



OpenCores.Org

# R65C02 IP Core Specification

*Author: Jens Gutschmidt  
scantara2003@yahoo.de*

**Rev. 0.7**  
**February 25, 2009**



*This page has been intentionally left blank.*

## Revision History

| Rev. | Date     | Author          | Description  |
|------|----------|-----------------|--|
| 0.1  | 12/18/06 | Jens Gutschmidt | First Draft  |
| 0.2  | 01/02/07 | Jens Gutschmidt | Pictures of FSM's  |
| 0.3  | 08/20/08 | Jens Gutschmidt | <ul style="list-style-type: none"><li>- Tables and timing diagrams</li><li>- Deleting pictures of FSM</li></ul>              |
| 0.4  | 10/01/08 | Jens Gutschmidt | <ul style="list-style-type: none"><li>- New ideas for timing diagrams</li></ul>  |
| 0.5  | 01/02/09 | Jens Gutschmidt | <ul style="list-style-type: none"><li>- Textual changes / spell checking</li><li>- Insert R6502 TC block diagram</li></ul>   |
| 0.6  | 02/01/09 | Jens Gutschmidt | <ul style="list-style-type: none"><li>- Work on Timing Diagrams</li></ul>  |
| 0.7  | 23/02/09 | Jens Gutschmidt | <ul style="list-style-type: none"><li>- Added new instruction codes (Timing diagrams for all new op codes missing)</li></ul> |

# Contents

ERROR! NO TABLE OF CONTENTS ENTRIES FOUND.

|          |                           |           |
|----------|---------------------------|-----------|
| <b>1</b> | <b>INTRODUCTION .....</b> | <b>1</b>  |
| <b>2</b> | <b>ARCHITECTURE .....</b> | <b>2</b>  |
| <b>3</b> | <b>OPERATION .....</b>    | <b>6</b>  |
|          | <i>ADC.....</i>           | <i>7</i>  |
|          | <i>AND.....</i>           | <i>8</i>  |
|          | <i>ASL.....</i>           | <i>9</i>  |
|          | <i>BBR.....</i>           | <i>9</i>  |
|          | <i>BBS.....</i>           | <i>10</i> |
|          | <i>BCC.....</i>           | <i>11</i> |
|          | <i>BCS.....</i>           | <i>12</i> |
|          | <i>BEQ.....</i>           | <i>13</i> |
|          | <i>BIT.....</i>           | <i>14</i> |
|          | <i>BMI.....</i>           | <i>15</i> |
|          | <i>BNE.....</i>           | <i>16</i> |
|          | <i>BPL.....</i>           | <i>16</i> |
|          | <i>BRA.....</i>           | <i>16</i> |
|          | <i>BRK.....</i>           | <i>16</i> |
|          | <i>BVC.....</i>           | <i>16</i> |
|          | <i>BVS.....</i>           | <i>16</i> |
|          | <i>CLC.....</i>           | <i>16</i> |
|          | <i>CLD.....</i>           | <i>16</i> |
|          | <i>CLI.....</i>           | <i>16</i> |
|          | <i>CLV.....</i>           | <i>16</i> |
|          | <i>CMP.....</i>           | <i>16</i> |
|          | <i>CPX.....</i>           | <i>16</i> |
|          | <i>CPY.....</i>           | <i>16</i> |
|          | <i>DEC.....</i>           | <i>16</i> |
|          | <i>DEX.....</i>           | <i>16</i> |
|          | <i>DEY.....</i>           | <i>16</i> |
|          | <i>EOR.....</i>           | <i>16</i> |
|          | <i>INC.....</i>           | <i>16</i> |
|          | <i>INX.....</i>           | <i>16</i> |
|          | <i>INY.....</i>           | <i>16</i> |
|          | <i>JMP.....</i>           | <i>16</i> |
|          | <i>JSR.....</i>           | <i>16</i> |
|          | <i>LDA.....</i>           | <i>16</i> |
|          | <i>LDX.....</i>           | <i>16</i> |
|          | <i>LDY.....</i>           | <i>16</i> |
|          | <i>LSR.....</i>           | <i>16</i> |
|          | <i>NOP.....</i>           | <i>16</i> |
|          | <i>ORA.....</i>           | <i>16</i> |
|          | <i>PHA.....</i>           | <i>16</i> |
|          | <i>PHP.....</i>           | <i>16</i> |
|          | <i>PHX.....</i>           | <i>16</i> |
|          | <i>PHY.....</i>           | <i>16</i> |

