

Table 4. W65SC816 Microprocessor Op Code Matrix

	LSD																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
MSD	0	BRK s 2 8	ORA(d,x) 2 6	CDP s 2*8	ORA sr 2*4	TSB d 2*5	ORA d 2 3	ASL d 2 5	ORA(dl) 2*6	PHP s 1 3	ORA imm 2 2	ASL acc 1 2	PHD s 1*4	TSB a 3*6	ORA a 3 4	ASL a 3 6	ORA al 4*5	0
	1	BPL r 2 2	ORA(d,y) 2 5	ORA (d) 2*5	ORA(sr,y) 2*7	TRB d 2*5	ORA d,x 2 4	ASL d,x 2 6	ORA(dl,y) 2*6	CLC imp 1 2	ORA a,y 3 4	INC acc 1*2	TCS imp 1*2	TRB a 3*6	ORA a,x 3 4	ASL a,x 3 7	ORA al,x 4*5	1
	2	JSR a 3 6	AND(d,x) 2 6	JSL al 4*8	AND sr 2*4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND(dl) 2*6	PLP s 1 4	AND imm 2 2	ROL acc 1 2	PLD s 1*5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4*5	2
	3	BMI r 2 2	AND(d,y) 2 5	AND(d) 2*5	AND(sr,y) 2*7	BIT d,x 2*4	AND d,x 2 4	ROL d,x 2 6	AND(dl,y) 2*6	SEC imp 1 2	AND a,y 3 4	DEC acc 1*2	TSC imp 1*2	BIT a,x 3*4	AND a,x 3 4	ROL a,x 3 7	AND al,x 4*5	3
	4	RTI s 1 7	EOR(d,x) 2 6	WDM RESERVED	EOR sr 2*4	MVP xyc 3*7	EOR d 2 3	LSR d 2 5	EOR(dl) 2*6	PHA s 1 3	EOR imm 2 2	LSR acc 1 2	PHK s 1*3	JMP a 3 3	EOR a 3 4	LSR a 3 6	EOR al 4*5	4
	5	BVC r 2 2	EOR(d,y) 2 5	EOR (d) 2*5	EOR(sr,y) 2*7	MVN xyc 3*7	EOR d,x 2 4	LSR d,x 2 6	EOR(dl,y) 2*6	CLI imp 1 2	EOR a,y 3 4	PHY s 1*3	TCD imp 1*2	JMP al 3*4	EOR a,x 3 4	LSR a,x 3 7	EOR al,x 4*5	5
	6	RTS s 1 6	ADC(d,x) 2 6	PER s 3*6	AOC sr 2*4	STZ d 2*3	AOC d 2 3	ROR d 2 5	AOC(dl) 2*6	PLA s 1 4	ADC imm 2 2	ROR acc 1 2	RTL s 1*6	JMP (a) 3 4	AOC a 3 4	ROR a 3 6	ADC al 4*5	6
	7	BVS r 2 2	ADC(d,y) 2 5	ADC(d) 2*5	ADC(sr,y) 2*7	STZ d,x 2*4	AOC d,x 2 4	ROR d,x 2 6	ADC(dl,y) 2*6	SEI imp 1 2	ADC a,y 3 4	PLY s 1*4	TOC imp 1*2	JMP(a,x) 3*6	ADC a,x 3 4	ROR a,x 3 7	ADC al,x 4*5	7
	8	BRA r 2*2	STA(d,x) 2 6	BRL rl 3*3	STA sr 2*4	STY d 2 3	STA d 2 3	STX d 2 3	STA(dl) 2*6	DEY imp 1 2	BIT imm 2*2	TXA imm 1 2	PHB s 1*3	STY a 3 4	STA a 3 4	STX a 3 6	STA al 4*5	8
	9	BCC r 2 2	STA(d,y) 2 6	STA (d) 2*5	STA(sr,y) 2*7	STY d,x 2 4	STA d,x 2 4	STX d,y 2 4	STA(dl,y) 2*6	TYA imp 1 2	STA a,y 3 5	TXS imp 1 2	TXY imp 1*2	STZ a 3*4	STA a,x 3 5	STZ a,x 3*5	STA al,x 4*5	9
	A	LDY imm 2 2	LDA(d,x) 2 6	LOX imm 2 2	LOA sr 2*4	LDY d 2 3	LOA d 2 3	LOX d 2 3	LDA(dl) 2*6	TAY imp 1 2	LOA imm 2 2	TAX imp 1 2	PLB s 1*4	LDY a 3 4	LOA a 3 4	LOX a 3 4	LDA al 4*5	A
	B	BCS r 2 2	LDA(d,y) 2 5	LDA(d) 2*5	LDA(sr,y) 2*7	LDY d,x 2 4	LOA d,x 2 4	LOX d,y 2 4	LDA(dl,y) 2*6	CLV imp 1 2	LOA a,y 3 4	TSX imp 1 2	TYX imp 1*2	LDY a,x 3 4	LOA a,x 3 4	LOX a,y 3 4	LDA al,x 4*5	B
	C	CPY imm 2 2	CMP(d,x) 2 6	REP imm 2*3	CMP sr 2*4	CPY d 2 3	CMP d 2 3	DEC d 2 5	CMP(dl) 2*6	INY imp 1 2	CMP imm 2 2	DEX imm 1 2	WAI imp 1*3	CPY a 3 4	CMP a 3 4	DEC a 3 6	CMP al 4*5	C
	D	BNE r 2 2	CMP(d,y) 2 5	CMP (d) 2*5	CMP(sr,y) 2*7	PEI s 2*6	CMP d,x 2 4	DEC d,x 2 6	CMP(dl,y) 2*6	CLO imp 1 2	CMP a,y 3 4	PHX s 1*3	STP imp 1*3	JML (a) 3*6	CMP a,x 3 4	DEC a,x 3 7	CMP al,x 4*5	D
	E	CPX imm 2 2	SBC(d,x) 2 6	SEP imm 2*3	SBC sr 2*4	LPX d 2 3	SBC d 2 3	INC d 2 5	SBC(dl) 2 6	INX imp 1 2	SBC imm 2 2	NOP imp 1 2	XBA imp 1*3	CPX a 3 4	SBC a 3 4	INC a 3 6	SBC al 4*5	E
	F	BEQ r 2 2	SBC(d,y) 2 5	SBC (d) 2*5	SBC(sr,y) 2*7	PEA s 3*5	SBC d,x 2 4	INC d,x 2 6	SBC(dl,y) 2*6	SED imp 1 2	SBC a,y 3 4	PLX s 1*4	XCE imp 1*2	JSR(a,x) 3*6	SBC a,x 3 4	INC a,x 3 7	SBC al,x 4*5	F

