[Return to Main Page] NMOS 6502 Opcodes by John Pickens, Updated by Bruce Clark and by Ed Spittles
[Up to Tutorials and Aids]

INDEX

ADC (ADd with Carry)

Affects Flags: N V Z C

```
MODE           SYNTAX       HEX LEN TIM
Immediate     ADC #$44      $69  2   2
Zero Page     ADC $44       $65  2   3
Zero Page,X   ADC $44,X     $75  2   4
Absolute      ADC $4400     $6D  3   4
Absolute,X    ADC $4400,X   $7D  3   4+
Absolute,Y    ADC $4400,Y   $79  3   4+
Indirect,X    ADC ($44,X)   $61  2   6
Indirect,Y    ADC ($44),Y   $71  2   5+

+ add 1 cycle if page boundary crossed
```

ADC results are dependant on the setting of the decimal flag. In decimal mode, addition is carried out on the assumption that the values involved are packed BCD (Binary Coded Decimal).

There is no way to add without carry.

AND (bitwise AND with accumulator)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     AND #$44      $29  2   2
Zero Page     AND $44       $25  2   3
Zero Page,X   AND $44,X     $35  2   4
Absolute      AND $4400     $2D  3   4
Absolute,X    AND $4400,X   $3D  3   4+
Absolute,Y    AND $4400,Y   $39  3   4+
Indirect,X    AND ($44,X)   $21  2   6
Indirect,Y    AND ($44),Y   $31  2   5+
```

```
+ add 1 cycle if page boundary crossed
```

ASL (Arithmetic Shift Left)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Accumulator    ASL A        $0A  1   2
Zero Page      ASL $44      $06  2   5
Zero Page,X    ASL $44,X    $16  2   6
Absolute       ASL $4400    $0E  3   6
Absolute,X     ASL $4400,X  $1E  3   7
```

ASL shifts all bits left one position. 0 is shifted into bit 0 and the original bit 7 is shifted into the Carry.

BIT (test BITs)

Affects Flags: N V Z

```
MODE           SYNTAX       HEX LEN TIM
Zero Page      BIT $44      $24  2   3
Absolute       BIT $4400    $2C  3   4
```

BIT sets the Z flag as though the value in the address tested were ANDed with the accumulator. The N and V flags are set to match bits 7 and 6 respectively in the value stored at the tested address.

BIT is often used to skip one or two following bytes as in:

```
CLOSE1 LDX #$10   If entered here, we
       .BYTE $2C  effectively perform
CLOSE2 LDX #$20   a BIT test on $20A2,
       .BYTE $2C  another one on $30A2,
CLOSE3 LDX #$30   and end up with the X
CLOSEX LDA #12    register still at $10
       STA ICCOM,X upon arrival here.
```

Beware: a BIT instruction used in this way as a NOP does have effects: the flags may be modified, and the read of the absolute address, if it happens to access an I/O device, may cause an unwanted action.

Branch Instructions

Affect Flags: none

All branches are relative mode and have a length of two bytes. Syntax is "Bxx Displacement" or (better) "Bxx Label". See the notes on the Program Counter for more on displacements.

Branches are dependant on the status of the flag bits when the op code is encountered. A branch not taken requires two machine cycles. Add one if the branch is taken and add one more if the branch crosses a page boundary.

```
MNEMONIC                       HEX
BPL (Branch on PLus)           $10
BMI (Branch on MInus)          $30
BVC (Branch on oVerflow Clear) $50
BVS (Branch on oVerflow Set)   $70
BCC (Branch on Carry Clear)    $90
BCS (Branch on Carry Set)      $B0
BNE (Branch on Not Equal)      $D0
BEQ (Branch on EQual)          $F0
```

There is no BRA (BRanch Always) instruction but it can be easily emulated by branching on the basis of a known condition. One of the best flags to use for this purpose is the oVerflow which is unchanged by all but addition and subtraction operations.