---

|   |           |
|---|-----------|
| PLA.....  | 16        |
| PLP.....  | 16        |
| PLX.....  | 16        |
| PLY.....  | 16        |
| ROL.....  | 16        |
| ROR.....  | 16        |
| RMB.....  | 16        |
| RTI.....  | 16        |
| RTS.....  | 16        |
| SBC.....  | 16        |
| SEC.....  | 16        |
| SED.....  | 16        |
| SEI.....  | 16        |
| SMB.....  | 16        |
| STA.....  | 16        |
| STX.....  | 16        |
| STY.....  | 16        |
| STZ.....  | 16        |
| TAX.....  | 16        |
| TAY.....  | 16        |
| TRB.....  | 16        |
| TSB.....  | 16        |
| TSX.....  | 16        |
| TXA.....  | 16        |
| TXS.....  | 16        |
| TYA.....  | 16        |
| <b>4.....</b>   | <b>16</b> |
| Registers.....  | 16        |
| List of Registers.....  | 16        |
| Register 1 – Description.....   | 16        |
| <b>5.....</b>   | <b>16</b> |
| Clocks.....   | 16        |
| <b>APPENDIX A.....</b>  | <b>16</b> |
| <b>TIMING DIAGRAMS.....</b>   | <b>16</b> |
| ADC, SBC.....   | 16        |
| AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA.....                               | 16        |
| BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE.....   | 16        |
| BRK.....  | 16        |
| ASL, LSR, ROL, ROR, DEC, INC.....   | 16        |
| CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, SEC, SED, SEI, TAX, TAY, TSX, TXA, TYA..... | 16        |
| JMP.....  | 16        |
| JSR.....  | 16        |
| NOP, TXS.....   | 16        |
| PLA, PLP.....   | 16        |
| PHA, PHP.....   | 16        |
| RTI.....  | 16        |
| RTS.....  | 16        |
| STA, STX, STY.....  | 16        |
| <b>INSTRUCTION TABLE.....</b>   | <b>16</b> |
| <b>INDEX.....</b>   | <b>16</b> |
| <b>LIST OF FIGURES</b>  |           |

|  |    |
|--|----|
| Figure (1): 6502 architecture .....  | 2  |
| Figure (2): R6502_TC IP core architecture .....  | 5  |
| Figure (3): ADC, SBC – Timing Diagram “Immediate” .....  | 16 |
| Figure (4): ADC, SBC – Timing Diagram “Zero Page” .....  | 16 |
| Figure (5): ADC, SBC – Timing Diagram “Zero Page, X” .....   | 16 |
| Figure (6): ADC, SBC – Timing Diagram “Absolute” .....   | 16 |
| Figure (7): ADC, SBC – Timing Diagram “Absolute, X” – no page crossing .....                           | 16 |
| Figure (8): ADC, SBC – Timing Diagram “Absolute, X” – page crossing .....                              | 16 |
| Figure (9): ADC, SBC – Timing Diagram “Absolute, Y” – no page crossing .....                           | 16 |
| Figure (10): ADC, SBC – Timing Diagram “Absolute, Y” – page crossing .....                             | 16 |
| Figure (11): ADC, SBC – Timing Diagram “(Indirect, X)” .....   | 16 |
| Figure (12): ADC, SBC – Timing Diagram “(Indirect), Y” – no page crossing .....                        | 16 |
| Figure (13): ADC, SBC – Timing Diagram “(Indirect), Y” – page crossing .....                           | 16 |
| Figure (14): AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Immediate” .....            | 16 |
| Figure (15): AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Zero Page” .....       | 16 |
| Figure (16): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Zero Page, X” .....                        | 16 |
| Figure (17): AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Absolute” .....        | 16 |
| Figure (18): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Absolute, X” – no page crossing .....      | 16 |
| Figure (19): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Absolute, X” – page crossing .....         | 16 |
| Figure (20): AND, CMP, EOR, LDA, LDX, ORA – Timing Diagram “Absolute, Y” – no page crossing .....      | 16 |
| Figure (21): AND, CMP, EOR, LDA, LDX, ORA – Timing Diagram “Absolute, Y” – page crossing .....         | 16 |
| Figure (22): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect, X)” .....                            | 16 |
| Figure (23): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect), Y” – no page crossing .....         | 16 |
| Figure (24): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect), Y” – page crossing .....            | 16 |
| Figure (25): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “no branch, same page” .....      | 16 |
| Figure (26): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “no branch, different page” ..... | 16 |
| Figure (27): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “branch, same page” .....         | 16 |

