6500

NUMBER OF BYTES		2		2		2		3		3		3		2		2		2		1		3		CONDITION CODES					
INSTRUCTIONS	AM	IMM		Z.PAGE		Z.PAGE,X		ABS		ABS, X		ABS, Y		(IND,X)		(IND),Y		Z.PAGE,Y		ACCUM.		INDIRECT							
MNEMONIC	OPERATION	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	OP	Cy	N	Z	C	I	D	V
LDA	A ⟵ M	A9	2	A5	3	B5	4	AD	4	BD	4	B9	4	A1	6	B1	5							✓	✓	–	–	–	–
STA	A ⟶ M		2	85	3	95	4	8D	4	9D	5	99	5	81	6	91	6							–	–	–	–	–	–
LDX	X ⟵ M	A2	2	A6	3			AE	4			BE	4					B6	4					✓	✓	–	–	–	–
STX	X ⟶ M		2	86	3			8E	4									96	4					–	–	–	–	–	–
LDY	Y ⟵ M	A0	2	A4	3	B4	4	AC	4	BC	4													✓	✓	–	–	–	–
STY	Y ⟶ M		2	84	3	94	4	8C	4															–	–	–	–	–	–
AND	A ⟵ A∧M	29	2	25	3	35	4	2D	4	3D	4	39	4	21	6	31	5							✓	✓	–	–	–	–
ORA	A ⟵ A∨M	09	2	05	3	15	4	0D	4	1D	4	19	4	01	6	11	5							✓	✓	–	–	–	–
EOR	A ⟵ A⊻M	49	2	45	3	55	4	4D	4	5D	4	59	4	41	6	51	5							✓	✓	–	–	–	–
BIT	A ∧ M		2	24	3			2C	4															M_7	✓	–	–	–	M_6
CMP	A [minus] M	C9	2	C5	3	D5	4	CD	4	DD	4	D9	4	C1	6	D1	5							✓	✓	✓	–	–	–
CPX	X —— M	E0	2	E4	3			EC	4															✓	✓	✓	–	–	–
CPY	Y —— M	C0	2	C4	3			CC	4															✓	✓	✓	–	–	–
ADC	A,C⟵A+M+C	69	2	65	3	75	4	6D	4	7D	4	79	4	61	6	71	5							✓	✓	✓	–	–	–
SBC	A,C⟵A−M−$\bar{C}$	E9	2	E5	3	**F5**	4	ED	4	FD	4	F9	4	E1	6	F1	5							✓	✓	✓	–	–	–
ASL	C←7 0←0			06	5	**16**	6	0E	6	1E	7									0A	2			✓	✓	✓	–	–	–
LSR	0→7 0→C			46	5	**56**	6	4E	6	5E	7									4A	2			0	✓	✓	–	–	–
ROL	7 0←C←┐			26	5	**36**	6	2E	6	3E	7									2A	2			✓	✓	✓	–	–	–
ROR	┌→C→7 0			66	5	**76**	6	6E	6	7E	7									6A	2			✓	✓	✓	–	–	–
INC	M+1 ⟶ M			E6	5	**F6**	6	EE	6	FE	7													✓	✓	–	–	–	–
DEC	M−1 ⟶ M			C6	5	**D6**	6	CE	6	DE	7													✓	✓	–	–	–	–
JMP	JUMP TO NEW LOC.							4C	3													6C	5	–	–	–	–	–	–
JSR	JUMP SUBROUTINE (see Note 1)							20	6															–	–	–	–	–	–