A page boundary crossing occurs when the branch destination is on a different page than the instruction AFTER the branch instruction. For example:

```
  SEC
  BCS LABEL
  NOP
```

A page boundary crossing occurs (i.e. the BCS takes 4 cycles) when (the address of) LABEL and the NOP are on different pages. This means that

```
        CLV
        BVC LABEL
  LABEL NOP
```

the BVC instruction will take 3 cycles no matter what address it is located at.

BRK (BReaK)

Affects Flags: B

```
MODE           SYNTAX       HEX LEN TIM
Implied       BRK           $00  1   7
```

BRK causes a non-maskable interrupt and increments the program counter by one. Therefore an RTI will go to the address of the BRK +2 so that BRK may be used to replace a two-byte instruction for debugging and the subsequent RTI will be correct.

CMP (CoMPare accumulator)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Immediate     CMP #$44      $C9  2   2
Zero Page     CMP $44       $C5  2   3
Zero Page,X   CMP $44,X     $D5  2   4
```

```
Absolute      CMP $4400     $CD  3   4
Absolute,X    CMP $4400,X   $DD  3   4+
Absolute,Y    CMP $4400,Y   $D9  3   4+
Indirect,X    CMP ($44,X)   $C1  2   6
Indirect,Y    CMP ($44),Y   $D1  2   5+

+ add 1 cycle if page boundary crossed
```

Compare sets flags as if a subtraction had been carried out. If the value in the accumulator is equal or greater than the compared value, the Carry will be set. The equal (Z) and negative (N) flags will be set based on equality or lack thereof and the sign (i.e. A>=$80) of the accumulator.

CPX (ComPare X register)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Immediate     CPX #$44      $E0  2   2
Zero Page     CPX $44       $E4  2   3
Absolute      CPX $4400     $EC  3   4
```

Operation and flag results are identical to equivalent mode accumulator CMP ops.

CPY (ComPare Y register)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Immediate     CPY #$44      $C0  2   2
Zero Page     CPY $44       $C4  2   3
Absolute      CPY $4400     $CC  3   4
```

Operation and flag results are identical to equivalent mode accumulator CMP ops.

DEC (DECrement memory)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Zero Page     DEC $44       $C6  2   5
Zero Page,X   DEC $44,X     $D6  2   6
Absolute      DEC $4400     $CE  3   6
Absolute,X    DEC $4400,X   $DE  3   7
```

EOR (bitwise Exclusive OR)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     EOR #$44      $49  2   2
Zero Page     EOR $44       $45  2   3
Zero Page,X   EOR $44,X     $55  2   4
Absolute      EOR $4400     $4D  3   4
Absolute,X    EOR $4400,X   $5D  3   4+
Absolute,Y    EOR $4400,Y   $59  3   4+
Indirect,X    EOR ($44,X)   $41  2   6
Indirect,Y    EOR ($44),Y   $51  2   5+

+ add 1 cycle if page boundary crossed
```

Flag (Processor Status) Instructions

Affect Flags: as noted

These instructions are implied mode, have a length of one byte and require two machine cycles.

```
MNEMONIC                      HEX
CLC (CLear Carry)             $18
SEC (SEt Carry)               $38
CLI (CLear Interrupt)         $58
SEI (SEt Interrupt)           $78
CLV (CLear oVerflow)          $B8
CLD (CLear Decimal)           $D8
SED (SEt Decimal)             $F8
```

Notes:

The Interrupt flag is used to prevent (SEI) or enable (CLI) maskable interrupts (aka IRQ's). It does not signal the presence or absence of an interrupt condition. The 6502 will set this flag automatically in response to an interrupt and restore it to its prior status on completion of the interrupt service routine. If you want your interrupt service routine to permit other maskable interrupts, you must clear the I flag in your code.

The Decimal flag controls how the 6502 adds and subtracts. If set, arithmetic is carried out in packed binary coded decimal. This flag is unchanged by interrupts and is unknown on power-up. The implication is that a CLD should be included in boot or interrupt coding.

The Overflow flag is generally misunderstood and therefore under-utilised. After an ADC or SBC instruction, the overflow flag will be set if the twos complement result is less than -128 or greater than +127, and it will cleared otherwise. In twos complement, $80 through $FF represents -128 through -1, and $00 through $7F represents 0 through +127. Thus, after:

