

Hex to Instruction Conversion

LSD →

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

Memory Map

ZERO PAGE	0000
	00FF
DATA & STACK*	0100
	01FF
	0200
RAM I/O ROM	
	FFF9
NMI VECTOR	FFFA&B
RES VECTOR	FFFC&D
IRQ VECTOR	FFFE&F

*In systems with < 512 bytes of RAM the hardware can ignore signal AB8, moving stack into page zero.

Status Flags

MSB	LSB
NV-BDIZC	
N=negative result	
V=overflow	
B=BRK instruction	
D=decimal mode	
I=IRQ disable	
Z=zero result	
C=carry-borrow	

Note: above is true when flag = 1.

Overflow normally signifies signed arithmetic result is out of range.

When D=1, only ADC and SBC use decimal (BCD) arithmetic.

Interrupts

IRQ is low level sensitive. NMI is falling edge sensitive. Reset sets I=1.

- Interrupts are processed by:
1. Push PC of unexecuted instruction.
 2. Push P.
 3. I=1.
 4. Jump via appropriate vector.

Effect on Flags

	NV	B	D	I	Z	C
ADC	NV	-	-	-	-	Z C ①
AND	N	-	-	-	-	Z
ANL	N	-	-	-	-	Z C
BIT	NV	-	-	-	-	Z ②
BRK	-	-	-	-	-	1 - - -
CLC	-	-	-	-	-	0
CLD	-	-	-	-	-	0
CLI	-	-	-	-	-	0
CLV	-	0	-	-	-	-
CMP	N	-	-	-	-	Z C
CPX	N	-	-	-	-	Z C
CPY	N	-	-	-	-	Z C
DEC	N	-	-	-	-	Z
DEX	N	-	-	-	-	Z
DEY	N	-	-	-	-	Z
EOR	N	-	-	-	-	Z
INC	N	-	-	-	-	Z
INX	N	-	-	-	-	Z
INY	N	-	-	-	-	Z
LDA	N	-	-	-	-	Z
LDX	N	-	-	-	-	Z
LDY	N	-	-	-	-	Z
LSR	0	-	-	-	-	Z C
ORA	N	-	-	-	-	Z
PLA	N	-	-	-	-	Z
PLP	NV	-	-	-	-	B D I Z C
ROL	N	-	-	-	-	Z C
ROR	N	-	-	-	-	Z C
RTI	NV	-	-	-	-	B D I Z C
SBC	NV	-	-	-	-	Z C ③
SEC	-	-	-	-	-	1
SED	-	-	-	-	-	1
SEI	-	-	-	-	-	1
TAX	N	-	-	-	-	Z
TAY	N	-	-	-	-	Z
TSX	N	-	-	-	-	Z
TXA	N	-	-	-	-	Z
TYA	N	-	-	-	-	Z

- ① If in decimal mode Z flag is invalid.
- ② N = data bit 7
V = data bit 6
Z = AND result
- ③ C = borrow

Note: unlisted instructions have no effect on flags

Addressing Modes

Note: Full 2 byte addresses in code, stack, and data areas are stored low byte followed by high byte. Thus, in hex, JMP \$1234 is: 4C 34 12.

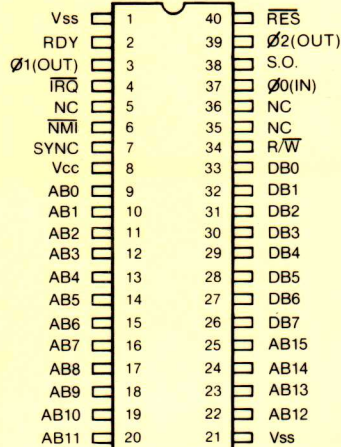
FORM	ADDRESSING	DESCRIPTION
nn	Absolute	Location nn holds data.
nn,X	Absolute X	Location nn+X holds data.
nn,Y	Absolute Y	Location nn+Y holds data.
A	Accumulator	Accumulator holds data.
#n	Immediate	n is data.
(n,X)	Ind X	Location n+X and next of page 0 hold address of data.**
(n),Y	Ind Y	Address of data is Y + address held by location n and next of page 0.**
(nn)	Indirect	Location nn and next hold address to jump to.**
n	Relative	Address to jump to is n + address of next instruction, with n treated as a signed number.
n	Zero Page	Location n of page 0 holds data.
n,X	Zero Page X	Location n+X of page 0 holds data.
n,Y	Zero Page Y	Location n+Y of page 0 holds data.

*n+X is computed discarding any carry.
 **2 bytes must not cross page boundary.