* New W65SC816 Op Codes

● W65SC02 Op Codes

Figure 3: The 65816's instruction set with hexadecimal equivalents, mnemonics, addressing-mode number of bytes for the op code and operand, and the basic number of clock cycles required for the instruction.

set. Not only is BRA position independent, it requires 1 less byte than the alternative JMP (unconditional jump) instruction. It is limited to destinations within 126 bytes before and 129 bytes after the branch instruction, but this range is frequently wide enough for loops within a program.

Because the 6502 has a limited number of registers, the index registers are frequently turned to general-purpose uses. Since there are no instructions to directly transfer data between index registers and the stack or between each other, the accumulator sometimes is used to mediate transfers. The se-

quence PHA (push the A register on the stack), TXA (transfer the contents of the X register to A), PHA, TYA (transfer the contents of Y to A), PHA is common in subroutines that must preserve all processor registers for the calling routine. The sequence for restoring the registers is equally clumsy. Also, the data in the A register is destroyed, and it is tricky for the subroutine to recover this data from the stack without destroying data for the index registers. The 65816 adds the instructions PHX, PHY (push index registers to stack), PLX, PLY (pull index registers from stack), TXY, and TYX (transfer X to Y and Y to X, respective-

ly). This sequence lets a subroutine save just those registers it needs to use and restore, or save them in a different sequence on the stack so that the data is available later in the correct sequence.

The 6502 has found applications both in general-purpose systems and in special applications, such as printer controllers. Most general-purpose applications don't often need the ability to set or reset specific bits of memory or input/output ports. Such dedicated applications as printer or appliance controllers, on the other hand, frequently rely heavily on this ability.

Multiple-processor systems need the