|  |    |
|--|----|
| Figure (28): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “branch, different page” .....                | 16 |
| Figure (29): BRK – Timing Diagram “Implied” .....  | 16 |
| Figure (30): ASL A, LSR A, ROL A, ROR A – Timing Diagram “Immediate” .....   | 16 |
| Figure (31): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Zero Page” .....                                       | 16 |
| Figure (32): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Zero Page, X” .....                                    | 16 |
| Figure (33): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Absolute” .....  | 16 |
| Figure (34): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Absolute, X” .....                                     | 16 |
| Figure (35): CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, SEC, SED, SEI, TAX, TAY, TSX, TXA, TYA – Timing Diagram ..... | 16 |
| Figure (36): JMP – Timing Diagram “Absolute” .....   | 16 |
| Figure (37): JMP – Timing Diagram “Indirect” .....   | 16 |
| Figure (38): JSR – Timing Diagram .....  | 16 |
| Figure (39): NOP, TXS – Timing Diagram .....   | 16 |
| Figure (40): PLA, PLP – Timing Diagram “Implied” .....   | 16 |
| Figure (41): PHA, PHP – Timing Diagram “Implied” .....   | 16 |
| Figure (42): RTI – Timing Diagram “Implied” .....  | 16 |
| Figure (43): RTS – Timing Diagram “Implied” .....  | 16 |
| Figure (44): STA, STX, STY – Timing Diagram “Zero Page” .....  | 16 |
| Figure (45): STA, STY – Timing Diagram “Zero Page, X” .....  | 16 |
| Figure (46): STX – Timing Diagram “Zero Page, Y” .....   | 16 |
| Figure (47): STA, STX, STY – Timing Diagram “Absolute” .....   | 16 |
| Figure (48): STA – Timing Diagram “Absolute, X” .....  | 16 |
| Figure (49): STA – Timing Diagram “Absolute, Y” .....  | 16 |
| Figure (50): STA – Timing Diagram “(Indirect, X)” .....  | 16 |
| Figure (51): STA – Timing Diagram “(Indirect), Y” .....  | 16 |

## LIST OF TABLES

|  |   |
|--|---|
| Table (1): ADC – Short Reference ..... | 7 |
| Table (2): AND – Short Reference ..... | 8 |
| Table (3): ASL – Short Reference ..... | 9 |
| Table (4): BBR – Short Reference ..... | 9 |