ASCII Character Set

MSD	0	1	2	3	4	5	6	7
LSD	000	001	010	011	100	101	110	111
0	0000 NUL	DLE	SP	0	@	P	.	p
1	0001 SOH	DC1	!	1	A	Q	a	q
2	0010 STX	DC2	"	2	B	R	b	r
3	0011 ETX	DC3	#	3	C	S	c	s
4	0100 EOT	DC4	\$	4	D	T	d	t
5	0101 ENQ	NAK	%	5	E	U	e	u
6	0110 ACK	SYN	&	6	F	V	f	v
7	0111 BEL	ETB	'	7	G	W	g	w
8	1000 BS	CAN	(8	H	X	h	x
9	1001 HT	EM)	9	I	Y	i	y
A	1010 LF	SUB	*	:	J	Z	j	z
B	1011 VT	ESC	+	:	K	[k	}
C	1100 FF	FS	<	L	\	l		l
D	1101 CR	GS	=	M]	m	~	}
E	1110 SO	RS	>	N	^	n	~	}
F	1111 SI	US	/	O	_	o	DEL	}

6502 Pins



Miscellaneous

S points to next free byte of stack.
 Stack push decrements S.
 In pushing PC, high byte is pushed first.

Pre 6/76 chips have no ROR instruction.

65XX is a totally software compatible family.

This card is based on specifications from MOS Technology, Inc.

Registers

- A ACCUMULATOR
- Y Y INDEX REG
- X X INDEX REG
- PC PROGRAM COUNTER
- S STACK PNTR
- P FLAGS

A, Y, X, S, P = 1 byte.
 Only PC is 2 bytes.

Unsigned Comparisons

example: CMP #n

A < n	BCC YES
A = n	BEQ YES
A > n	BCC NO
A ≥ n	BNE YES
A ≠ n	BNE YES
A ≤ n	BCC YES
A ≤ n	BEQ YES

YES represents label for code to be executed if condition is true. For > & ≤, test requires both instructions.

Internally, A-n is computed to determine N,Z,C flags.

Abbreviations

B = number of Bytes
 C = number of Cycles, also Carry.

n = 1 byte quantity
 nn = 2 byte quantity

IRQ = Interrupt ReQuest
 NMI = Non Maskable Interrupt
 RES = RESet
 XOR = eXclusive OR
 (00>0 01>1 10>1 11>0)

A,P,S,X,Y,PC= see "Registers"
 N,V,B,D,I,Z,C = see "Status Flags"
 #,\$@,%(!); = see "Assembler Symbols"

