2014

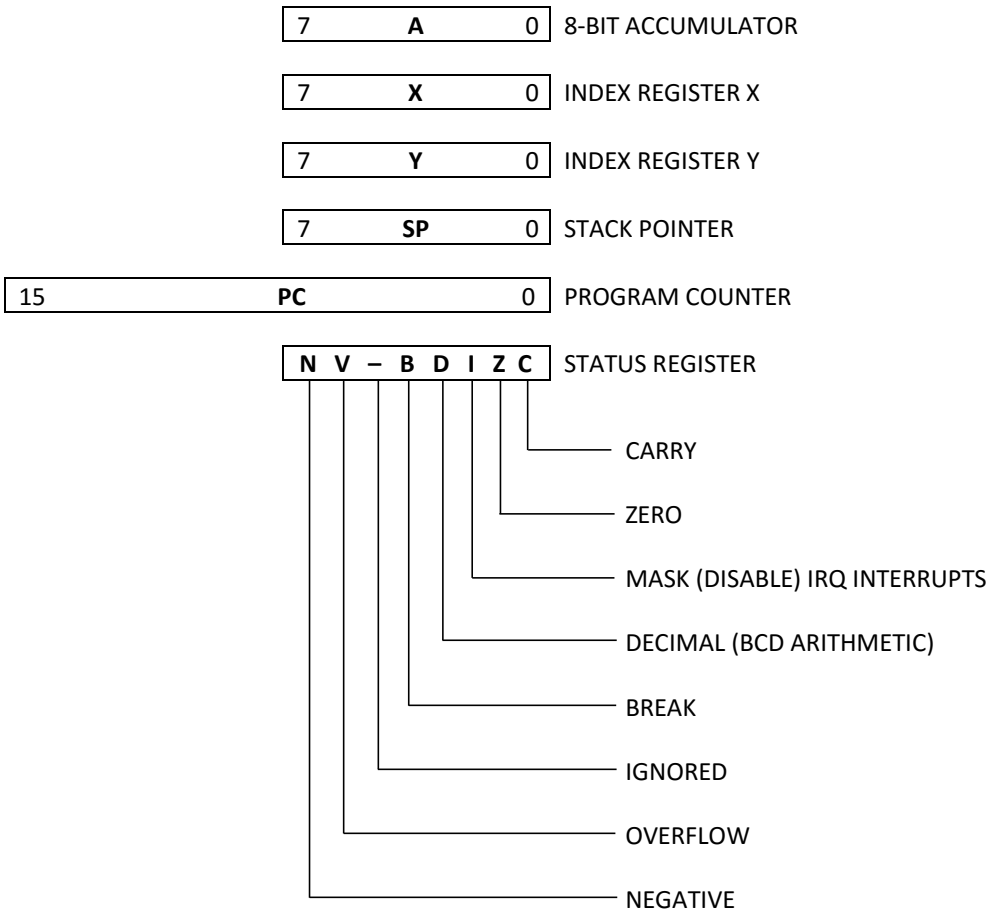
6502 Reference Guide

MOS TECHNOLOGY

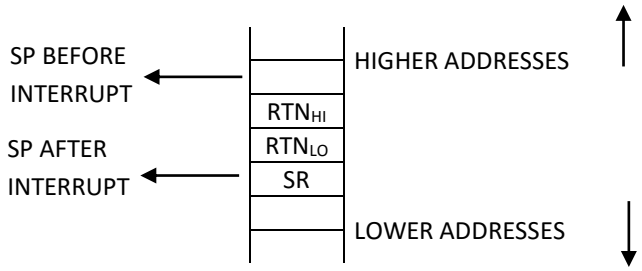
ALEX BUSMAN

6502 INSTRUCTION SET

PROGRAMMING MODEL



STACK AND MEMORY LAYOUT



INTERRUPT VECTOR LOCATIONS

$FFFE, $FFFF	IRQ/BRK: Interrupt Request or Break
$FFFC, $FFFD	RESET: Power-On (POR) or External Reset
$FFFA, $FFFB	NMI: Non-maskable Interrupt