```
CLC
LDA #$7F ;   +127
ADC #$01 ; +   +1
```

the overflow flag is 1 (+127 + +1 = +128), and after:

```
CLC
LDA #$81 ;   -127
ADC #$FF ; +   -1
```

the overflow flag is 0 (-127 + -1 = -128). The overflow flag is not affected by increments, decrements, shifts and logical operations i.e. only ADC, BIT, CLV, PLP, RTI and SBC affect it. There is no op code to set the overflow

but a BIT test on an RTS instruction will do the trick.

INC (INCrement memory)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Zero Page     INC $44       $E6  2   5
Zero Page,X   INC $44,X     $F6  2   6
Absolute      INC $4400     $EE  3   6
Absolute,X    INC $4400,X   $FE  3   7
```

JMP (JuMP)

Affects Flags: none

```
MODE           SYNTAX       HEX LEN TIM
Absolute      JMP $5597     $4C  3   3
Indirect      JMP ($5597)   $6C  3   5
```

JMP transfers program execution to the following address (absolute) or to the location contained in the following address (indirect). Note that there is no carry associated with the indirect jump so:

AN INDIRECT JUMP MUST NEVER USE A VECTOR BEGINNING ON THE LAST BYTE OF A PAGE

For example if address $3000 contains $40, $30FF contains $80, and $3100 contains $50, the result of JMP ($30FF) will be a transfer of control to $4080 rather than $5080 as you intended i.e. the 6502 took the low byte of the address from $30FF and the high byte from $3000.

JSR (Jump to SubRoutine)

Affects Flags: none

```
MODE           SYNTAX       HEX LEN TIM
Absolute      JSR $5597     $20  3   6
```

JSR pushes the address-1 of the next operation on to the stack before transferring program control to the following address. Subroutines are normally terminated by a RTS op code.

LDA (LoaD Accumulator)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     LDA #$44      $A9  2   2
Zero Page     LDA $44       $A5  2   3
Zero Page,X   LDA $44,X     $B5  2   4
Absolute      LDA $4400     $AD  3   4
Absolute,X    LDA $4400,X   $BD  3   4+
Absolute,Y    LDA $4400,Y   $B9  3   4+
Indirect,X    LDA ($44,X)   $A1  2   6
Indirect,Y    LDA ($44),Y   $B1  2   5+

+ add 1 cycle if page boundary crossed
```

LDX (LoaD X register)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     LDX #$44      $A2  2   2
Zero Page     LDX $44       $A6  2   3
Zero Page,Y   LDX $44,Y     $B6  2   4
Absolute      LDX $4400     $AE  3   4
Absolute,Y    LDX $4400,Y   $BE  3   4+

+ add 1 cycle if page boundary crossed
```

LDY (LoaD Y register)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     LDY #$44      $A0  2   2
Zero Page     LDY $44       $A4  2   3
Zero Page,X   LDY $44,X     $B4  2   4
Absolute      LDY $4400     $AC  3   4
Absolute,X    LDY $4400,X   $BC  3   4+

+ add 1 cycle if page boundary crossed
```

LSR (Logical Shift Right)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Accumulator   LSR A         $4A  1   2
Zero Page     LSR $44       $46  2   5
Zero Page,X   LSR $44,X     $56  2   6
Absolute      LSR $4400     $4E  3   6
Absolute,X    LSR $4400,X   $5E  3   7
```

LSR shifts all bits right one position. 0 is shifted into bit 7 and the original bit 0 is shifted into the Carry.

Wrap-Around

Use caution with indexed zero page operations as they are subject to wrap-around. For example, if the X register holds $FF and you execute LDA $80,X you will not access $017F as you might expect; instead you access $7F i.e. $80-1. This characteristic can be used to advantage but make sure your code is well commented.