|   |    |
|---|----|
| Table (5): BBS – Short Reference .....  | 10 |
| Table (6): BCC – Short Reference.....   | 11 |
| Table (7): BCS – Short Reference .....  | 12 |
| Table (8): BEQ – Short Reference.....   | 13 |
| Table (9): BIT – Short Reference .....  | 14 |
| Table (10): BMI – Short Reference ..... | 15 |
| Table (11): BNE – Short Reference ..... | 16 |
| Table (12): BPL – Short Reference ..... | 16 |
| Table (13): BRA – Short Reference ..... | 16 |
| Table (14): BRK – Short Reference ..... | 16 |
| Table (15): BVC – Short Reference ..... | 16 |
| Table (16): BVS – Short Reference.....  | 16 |
| Table (17): CLC – Short Reference.....  | 16 |
| Table (18): CLD – Short Reference ..... | 16 |
| Table (19): CLI – Short Reference ..... | 16 |
| Table (20): CLV – Short Reference ..... | 16 |
| Table (21): CMP – Short Reference ..... | 16 |
| Table (22): CPX – Short Reference.....  | 16 |
| Table (23): CPY – Short Reference.....  | 16 |
| Table (24): DEC – Short Reference ..... | 16 |
| Table (25): DEX – Short Reference ..... | 16 |
| Table (26): DEX – Short Reference ..... | 16 |
| Table (27): EOR – Short Reference ..... | 16 |
| Table (28): INC – Short Reference.....  | 16 |
| Table (29): INX – Short Reference ..... | 16 |
| Table (30): INY – Short Reference ..... | 16 |
| Table (31): JMP – Short Reference ..... | 16 |
| Table (32): JSR – Short Reference ..... | 16 |
| Table (33): LDA – Short Reference ..... | 16 |
| Table (34): LDX – Short Reference ..... | 16 |
| Table (35): LDY – Short Reference ..... | 16 |
| Table (36): LSR – Short Reference ..... | 16 |
| Table (37): NOP – Short Reference ..... | 16 |

---

|   |    |
|---|----|
| Table (38): OR – Short Reference .....  | 16 |
| Table (39): PHA – Short Reference ..... | 16 |
| Table (40): PHP – Short Reference ..... | 16 |
| Table (41): PHX – Short Reference ..... | 16 |
| Table (42): PHY – Short Reference ..... | 16 |
| Table (43): PLA – Short Reference ..... | 16 |
| Table (44): PLP – Short Reference ..... | 16 |
| Table (45): PLX – Short Reference ..... | 16 |
| Table (46): PLY – Short Reference ..... | 16 |
| Table (47): ROL – Short Reference ..... | 16 |
| Table (48): ROR – Short Reference ..... | 16 |
| Table (49): RMB – Short Reference ..... | 16 |
| Table (50): RTI – Short Reference ..... | 16 |
| Table (51): RTS – Short Reference ..... | 16 |
| Table (52): SBC – Short Reference ..... | 16 |
| Table (53): SEC – Short Reference ..... | 16 |
| Table (54): CLD – Short Reference ..... | 16 |
| Table (55): SEI – Short Reference ..... | 16 |
| Table (56): SMB – Short Reference ..... | 16 |
| Table (57): STA – Short Reference ..... | 16 |
| Table (58): STX – Short Reference ..... | 16 |
| Table (59): STY – Short Reference ..... | 16 |
| Table (60): STZ – Short Reference ..... | 16 |
| Table (61): TAX – Short Reference ..... | 16 |
| Table (62): TAY – Short Reference ..... | 16 |
| Table (63): TRB – Short Reference ..... | 16 |
| Table (64): TSB – Short Reference ..... | 16 |
| Table (65): TSX – Short Reference ..... | 16 |
| Table (66): TXA – Short Reference ..... | 16 |
| Table (67): TXS – Short Reference ..... | 16 |
| Table (68): TYA – Short Reference ..... | 16 |

---



# 1 Introduction

---

The Central Processing Unit (CPU) 6502 was introduced at 1976 by Commodore Computers. It is an 8 Bit processor which is well known worldwide. The also most known computer system in the 70s/80s was the APPLE based on this famous 6502.

In this century building of little computer systems based on an 8 Bit architecture is easier than ever before. Many CAD/CAM tool are existing on the market to give you help to do this job.