NOTATION USED IN INSTRUCTION SET SUMMARY

CPU Register Notation

Accumulator – A or a
Index Register X – X or x
Index Register Y – Y or y
Stack Pointer – SP, sp, or s
Program Counter – PC, pc, or p
Status Register/Condition Code Register – SR, CCR, c

Explanation of Italic Expressions in Source Form Column

opr8i	–	8-bit immediate value
opr8a	–	8-bit address used with zero page address mode
opr16a	–	16-bit address value
oprx8	–	Any integer in the range -128 … +127
oprx16	–	Any integer in the range -32,768 … +65,535
xy	–	Index X or Index Y

Operators

+	–	Addition
-	–	Subtraction
·	–	Logical AND
+	–	Logical OR (inclusive)
⊕	–	Logical exclusive OR
:	–	Concatenate
⇒	–	Transfer

Address Mode Notation

ACC	–	Accumulator; no operands; operation performed on accumulator.
IMP	–	Implied; no operands in object code
IMM	–	Immediate; operand in object code
ZER	–	Zero Page; operand is the lower byte of an address from $0000 to $00FF
ABS	–	Absolute; operand is a 16-bit address
REL	–	Two's complement relative offset; for branch instructions
IDX1	–	8-bit signed offset from X or Y; 1 extension byte
IDX2	–	16-bit signed offset from X or Y; 2 extensions bytes
[IDX1]	–	Indexed-indirect; 8-bit offset from X or Y

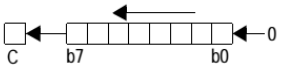
Machine Coding

`ee`	–	High-order byte of a 16-bit constant offset for indexed addressing.
`ff`	–	Low-order eight bits of an 8-bit signed constant offset for indexed addressing, or low-order byte of a 16-bit constant offset for indexed addressing.
`hh`	–	High-order byte of a 16-bit address.
`ii`	–	8-bit immediate data value.
`ll`	–	Low-order byte of a 16-bit extended address.
`rr`	–	Signed relative offset $80 (-128) to $7F (+127). Offset relative to the byte following the relative offset byte.
`zz`	–	8-bit zero page address $0000 to $00FF. (High byte assumed to be $00).

Condition Codes Columns

–	–	Status bit not affected by operation.
0	–	Status bit cleared by operation.
1	–	Status bit set by operation.
Δ	–	Status bit affected by operation.

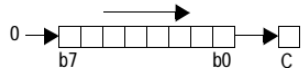
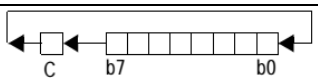
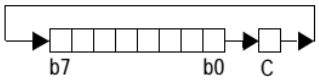
INSTRUCTION SET SUMMARY