It is possible, however, to access $017F when X = $FF by using the Absolute,X addressing mode of LDA $80,X. That is, instead of:

```
LDA $80,X    ; ZeroPage,X - the resulting object code is: B5 80
```

which accesses $007F when X=$FF, use:

```
LDA $0080,X  ; Absolute,X - the resulting object code is: BD 80 00
```

which accesses $017F when X = $FF (a at cost of one additional byte and one additional cycle). All of the ZeroPage,X and ZeroPage,Y instructions except STX ZeroPage,Y and STY ZeroPage,X have a corresponding Absolute,X and Absolute,Y instruction. Unfortunately, a lot of 6502 assemblers don't have an easy way to force Absolute addressing, i.e. most will assemble a LDA $0080,X as B5 80. One way to overcome this is to insert the bytes using the .BYTE pseudo-op (on some 6502 assemblers this pseudo-op is called DB or DFB, consult the assembler documentation) as follows:

```
.BYTE $BD,$80,$00  ; LDA $0080,X (absolute,X addressing mode)
```

The comment is optional, but highly recommended for clarity.

In cases where you are writing code that will be relocated you must consider wrap-around when assigning dummy values for addresses that will be adjusted. Both zero and the semi-standard $FFFF should be avoided for dummy labels. The use of zero or zero page values will result in assembled code with zero page opcodes when you wanted absolute codes. With $FFFF, the problem is in addresses+1 as you wrap around to page 0.

Program Counter

When the 6502 is ready for the next instruction it increments the program counter before fetching the instruction. Once it has the op code, it increments the program counter by the length of the operand, if any. This must be accounted for when calculating branches or when pushing bytes to create a false return address (i.e. jump table addresses are made up of addresses-1 when it is intended to use an RTS rather than a JMP).

The program counter is loaded least signifigant byte first. Therefore the most signifigant byte must be pushed first when creating a false return address.

When calculating branches a forward branch of 6 skips the following 6 bytes so, effectively the program counter points to the address that is 8 bytes beyond the address of the branch opcode; and a backward branch of $FA (256-6) goes to an address 4 bytes before the branch instruction.

Execution Times

Op code execution times are measured in machine cycles; one machine cycle equals one clock cycle. Many instructions require one extra cycle for execution if a page boundary is crossed; these are indicated by a + following the time values shown.

NOP (No OPeration)

Affects Flags: none

```
MODE           SYNTAX       HEX LEN TIM
Implied       NOP           $EA  1   2
```

NOP is used to reserve space for future modifications or effectively REM out existing code.

ORA (bitwise OR with Accumulator)

Affects Flags: N Z

```
MODE           SYNTAX       HEX LEN TIM
Immediate     ORA #$44      $09  2   2
Zero Page     ORA $44       $05  2   3
Zero Page,X   ORA $44,X     $15  2   4
Absolute      ORA $4400     $0D  3   4
Absolute,X    ORA $4400,X   $1D  3   4+
Absolute,Y    ORA $4400,Y   $19  3   4+
Indirect,X    ORA ($44,X)   $01  2   6
Indirect,Y    ORA ($44),Y   $11  2   5+

+ add 1 cycle if page boundary crossed
```

Register Instructions

Affect Flags: N Z

These instructions are implied mode, have a length of one byte and require two machine cycles.

```
MNEMONIC                 HEX
TAX (Transfer A to X)    $AA
TXA (Transfer X to A)    $8A
DEX (DEcrement X)        $CA
INX (INcrement X)        $E8
TAY (Transfer A to Y)    $A8
TYA (Transfer Y to A)    $98
DEY (DEcrement Y)        $88
INY (INcrement Y)        $C8
```

ROL (ROtate Left)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Accumulator   ROL A         $2A  1   2
Zero Page     ROL $44       $26  2   5
```

```
Zero Page,X   ROL $44,X     $36  2   6
Absolute      ROL $4400     $2E  3   6
Absolute,X    ROL $4400,X   $3E  3   7
```

ROL shifts all bits left one position. The Carry is shifted into bit 0 and the original bit 7 is shifted into the Carry.

ROR (ROtate Right)

Affects Flags: N Z C

```
MODE           SYNTAX       HEX LEN TIM
Accumulator   ROR A         $6A  1   2
Zero Page     ROR $44       $66  2   5
Zero Page,X   ROR $44,X     $76  2   6
Absolute      ROR $4400     $6E  3   6
Absolute,X    ROR $4400,X   $7E  3   7
```

ROR shifts all bits right one position. The Carry is shifted into bit 7 and the original bit 0 is shifted into the Carry.

RTI (ReTurn from Interrupt)

Affects Flags: all