		1		2		CONDITION CODES						
INSTRUCTIONS	AM	IMPLIED		REL								
MNEMONIC	OPERATION	OP	Cy	OP	Cy	N	Z	C	I	D	V	
BRK	BREAK	00	7			–	–	–	1	–	–	
RTS	RETURN SUB.	60	6			–	–	–	–	–	–	
RTI	RETURN INT.	40	6			RESTORED						
PHP	PUSH PROC.	08	3			SAVED						
PLP	PULL PROC.	28	4			RESTORED						
PHA	PUSH ACCUM.	48	3			–	–	–	–	–	–	
PLA	PULL ACCUM.	68	4			✓	✓	–	–	–	–	
INX	INC XREG	E8	2			✓	✓	–	–	–	–	
DEX	DEC XREG	CA	2			✓	✓	–	–	–	–	
INY	INC YREG	C8	2			✓	✓	–	–	–	–	
DEY	DEC YREG	88	2			✓	✓	–	–	–	–	
TAX	ACC. TO XREG	AA	2			✓	✓	–	–	–	–	
TXA	XREG TO ACC.	8A	2			✓	✓	–	–	–	–	
TAY	ACC. TO YREG	A8	2			✓	✓	–	–	–	–	
TYA	YREG TO ACC.	98	2			✓	✓	–	–	–	–	
TSX	STACK TO XREG	BA	2			✓	✓	–	–	–	–	
TXS	XREG TO STACK	9A	2			–	–	–	–	–	–	
SED	SET DECIMAL	F8	2			–	–	–	–	1	–	
CLD	CLEAR DECIMAL	D8	2			–	–	–	–	0	–	
SEI	SET INT. INHIBIT	78	2			–	–	–	1	–	–	
CLI	CLR INT. INHIBIT	58	2			–	–	–	0	–	–	
SEC	SET CARRY	38	2			–	–	1	–	–	–	
CLC	CLEAR CARRY	18	2			–	–	0	–	–	–	
CLV	CLR OVERFLOW	B8	2			–	–	–	–	–	0	
BEQ	BRANCH ON Z = 0			F0	2	–	–	–	–	–	–	
BNE	BRANCH ON Z = 1			D0	2	–	–	–	–	–	–	
BMI	BRANCH ON N = 1			30	2	–	–	–	–	–	–	
BPL	BRANCH ON N = 0			10	2	–	–	–	–	–	–	
BCC	BRANCH ON C = 0			90	2	–	–	–	–	–	–	
BCS	BRANCH ON C = 1			B0	2	–	–	–	–	–	–	
BVC	BRANCH ON V = 0			50	2	–	–	–	–	–	–	
BVS	BRANCH ON V = 1			70	2	–	–	–	–	–	–	
NOP	NO OPERATION	EA	2			–	–	–	–	–	–	

(1) ADD 1 TO "Cy" IF PAGE BOUNDARY IS CROSSED ON A LOAD OR LOGICAL.

(2) ADD 1 TO "Cy" IF BRANCH OCCURS TO SAME PAGE.
ADD 2 TO "Cy" IF BRANCH OCCURS TO DIFFERENT PAGE.

(3) CARRY NOT = BORROW (~C).

(4) IF IN DECIMAL MODE Z FLAG IS INVALID.

NOTE 1: The next executable address is pushed onto the stack and the program counter (PC) is loaded with the next two bytes of the instruction.