Source Form	Operation	Addr. Mode	Machine Coding (hex)	B D I	N Z V C	Cycles
ADC *#opr8i*	(A) + (M) + C ⇒ A	IMM	69 ii	– – –	Δ Δ Δ Δ	2
ADC *opr8a*	Add Memory to Accumulator with Carry	ZER	65 zz			3
ADC *opr16a*		ABS	6D hh ll			4
ADC *oprx8,x*		IDX1	75 ff			4
ADC *oprx16,x*		IDX2	7D ee ff			4*
ADC *oprx16,y*		IDX2	79 ee ff			4*
ADC *(oprx8,x)*		[IDX1]	61 ff			6
ADC *(oprx8),y*		[IDX1]	71 ff			5*
AND *#opr8i*	(A) · (M) ⇒ A	IMM	29 ii	– – –	Δ Δ – –	2
AND *opr8a*	AND Memory with Accumulator	ZER	25 zz			3
AND *opr16a*		ABS	2D hh ll			4
AND *oprx8,x*		IDX1	35 ff			4
AND *oprx16,x*		IDX2	3D ee ff			4*
AND *oprx16,y*		IDX2	39 ee ff			4*
AND *(oprx8,x)*		[IDX1]	21 ff			6
AND *(oprx8),y*		[IDX1]	31 ff			5*
ASL A		ACC	0A	– – –	Δ Δ – Δ	2
ASL *opr8a*		ZER	06 zz			5
ASL *opr16a*		ABS	0E hh ll			6
ASL *oprx8,x*	Arithmetic Shift Left (Memory or A)	IDX1	16 ff			6
ASL *oprx16,x*		IDX2	1E ee ff			7
BCC *rel8*	Branch if Carry Clear (if C = 0)	REL	90 rr	– – –	– – – –	2**
BCS *rel8*	Branch if Carry Set (if C = 1)	REL	B0 rr	– – –	– – – –	2**
BEQ *rel8*	Branch if Equal (if Z = 1)	REL	F0 rr	– – –	– – – –	2**
BIT *opr8a*	(A) · (M)	ZER	24 zz	– – –	Δ Δ Δ –	3
BIT *opr16a*	Test Bits in Memory with Accumulator	ABS	2C hh ll			4
BMI *rel8*	Branch if Minus (if N = 1)	REL	30 rr	– – –	– – – –	2**
BNE *rel8*	Branch if Not Equal (if Z = 0)	REL	D0 rr	– – –	– – – –	2**
BPL *rel8*	Branch if Plus (if N = 0)	REL	10 rr	– – –	– – – –	2**
BRK	Force Break	IMP	00	1 – 1	– – – –	7
BVC *rel8*	Branch if Overflow Bit Clear (if V = 0)	REL	50 rr	– – –	– – – –	2**
BVS *rel8*	Branch if Overflow Bit Set (if V = 1)	REL	70 rr	– – –	– – – –	2**
CLC	0 ⇒ C Clear Carry Flag	IMP	18	– – –	– – – 0	2
CLD	0 ⇒ D Clear Decimal Mode	IMP	D8	– 0 –	– – – –	2
CLI	0 ⇒ I Clear Interrupt Disable Bit	IMP	58	– – 0	– – – –	2
CLV	0 ⇒ V Clear Overflow Flag	IMP	B8	– – –	– – 0 –	2
CMP *#opr8i*	(A) – (M)	IMM	C9 ii	– – –	Δ Δ Δ –	2
CMP *opr8a*	Compare Memory with Accumulator	ZER	C5 zz			3
CMP *opr16a*		ABS	CD hh ll			4
CMP *oprx8,x*		IDX1	D5 ff			4
CMP *oprx16,x*		IDX2	DD ee ff			4*
CMP *oprx16,y*		IDX2	D9 ee ff			4*
CMP *(oprx8,x)*		[IDX1]	C1 ff			6
CMP *(oprx8),y*		[IDX1]	D1 ff			5*

