register pair RmP, specified by number m, are output to the data bus: the upper 4 bits during state X2, and the lower 4 bits during state X3. These values serve as address information for chip selection when accessing ROM (I/O ports), or RAM (output ports, character data, status characters). ROM and RAM receive and retain these values for upcoming access instructions.

Fetch Index Register from ROM	**FIN**	`fin 7p`
		`fin 7<`
		`fin 14`
		
		`These are identical.`
			0	0	1	1	m	0	命令サイクル1で
									●PC+1→PCTEMP
									
									命令サイクル2で
									●R0P[3:0]→DB @A1
									●R0P[7:4]→DB @A2
									●PCTEMP[11:8]→DB @A3
									●DB→RmP[7:4] @M1(OPR)
									●DB→RmP[3:0] @M2(OPA)
									●PCTEMP→PC
										Although this is a one-word instruction, it requires two instruction cycles to execute. In instruction cycle 1, the PC is incremented (+1) to point to the next instruction.
										
										In instruction cycle 2, the ROM address is built as follows:
										- During state A1: the lower 4 bits of index register pair R0P (i.e., R1) are output to the data bus
										- During state A2: the upper 4 bits of R0P (i.e., R0) are output to the data bus
										-During state A3: the upper bits of PCTEMP (PC[11:8], the page number) are output to the data bus
										
										ROM data is read via states M1 and M2:
										- M1 retrieves the upper 4 bits (OPR) and loads them into the upper part of RmP
										- M2 retrieves the lower 4 bits (OPA) and loads them into the lower part of RmP
										- Finally, PCTEMP is transferred to PC
										
										Note: If PC[7:0] was 0xFF before executing the FIN instruction, the PC increment causes a carry into PC[11:8]. The read address for ROM is then constructed using this new page value and the R0P pair. If the FIN instruction is placed at the final address of a page (PC[7:0] == 0xFF), the ROM access address will be inside the next page.