In the last decade the technology give us more and more possibilities to reach our dreams – higher, faster, wider. FPGA's (Field Programmable Gate Arrays) and CPLD's (Complex Programmable Logic Device)s are coming up. They shortening the time of development, simulation and verification of such systems dramatically. On the other hand stands the rising complexity of designs and well knowing of desired description languages – VHDL (Very high scale integration Hardware Description Language) or Verilog.

Many operations and functions of computer tasks aren't need really the power of “modern” processors today like Intel's Pentium. Nevertheless it is not usable – some times also impossible - to build simple systems with discrete IC's. The necessity of flexibility compel us the using of high tech parts and tools...to build a “simple” system.

The using of standard IP's (Intellectual Properties) is the key to reach that goal.

Now talking about the specification of the **6502 True Cycle Core IP**. Written in VHDL but developed with Mentor's HDL Designer. The 6502 is very easy to handle for people were have experience with other cpu's. Also suitable for beginners who will learning to build her own first cpu system.

# 2 Architecture

The following figure shows the internal architecture of the Commodore's 6502.

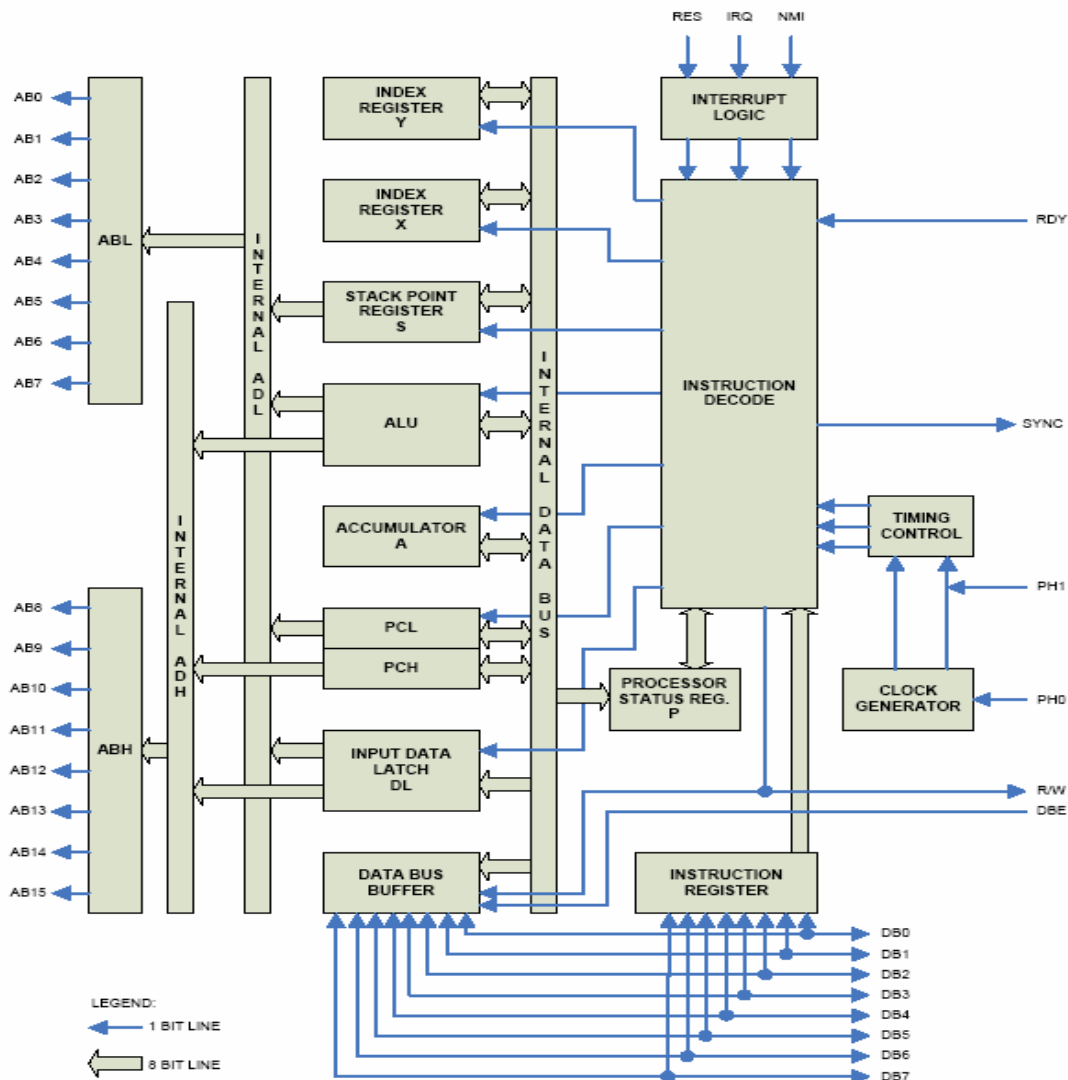


Figure (1): 6502 architecture

First, for more clearance and better understanding of the **6502 True Cycle Core IP** let me summarize the most important quality of Commodore's 6502.

This architecture based on asynch logic. The two phase clock is a special attribute of that. The Data Bus is only active while clock phase PH1 is "1" for reading and writing (PH0 is "0"). All transfers of data are occur at active phase of PH1.

The whole 6502 is fully statically, so all registers holding her values if the clocking will be going to D.C. Internally all operation of 16 Bit are splitted into two 8 Bit busses. This is very interesting and causes one more clock cycle in some operations (see “PCL”, “PCH” -> Program Counter Low/High) which can operate over page boundaries.

Address lines are 16 Bit wide. 65.536 addresses for byte access are possible. There are no differences exist between memory and I/O cycles. All I/O devices must be connected via memory mapped I/O. The mnemonics are also all the same for both. The usage of address space is normally organized form top to the button “ROM – I/O – RAM” (0xFFFF – 0x0000).

Only one pin  $\overline{R\overline{W}}$  controls the read and write operations.

