6502 Instruction set

Load a byte value from memory into register

Flags affected: N, Z

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
lda	A	✓	✓	✓		✓	✓	✓	✓	✓
ldx	X	✓	✓		✓	✓		✓		
ldy	Y	✓	✓	✓		✓	✓			

Store a byte value from register into memory address

Flags affected: none

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
sta	-		✓	✓		✓	✓	✓	✓	✓
stx	-		✓		✓	✓		✓		
sty	-		✓	✓		✓	✓			

ADC: Add value to accumulator with carry
SBC: Subtract value from accumulator with borrow

Flags affected: N, Z, C, V

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
adc	A	✓	✓	✓		✓	✓	✓	✓	✓
sbc	A	✓	✓	✓		✓	✓	✓	✓	✓

* will add or subract one extra if the carry flag is set

Logical operations: AND, OR, XOR

Flags affected: N, Z

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
and	A	✓	✓	✓		✓	✓	✓	✓	✓
ora	A	✓	✓	✓		✓	✓	✓	✓	✓
eor	A	✓	✓	✓		✓	✓	✓	✓	✓

Bitwise operations on A or memory
ROL / ROR - rotate bits 1 left or right through carry
ASL / LSR - shift bits out to carry filling other side with zero

Flags affected: N, Z, C

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
rol	A		✓	✓		✓	✓	✓	✓	✓
ror	A		✓	✓		✓	✓	✓	✓	✓
asl	A		✓	✓		✓	✓	✓	✓	✓
lsr	A		✓	✓		✓	✓	✓	✓	✓

* asl and lsr are often used to multiply or divide byte with 2

Decrement or Increment memory or registers

Flags affected: N, Z

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
dec	-		✓	✓		✓	✓			
inc	-		✓	✓		✓	✓			
dex	X									
inx	X									
dey	Y									
iny	Y									

* note there is no dec/inc that works on A register (use sbc/adc)

Compare register and memory

Flags affected: N, Z, C

opcode	reg	Imm	zp	zp,x	zp,y	abs	abs,x	abs,y	(ind,x)	(ind),y
cmp	-	✓	✓	✓		✓	✓	✓	✓	✓
cpx	-	✓	✓	✓		✓	✓	✓	✓	✓
cpy	-	✓	✓	✓		✓	✓	✓	✓	✓

* actually subtract value from register and then affects N, Z, C flags used for branching after

Setting or clearing CPU flags

Flags affected: each respective flag

clc	-	clear carry
sec	-	set carry flag
cld	-	clear decimal flag
sed	-	set decimal flag
cli	-	clear interrupt flag
sei	-	set interrupt flag
clv	-	clear overflow flag *

* no set overflow flag exists

Conditional branching

Flags affected: none

bcc	-	on carry flag clear
bcs	-	on carry flag set
bne	-	on zero flag clear
beq	-	on zero flag set
bpl	-	on negative flag clear
bmi	-	on negative flag set
bvc	-	on overflow flag clear
bvs	-	on overflow flag set

Copy registers to other registers

Flags affected: N, Z

tax	X	copy A to X
tay	Y	copy A to Y
txa	A	copy X to A
tya	A	copy Y to A
tsx	X	copy stack pointer to X
txs	SP	copy X to stack pointer (no flags affected)

* no copying between x and y exists

Stack operations

pha	-	push A to stack
pla	A	pull byte from stack to register A (flags N,Z)
php	-	push status register to stack
plp	SR	pull byte from stack to CPU flags (all flags)

Jumping and subroutines

Flags affected: none

jmp	-	jump to new address *
jsr	-	jump to subroutine
rts	-	return from subroutine
rti	-	return from interrupt

* jmp has adressing mode absolute and indirect (rarely used)

Miscellaneous

brk	-	trigger brk interrupt
nop	-	no operation (just burns 2 cpu cycles)
bit	-	test bits *

* bit is very special it copies bit 7 of memory to N flag
and bit 6 to V flag as well as performing an AND between
A and memory only affecting Z flag (A is not modified)
Has zp as well as abs adressing modes

Registers (all 8 bit values)

A	Accumulator - for all arithmetic, logical and bit operations
X	Index X register - for offsetting addresses in indirect adressing
Y	Index Y register - for offsetting addresses in indirect adressing
SP	Stack pointer - for stack operations (jsr/rts)
SR	Status registers - also called CPU Flags or P register (NV-BDIZC)

PC	16 bit program counter - points to memory address where next instruction is

Adressing modes

Imm	Immediate	Byte value that is stored in register or used for logical operation
zp	Zero Page	8 bit address in memory
zp,x	Zero Page x indexed	8 bit address + x = actual memory address (wraps on page)
zp,y	Zero Page y indexed	8 bit address + y = actual memory address (wraps on page)
abs	Absolute	16 bit address in memory
abs,x	Absolute x indexed	16 bit address + x = actual memory address
abs,y	Absolute y indexed	16 bit address + y = actual memory address
(ind,x)	x indexed indirect	8 bit zp + x pointer used indirectly to get actual 16 bit address*
(ind),y	Indirect y indexed	8 bit zp pointer used indirectly + y to get actual 16 bit address*

* note that two consecutive bytes in zero page is used as a 16 bit address for these indirect modes
* some instructions has an implicit addressing mode and operates on a register (e.g. rol or dex)

CPU Flags (status register)

C	Carry flag - bit used by arithmetic and logical operations
Z	Zero flag - set if all bits in register used is 0
I	Interrupt flag - flag set to 1 if an interrupt happened
D	Decimal flag - flag can be adjusted to turn CPU decimal mode on/off
B	Break flag - set by CPU whenever a brk interrupt happened
-	(bit 5 in status register not used)
V	Overflow flag - flag is set if an arithmetic operation caused an overflow
N	Negative flag - set if bit7 of register used is 1

One instruction:

1 byte	0-2 bytes
OPCODE	**VALUE or ADDRESS**

Instructions with 3 bytes always have a 2 byte absolute memory address as parameter
CPU can read/write to all 64KB of memory with a 2 byte / 16 bit value address
Zero Page is first page of memory. Addresses 0 - 255. Faster instructions and indirect addressing.
Stack resides in the second page of memory. Addresses 256 - 511
Bit 7 of a byte is often called the sign bit and is used by some instructions to indicate if its negative
Adressing mode zp,y is only used by ldx and stx
CPU flags are generally only affected whenever the contents of a register changes

To distinguish between immediate and zero page address we write a # in front of the value

LDA #100	Load the byte value 100 into register A
LDA 100	Load the byte value from the address 100 into A
LDA 1000	Load the byte value from the address 1000 into A
~~LDA #1000~~	Immediate values can only be 0-255 as that is the size of a register

Not all opcode byte values have working instructions, these are often called "illegal opcodes"

More detailed online info including instruction cycle times:
https://www.masswerk.at/6502/6502_instruction_set.html