Source Form	Operation	Addr. Mode	Machine Coding (hex)	B D I	N Z V C	Cycles
CPX *#opr8i*	(X) – (M)	IMM	E0 ii	– – –	Δ Δ Δ –	2
CPX *opr8a*	Compare Memory and Index X	ZER	E4 zz			3
CPX *opr16a*		ABS	EC hh ll			4
CPY *#opr8i*	(Y) – (M)	IMM	C0 ii	– – –	Δ Δ Δ –	2
CPY *opr8a*	Compare Memory and Index Y	ZER	C4 zz			3
CPY *opr16a*		ABS	CC hh ll			4
DEC *opr8a*	(M) – $01 ⇒ M	ZER	C6 zz	– – –	Δ Δ – –	5
DEC *opr16a*	Decrement Memory by 1	ABS	CE hh ll			6
DEC *oprx8,x*		IDX1	D6 ff			6
DEC *oprx16,x*		IDX2	DE ee ff			7
DEX	(X) – $01 ⇒ X	IMP	CA	– – –	Δ Δ – –	2
	Decrement Index X by 1					
DEY	(Y) – $01 ⇒ Y	IMP	88	– – –	Δ Δ – –	2
	Decrement Index Y by 1					
EOR *#opr8i*	(A) ⊕ (M) ⇒ A	IMM	49 ii	– – –	Δ Δ – –	2
EOR *opr8a*	Exclusive-OR Memory with Accumulator	ZER	45 zz			3
EOR *opr16a*		ABS	4D hh ll			4
EOR *oprx8,x*		IDX1	55 ff			4
EOR *oprx16,x*		IDX2	5D ee ff			4*
EOR *oprx16,y*		IDX2	59 ee ff			4*
EOR *(oprx8,x)*		[IDX1]	41 ff			6
EOR *(oprx8),y*		[IDX1]	51 ff			5*
INC *opr8a*	(M) + $01 ⇒ M	ZER	E6 zz	– – –	Δ Δ – –	5
INC *opr16a*	Increment Memory by 1	ABS	EE hh ll			6
INC *oprx8,x*		IDX1	F6 ff			6
INC *oprx16,x*		IDX2	FE ee ff			7
INX	(X) + $01 ⇒ X	IMP	E8	– – –	Δ Δ – –	2
	Increment Index X by 1					
INY	(Y) + $01 ⇒ Y	IMP	C8	– – –	Δ Δ – –	2
	Increment Index Y by 1					
JMP *opr16a*	Jump to New Location	ABS	4C hh ll	– – –	– – – –	3
JMP *(opr16a)*	Routine Address ⇒ PC	IND	6C ee ff			5
JSR *opr16a*	Jump to Subroutine Saving Return Address	ABS	20 hh ll	– – –	– – – –	6
	(SP) – 2 ⇒ SP					
	RTN_{HI}:RTN_{LO} ⇒ $M_{(SP+2)}$:$M_{(SP+1)}$					
	Subroutine address ⇒ PC					
LDA *#opr8i*	(M) ⇒ A	IMM	A9 ii	– – –	Δ Δ – –	2
LDA *opr8a*	Load Accumulator with Memory	ZER	A5 zz			3
LDA *opr16a*		ABS	AD hh ll			4
LDA *oprx8,x*		IDX1	B5 ff			4
LDA *oprx16,x*		IDX2	BD ee ff			4*
LDA *oprx16,y*		IDX2	B9 ee ff			4*
LDA *(oprx8,x)*		[IDX1]	A1 ff			6
LDA *(oprx8),y*		[IDX1]	B1 ff			5*
LDX *#opr8i*	(M) ⇒ X	IMM	A2 ii	– – –	Δ Δ – –	2
LDX *opr8a*	Load Index X with Memory	ZER	A6 zz			3
LDX *opr16a*		ABS	AE hh ll			4
LDX *oprx8,y*		IDX1	B6 ff			4
LDX *oprx16,y*		IDX2	BE ee ff			4*

Source Form	Operation	Addr. Mode	Machine Coding (hex)	B D I	N Z V C	Cycles

LDY *#opr8i*
LDY *opr8a*
LDY *opr16a*
LDY *oprx8,x*
LDY *oprx16,x*
	(M) ⇒ Y
	Load Index Y with Memory
		IMM
		ZER
		ABS
		IDX1
		IDX2
			A0 ii
			A4 zz
			AC hh ll
			B4 ff
			BC ee ff
				– – –	Δ Δ – –	2
						3
						4
						4
						4*

LSR A LSR *opr8a* LSR *opr16a* LSR *oprx8,x* LSR *oprx16,x*	 Logical Shift Right	ACC ZER ABS IDX1 IDX2	4A 46 zz 4E hh ll 56 ff 5E ee ff	– – –	– Δ – Δ	2 5 6 6 7
NOP	No Operation	IMP	EA	– – –	– – – –	2
ORA *#opr8i* ORA *opr8a* ORA *opr16a* ORA *oprx8,x* ORA *oprx16,x* ORA *oprx16,y* ORA *(oprx8,x)* ORA *(oprx8),y*	(A) + (M) ⇒ A Logical OR Memory with Accumulator	IMM ZER ABS IDX1 IDX2 IDX2 [IDX1] [IDX1]	09 ii 05 zz 0D hh ll 15 ff 1D ee ff 19 ee ff 01 ff 11 ff	– – –	Δ Δ – –	2 3 4 4 4* 4* 6 5*
PHA	(A) ⇒ $M_{(SP)}$ (SP) – 1 ⇒ SP Push Accumulator on Stack	IMP	48	– – –	– – – –	3
PHP	(SR) ⇒ $M_{(SP)}$ (SP) – 1 ⇒ SP Push Processor Status on Stack	IMP	08	– – –	– – – –	3
PLA	(SP) + 1 ⇒ SP ($M_{(SP)}$) ⇒ A Pull Accumulator from Stack	IMP	68	– – –	Δ Δ – –	4
PLP	(SP) + 1 ⇒ SP ($M_{(SP)}$) ⇒ SR Pull Processor Status from Stack	IMP	28	Δ Δ Δ	Δ Δ Δ Δ	4
ROL A ROL *opr8a* ROL *opr16a* ROL *oprx8,x* ROL *oprx16,x*	 Rotate Left through Carry	ACC ZER ABS IDX1 IDX2	2A 26 zz 2E hh ll 36 ff 3E ee ff	– – –	Δ Δ – Δ	2 5 6 6 7
ROR A ROR *opr8a* ROR *opr16a* ROR *oprx8,x* ROR *oprx16,x*	 Rotate Right through Carry	ACC ZER ABS IDX1 IDX2	6A 66 zz 6E hh ll 76 ff 7E ee ff	– – –	Δ Δ – Δ	2 5 6 6 7
RTI	(SP) + 1 ⇒ SP; ($M_{(SP)}$) ⇒ SR (SP) + 2 ⇒ SP; ($M_{(SP)}$:$M_{(SP-1)}$) ⇒ PC_{HI}:PC_{LO} Return from Interrupt	IMP	40	Δ Δ Δ	Δ Δ Δ Δ	6
RTS	(SP) + 2 ⇒ SP ($M_{(SP)}$:$M_{(SP-1)}$) + 1 ⇒ PC_{HI}:PC_{LO} Return from Subroutine	IMP	60	– – –	– – – –	6

