

**Table 3-2. Summary of Indexed Operations**

Postbyte Code (xb)	Source Code Syntax	Comments rr; 00 = X, 01 = Y, 10 = SP, 11 = PC
rr0nnnnn	,r n,r -n,r	<b>5-bit constant offset</b> $n = -16$ to $+15$ r can specify X, Y, SP, or PC
111rr0zs	n,r -n,r	<b>Constant offset</b> (9- or 16-bit signed) z- 0 = 9-bit with sign in LSB of postbyte(s) $-256 \leq n \leq 255$ 1 = 16-bit $-32,768 \leq n \leq 65,535$ if $z = s = 1$ , 16-bit offset indexed-indirect (see below) r can specify X, Y, SP, or PC
111rr011	[n,r]	<b>16-bit offset indexed-indirect</b> rr can specify X, Y, SP, or PC $-32,768 \leq n \leq 65,535$
rr1pnnnn	n,-r n,+r n,r- n,r+	<b>Auto predecrement, preincrement, postdecrement, or postincrement;</b> p = pre-(0) or post-(1), $n = -8$ to $-1$ , $+1$ to $+8$ r can specify X, Y, or SP (PC not a valid choice) +8 = 0111 ... +1 = 0000 -1 = 1111 ... -8 = 1000
111rr1aa	A,r B,r D,r	<b>Accumulator offset</b> (unsigned 8-bit or 16-bit) aa-00 = A 01 = B 10 = D (16-bit) 11 = see accumulator D offset indexed-indirect r can specify X, Y, SP, or PC
111rr111	[D,r]	<b>Accumulator D offset indexed-indirect</b> r can specify X, Y, SP, or PC

Indexed addressing mode instructions use a postbyte to specify index registers (X and Y), stack pointer (SP), or program counter (PC) as the base index register and to further classify the way the effective address is formed. A special group of instructions cause this calculated effective address to be loaded into an index register for further calculations:

- Load stack pointer with effective address (LEAS)
- Load X with effective address (LEAX)
- Load Y with effective address (LEAY)