```
MODE           SYNTAX       HEX LEN TIM
Implied       RTI           $40  1   6
```

RTI retrieves the Processor Status Word (flags) and the Program Counter from the stack in that order (interrupts push the PC first and then the PSW).

Note that unlike RTS, the return address on the stack is the actual address rather than the address-1.

RTS (ReTurn from Subroutine)

Affects Flags: none

```
MODE           SYNTAX       HEX LEN TIM
Implied       RTS           $60  1   6
```

RTS pulls the top two bytes off the stack (low byte first) and transfers program control to that address+1. It is used, as expected, to exit a subroutine invoked via JSR which pushed the address-1.

RTS is frequently used to implement a jump table where addresses-1 are pushed onto the stack and accessed via RTS eg. to access the second of four routines:

```
 LDX #1
 JSR EXEC
```

```
  JMP SOMEWHERE

LOBYTE
 .BYTE <ROUTINE0-1,<ROUTINE1-1
 .BYTE <ROUTINE2-1,<ROUTINE3-1

HIBYTE
 .BYTE >ROUTINE0-1,>ROUTINE1-1
 .BYTE >ROUTINE2-1,>ROUTINE3-1

EXEC
 LDA HIBYTE,X
 PHA
 LDA LOBYTE,X
 PHA
 RTS
```

SBC (SuBtract with Carry)

Affects Flags: N V Z C

```
MODE           SYNTAX       HEX LEN TIM
Immediate     SBC #$44      $E9  2   2
Zero Page     SBC $44       $E5  2   3
Zero Page,X   SBC $44,X     $F5  2   4
Absolute      SBC $4400     $ED  3   4
Absolute,X    SBC $4400,X   $FD  3   4+
Absolute,Y    SBC $4400,Y   $F9  3   4+
Indirect,X    SBC ($44,X)   $E1  2   6
Indirect,Y    SBC ($44),Y   $F1  2   5+

+ add 1 cycle if page boundary crossed
```

SBC results are dependant on the setting of the decimal flag. In decimal mode, subtraction is carried out on the assumption that the values involved are packed BCD (Binary Coded Decimal).

There is no way to subtract without the carry which works as an inverse borrow. i.e, to subtract you set the carry before the operation. If the carry is cleared by the operation, it indicates a borrow occurred.

STA (STore Accumulator)

Affects Flags: none

```
MODE           SYNTAX       HEX LEN TIM
Zero Page     STA $44       $85  2   3
Zero Page,X   STA $44,X     $95  2   4
Absolute      STA $4400     $8D  3   4
Absolute,X    STA $4400,X   $9D  3   5
Absolute,Y    STA $4400,Y   $99  3   5
Indirect,X    STA ($44,X)   $81  2   6
Indirect,Y    STA ($44),Y   $91  2   6
```

Stack Instructions

These instructions are implied mode, have a length of one byte and require machine cycles as indicated. The "PuLl" operations are known as "POP" on most other microprocessors. With the 6502, the stack is always on page one ($100-$1FF) and works top down.

MNEMONIC	HEX	TIM
TXS (Transfer X to Stack ptr)	$9A	2
TSX (Transfer Stack ptr to X)	$BA	2
PHA (PusH Accumulator)	$48	3
PLA (PuLl Accumulator)	$68	4
PHP (PusH Processor status)	$08	3
PLP (PuLl Processor status)	$28	4

STX (STore X register)

Affects Flags: none

MODE	SYNTAX	HEX	LEN	TIM
Zero Page	STX $44	$86	2	3
Zero Page,Y	STX $44,Y	$96	2	4
Absolute	STX $4400	$8E	3	4

STY (STore Y register)

Affects Flags: none

MODE	SYNTAX	HEX	LEN	TIM
Zero Page	STY $44	$84	2	3
Zero Page,X	STY $44,X	$94	2	4
Absolute	STY $4400	$8C	3	4

Last Updated Oct 17, 2020.