Source Form	Operation	Addr. Mode	Machine Coding (hex)	B D I	N Z V C	Cycles
SBC *#opr8i* SBC *opr8a* SBC *opr16a* SBC *oprx8,x* SBC *oprx16,x* SBC *oprx16,y* SBC *(oprx8,x)* SBC *(oprx8),y*	(A) – (M) – (C) ⇒ A Subtract Memory from A with Borrow	IMM ZER ABS IDX1 IDX2 IDX2 [IDX1] [IDX1]	E9 ii E5 zz ED hh ll F5 ff FD ee ff F9 ee ff E1 ff F1 ff	– – –	Δ Δ Δ Δ	2 3 4 4 4* 4* 6 5*
SEC	1 ⇒ C Set Carry Flag	IMP	38	– – –	– – – 1	2
SED	1 ⇒ D Set Decimal Mode	IMP	F8	– 1 –	– – – –	2
SEI	1 ⇒ I Set Interrupt Disable Bit	IMP	78	– – 1	– – – –	2
STA *opr8a* STA *opr16a* STA *oprx8,x* STA *oprx16,x* STA *oprx16,y* STA *(oprx8,x)* STA *(oprx8),y*	(A) ⇒ M Store Accumulator in Memory	ZER ABS IDX1 IDX2 IDX2 [IDX1] [IDX1]	85 zz 8D hh ll 95 ff 9D ee ff 99 ee ff 81 ff 91 ff	– – –	– – – –	3 4 4 5 5 6 6
STX *opr8a* STX *opr16a* STX *oprx8,y*	(X) ⇒ M Store Index X in Memory	ZER ABS IDX1	86 zz 8E hh ll 96 ff	– – –	– – – –	3 4 4
STY *opr8a* STY *opr16a* STY *oprx8,x*	(Y) ⇒ M Store Index Y in Memory	ZER ABS IDX1	84 zz 8C hh ll 94 ff	– – –	– – – –	3 4 4
TAX	(A) ⇒ X Transfer Accumulator to Index X	IMP	AA	– – –	Δ Δ – –	2
TAY	(A) ⇒ Y Transfer Accumulator to Index Y	IMP	A8	– – –	Δ Δ – –	2
TSX	(SP) ⇒ X Transfer Stack Pointer to Index X	IMP	BA	– – –	Δ Δ – –	2
TXA	(X) ⇒ A Transfer Index X to Accumulator	IMP	8A	– – –	Δ Δ – –	2
TXS	(X) ⇒ SP Transfer Index X to Stack Pointer	IMP	9A	– – –	Δ Δ – –	2
TYA	(Y) ⇒ A Transfer Index Y to Accumulator	IMP	98	– – –	Δ Δ – –	2

* Add 1 to cycles if page boundary is crossed

** Add 1 to cycles if branch occurs on same page; Add 2 to cycles if branch occurs on different page.