100% PLASTIC

INST ACC-SS

INSTRUCTION SET

INSTRUCTION	OP	C	B	DESCRIPTION	ADDRESSING	INSTRUCTION	OP	C	B	DESCRIPTION	ADDRESSING
ADC #n	69	2	2	Add with carry to A	Immediate	LDA #n	A9	2	2	Load A	Immediate
ADC nn	6D	4	3	Add with carry to A	Absolute	LDA nn	AD	4	3	Load A	Absolute
ADC n	65	3	2	Add with carry to A	Zero Page	LDA n	A5	3	2	Load A	Zero Page
ADC (n,X)	61	6	2	Add with carry to A	Ind X	LDA (n,X)	A1	6	2	Load A	Ind X
ADC (n),Y	71	5+	2	Add with carry to A	Ind Y	LDA (n),Y	B1	5+	2	Load A	Ind Y
ADC n,X	75	4	2	Add with carry to A	Zero Page X	LDA n,X	B5	4	2	Load A	Zero Page X
ADC nn,X	7D	4+	3	Add with carry to A	Absolute X	LDA nn,X	BD	4+	3	Load A	Absolute X
ADC nn,Y	79	4+	3	Add with carry to A	Absolute Y	LDA nn,Y	B9	4+	3	Load A	Absolute Y
AND #n	29	2	2	AND to A	Immediate	LDX #n	A2	2	2	Load X	Immediate
AND nn	2D	4	3	AND to A	Absolute	LDX nn	AE	4	3	Load X	Absolute
AND n	25	3	2	AND to A	Zero Page	LDX n	A6	3	2	Load X	Zero Page
AND (n,X)	21	6	2	AND to A	Ind X	LDX n,Y	BE	4+	3	Load X	Absolute Y
AND (n),Y	31	5+	2	AND to A	Ind Y	LDX nn,Y	B6	4	2	Load X	Zero Page Y
AND n,X	35	4	2	AND to A	Zero Page X	LDY #n	A0	2	2	Load Y	Immediate
AND nn,X	3D	4+	3	AND to A	Absolute X	LDY nn	AC	4	3	Load Y	Absolute
AND nn,Y	39	4+	3	AND to A	Absolute Y	LDY n	A4	3	2	Load Y	Zero Page
ASL nn	0E	6	3	Arithmetic shift left	Absolute	LDY n,X	B4	4	2	Load Y	Zero Page X
ASL n	06	5	2	Arithmetic shift left	Zero Page	LDY nn,X	BC	4+	3	Load Y	Absolute X
ASL A	0A	2	1	Arithmetic shift left	Accumulator	LSR nn	4E	6	3	Logical shift right	Absolute
ASL nn,X	16	6	2	Arithmetic shift left	Zero Page X	LSR n	46	5	2	Logical shift right	Zero Page
ASL nn,Y	1E	7	3	Arithmetic shift left	Absolute X	LSR A	4A	2	1	Logical shift right	Accumulator
BCC n	90	2+	2	Branch if carry clear (C=0)	Relative	LSR n,X	56	6	2	Logical shift right	Zero Page X
BCS n	B0	2+	2	Branch if carry set (C=1)	Relative	LSR nn,X	5E	7	3	Logical shift right	Absolute X
BEQ n	F0	2+	2	Branch if equal (Z=1)	Relative	NOP	EA	2	1	No operation	None
BNE n	D0	2+	2	Branch if not equal (Z=0)	Relative	ORA #n	09	2	2	OR to A	Immediate
BMI n	30	2+	2	Branch if minus (N=1)	Relative	ORA nn	0D	4	3	OR to A	Absolute
BPL n	10	2+	2	Branch if plus (N=0)	Relative	ORA n	05	3	2	OR to A	Zero Page
BVC n	50	2+	2	Branch if ovfl clear (V=0)	Relative	ORA (n,X)	01	6	2	OR to A	Ind X
BVS n	70	2+	2	Branch if ovfl set (V=1)	Relative	ORA (n),Y	11	5+	2	OR to A	Ind Y
BIT nn	2C	4	3	AND with A (A unchanged)	Absolute	ORA n,X	15	4	2	OR to A	Zero Page X
BIT n	24	3	2	AND with A (A unchanged)	Zero Page	ORA nn,X	1D	4+	3	OR to A	Absolute X
BRK	00	7	1	Break (force interrupt)	None	ORA nn,Y	19	4+	3	OR to A	Absolute Y
CLC	18	2	1	Clear carry	None	PHA	48	3	1	Push A onto stack	None
CLD	D8	2	1	Clear decimal mode	None	PHP	08	3	1	Push P onto stack	None
CLI	58	2	1	Clear IRQ disable	None	PLA	68	4	1	Pull (pop) A from stack	None
CLV	B8	2	1	Clear overflow	None	PLP	28	4	1	Pull (pop) P from stack	None
CMP #n	C9	2	2	Compare with A	Immediate	ROL nn	2E	6	3	Rotate left through carry	Absolute
CMP nn	CD	4	3	Compare with A	Absolute	ROL n	26	5	2	Rotate left through carry	Zero Page
CMP n	C5	3	2	Compare with A	Zero Page	ROL A	2A	2	1	Rotate left through carry	Accumulator
CMP (n,X)	C1	6	2	Compare with A	Ind X	ROL n,X	36	6	2	Rotate left through carry	Zero Page X
CMP (n),Y	D1	5+	2	Compare with A	Ind Y	ROL nn,X	3E	7	3	Rotate left through carry	Absolute X
CMP n,X	D5	4	2	Compare with A	Zero Page X	ROR nn	6E	6	3	Rotate right through carry	Absolute
CMP nn,X	DD	4+	3	Compare with A	Absolute X	ROR n	66	5	2	Rotate right through carry	Zero Page
CMP nn,Y	D9	4+	3	Compare with A	Absolute Y	ROR A	6A	2	1	Rotate right through carry	Accumulator
CPX #n	E0	2	2	Compare with X	Immediate	ROR n,X	76	6	2	Rotate right through carry	Zero Page X
CPX nn	EC	4	3	Compare with X	Absolute	ROR nn,X	7E	7	3	Rotate right through carry	Absolute X
CPX n	E4	3	2	Compare with X	Zero Page	RTI	40	6	1	Return from interrupt	None
CPY #n	C0	2	2	Compare with Y	Immediate	RTS	60	6	1	Return from subroutine	None
CPY nn	CC	4	3	Compare with Y	Absolute	SBC #n	E9	2	2	Subtract with borrow from A	Immediate
CPY n	C4	3	2	Compare with Y	Zero Page	SBC nn	ED	4	3	Subtract with borrow from A	Absolute
DEC nn	CE	6	3	Decrement by one	Absolute	SBC n	E5	3	2	Subtract with borrow from A	Zero Page
DEC n	C6	5	2	Decrement by one	Zero Page	SBC (n,X)	E1	6	2	Subtract with borrow from A	Ind X
DEC n,X	D6	6	2	Decrement by one	Zero Page X	SBC (n),Y	F1	5+	2	Subtract with borrow from A	Ind Y
DEC nn,X	DE	7	3	Decrement by one	Absolute X	SBC n,X	F5	4	2	Subtract with borrow from A	Zero Page X
DEX	CA	2	1	Decrement X by one	None	SBC nn,X	FD	4+	3	Subtract with borrow from A	Absolute X
DEY	8A	2	1	Decrement Y by one	None	SBC nn,Y	F9	4+	3	Subtract with borrow from A	Absolute Y
EOR #n	49	2	2	XOR to A	Immediate	SEC	38	2	1	Set carry	None
EOR nn	4D	4	3	XOR to A	Absolute	SED	F8	2	1	Set decimal mode	None
EOR n	45	3	2	XOR to A	Zero Page	SEI	78	2	1	Set IRQ disable	None
EOR (n,X)	41	6	2	XOR to A	Ind X	STA nn	8D	4	3	Store A	Absolute
EOR (n),Y	51	5+	2	XOR to A	Ind Y	STA n	85	3	2	Store A	Zero Page
EOR n,X	55	4	2	XOR to A	Zero Page X	STA (n,X)	81	6	2	Store A	Ind X
EOR nn,X	5D	4+	3	XOR to A	Absolute X	STA (n),Y	91	6	2	Store A	Ind Y
EOR nn,Y	59	4+	3	XOR to A	Absolute Y	STA n,X	95	4	2	Store A	Zero Page X
INC nn	EE	6	3	Increment by one	Absolute	STA nn,X	9D	5	3	Store A	Absolute X
INC n	E6	5	2	Increment by one	Zero Page	STA nn,Y	99	5	3	Store A	Absolute Y
INC n,X	F6	6	2	Increment by one	Zero Page X	STX nn	8E	4	3	Store X	Absolute
INC nn,X	FE	7	3	Increment by one	Absolute X	STX n	86	3	2	Store X	Zero Page
INX	E8	2	1	Increment X by one	None	STX n,Y	96	4	2	Store X	Zero Page Y
INY	C8	2	1	Increment Y by one	None	STY nn	8C	4	3	Store Y	Absolute
JMP nn	4C	3	3	Jump to new location	Absolute	STY n	84	3	2	Store Y	Zero Page
JMP (nn)	6C	5	3	Jump to new location	Indirect	STY n,X	94	4	2	Store Y	Zero Page X
JSR nn	20	6	3	Jump to subroutine	Absolute	TAX	AA	2	1	Transfer A to X	None
						TAY	A8	2	1	Transfer A to Y	None
						TSX	BA	2	1	Transfer S to X	None
						TXA	8A	2	1	Transfer X to A	None
						TXS	9A	2	1	Transfer X to S	None
						TYA	98	2	1	Transfer Y to A	None