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS

Ms MEMORY PER STACK POINTER

\+ ADD
\- SUBTRACT
∧ AND

V OR
⊻ EXCLUSIVE OR
✓ MODIFIED
-- NOT MODIFIED
M7 MEMORY BIT 7
M6 MEMORY BIT 6
Cy NUMBER OF CYCLES
S STACK POINTER

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD \ LSD
0	BRK	ORA·IND,X				ORA·Z.Page	ASL·Z.Page		PHP	ORA·IMM	ASL·ACC			ORA·ABS	ASL·ABS		0
1	BPL	ORA·IND,Y				ORA·Z.Page,X	ASL·Z.Page,X		CLC	ORA·ABS,Y				ORA·ABS,X	ASL·ABS,X		1
2	JSR	AND·IND,X			BIT·Z.Page	AND·Z.Page	ROL·Z.Page		PLP	AND·IMM	ROL·ACC		BIT·ABS	AND·ABS	ROL·ABS		2
3	BMI	AND·IND,Y				AND·Z.Page,X	ROL·Z.Page,X		SEC	AND·ABS,Y				AND·ABS,X	ROL·ABS,X		3
4	RTI	EOR·IND,X				EOR·Z.Page	LSR·Z.Page		PHA	EOR·IMM	LSR·ACC		JMP·ABS	EOR·ABS	LSR·ABS		4
5	BVC	EOR·IND,Y				EOR·Z.Page,X	LSR·Z.Page,X		CLI	EOR·ABS,Y				EOR·ABS,X	LSR·ABS,X		5
6	RTS	ADC·IND,X				ADC·Z.Page	ROR·Z.Page		PLA	ADC·IMM	ROR·ACC		JMP·IND	ADC·ABS	ROR·ABS		6
7	BVS	ADC·IND,Y				ADC·Z.Page,X	ROR·Z.Page,X		SEI	ADC·ABS,Y				ADC·ABS,X	ROR·ABS,X		7
8	BRK	STA·IND,X			STY·Z.Page	STA·Z.Page	STX·Z.Page		DEY		TXA		STY ABS	STA·ABS	STX·ABS		8
9	BCC	STA·IND,Y			STY·Z.Page,X	STA·Z.Page,X	STX·Z.Page,X		TYA	STA·ABS,Y	TXS			STA·ABS,X			9
A	LDY·IMM	LDA·IND,X	LDX IMM		LDY·Z.Page	LDA·Z.Page	LDX·Z.Page		TAY	LDA·IMM	TAX		LDY·ABS	LDA·ABS	LDX·ABS,Y		A
B	BCS	LDA·IND,Y			LDY·Z.Page,X	LDA·Z.Page,X	LDX·Z.Page,X		CLV	LDA·ABS,Y	TSX		LDY·ABS,X	LDA·ABS,X	LDX·ABS		B
C	CPY·IMM	CMP·IND,X			CPY·Z.Page	CMP·Z.Page	DEC·Z.Page		INY	CMP·IMM	DEX		CPY·ABS	CMP·ABS	DEC·ABS		C
D	BNE	CMP·IND,Y				CMP·Z.Page,X	DEC·Z.Page,X		CLD	CMP·ABS,Y				CMP·ABS,X	DEC·ABS,X		D
E	CPX·IMM	SBC·IND,X			CPX·Z.Page	SBC·Z.Page	INC·Z.Page		INX	SBC·IMM	NOP		CPX·ABS	SBC·ABS	INC·ABS		E
F	BEQ	SBC·IND,Y				SBC·Z.Page,X	INC·Z.Page,X		SED	SBC·ABS,Y				SBC·ABS,X	INC·ABS,X		F

IMM • Immediate Addressing — The operand is contained in the second byte of the instruction

ABS • Absolute addressing — The second byte of the instruction contains the 8 low order bits of the effective address. The third byte contains the 8 high order bits of the effective address. [EA]

Z.PAGE • ZERO PAGE ADDRESSING — Second byte contains the 8 low order bits of the effective address. 8 high-order byte of the address is assumed to be 0 - i.e., points to page zero.

Z.PAGE,X Z.PAGE,Y • ZERO PAGE INDEXED — The second byte of the instruction is added modulus 8 to the index (X or Y) to form the lower order 8 bits of the EA. High-order 8 bits are zero - same as Zero Page Addressing.

A • ACCUMULATOR — One byte instruction operating on the accumulator.

ABS,X ABS,Y • ABSOLUTE INDEXED — The EA is formed by adding the index to the second byte and third bytes of the instruction.

(IND,X) • INDEXED INDIRECT — The X index is added modulo 8 to the second byte of the instruction. The result points to a zero page addess that contains the 8 lower bits of the EA. The next byte contains the 8 high order bits of the EA.

(IND),Y • INDIRECT INDEXED — The second byte of the instruction points to a zero page location. The Y index is added to the contents of this location which operation forms the 8 lower order bits of the EA. The carry from this operation is added to the contents of the next zero page location to form the 8 higher-order bits of the EA.

Font: Leroy Lettering via Coniglio Sublime