DO NOT PLACE ON HOT SURFACES

COPYRIGHT © 1980 AND PUBLISHED BY MICRO LOGIC CORP. POB 174 HACKENSACK, N.J. 07602 ALL RIGHTS RESERVED-PRINTED IN U.S.A.

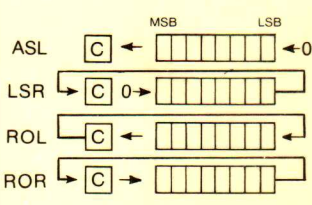
WHILE PREPARED WITH EXTREME CARE, THERE ARE NO WARRANTIES EXPRESSED OR IMPLIED AS TO MERCHANTABILITY OR FITNESS FOR PURPOSE.

MICRO LOGIC CORP. JMM, A. LEWIS

Instruction Notes

ADC	A+DATA+C→A
BRK	Ignore I flag, Set B=1 Push return address+1 Push P Jump to IRQ vector
JSR	Push return address-1 Jump absolute
RTI	Pop P, Pop PC
RTS	Pop PC, Increment PC
SBC	A-DATA-C→A

Shift Instructions



Added Cycle Time

A (+) in the (C) column for branch instructions means:
Add 0 if branch not taken.
Add 1 if taken within page.
Add 2 if taken across pages.

A (+) in the (C) column for other instructions means:
Add 1 if indexing across page boundary.

Assembler Symbols

- . Assembler directive
- # Immediate addressing
- \$ Hex number prefix
- @ Octal number prefix
- % Binary number prefix
- ' ASCII character prefix
- () Indirect addressing
- ; In col 1 for comment