The 6502 can hold in every cycle – except write cycles - by applying RDY. So wait states can be generate by using RDY.

There are two interrupts  $\overline{IRQ}$  and  $\overline{NMI}$ .  $\overline{IRQ}$  is mask able,  $\overline{NMI}$  is not. Note that  $\overline{RES}$  is internally handled as an interrupt, but is resetting the 6502 to the starting point of operation at vector address 0xFFFF. Every interrupt has its own vector which can point elsewhere to the 16 Bit address space. The vectors are located from 0xFFFF to 0xFFFA.

The Stack Pointer is 256 Bytes deep and is hard-wired to 0x01FF (internally 0x0200 –0x01FF going to the address bus) for the first memory location and grows downward to 0x0100. Decrementing again will produce a wrap around back to 0x01FF.

Arithmetical operations like ADC and SBC can handle binary values from 0-255 and decimal values from 0-99 without using other op codes. Only one user changeable bit into the Status Register determine the exact arithmetic mode of that operations.

The 6502 generally operates into a pipeline mode. That means that a finish of an OP code falls every time into a fetch cycle of the next OP. This save one clock cycle

Every operation consume between two and seven clock cycles.

To build an useful und backward software and hardware compatible VHDL Core for the 6502 many unavoidable changes may be forced to simulation and verification before any real core will be made.

The requirements:

- Vendor indepent for implementation in any FPGA
- True Cycle for all operations as described in original publications
- No “Mixed Mode” – Only 6502, not 65C02 -> for performance and area
- No fantastic or useful extensions – as like the original 6502 as possible
- Don’t implement the “undocumented OP’s” at this time -> goodie for future releases
- Only one clock
- Full synch design
- Operating speed up to 60 MHz – or higher
- Easily to change for future requirement – building variants
- The activity on data and address busses since the execution of OP codes is not forced to be like the original – may be or may be not.

The design based on finite state machines (fsm). Every OP code has its own fsm. Some registers are for internally use only and stores temporally values. These registers are not accessible by the programmer.

Some fsm’s own registers for internally use and don’t share this registers with other fsm’s. This help to decrease fan-in and increase performance.

The next picture shows the hierarchical structure of the R6502 IP core.

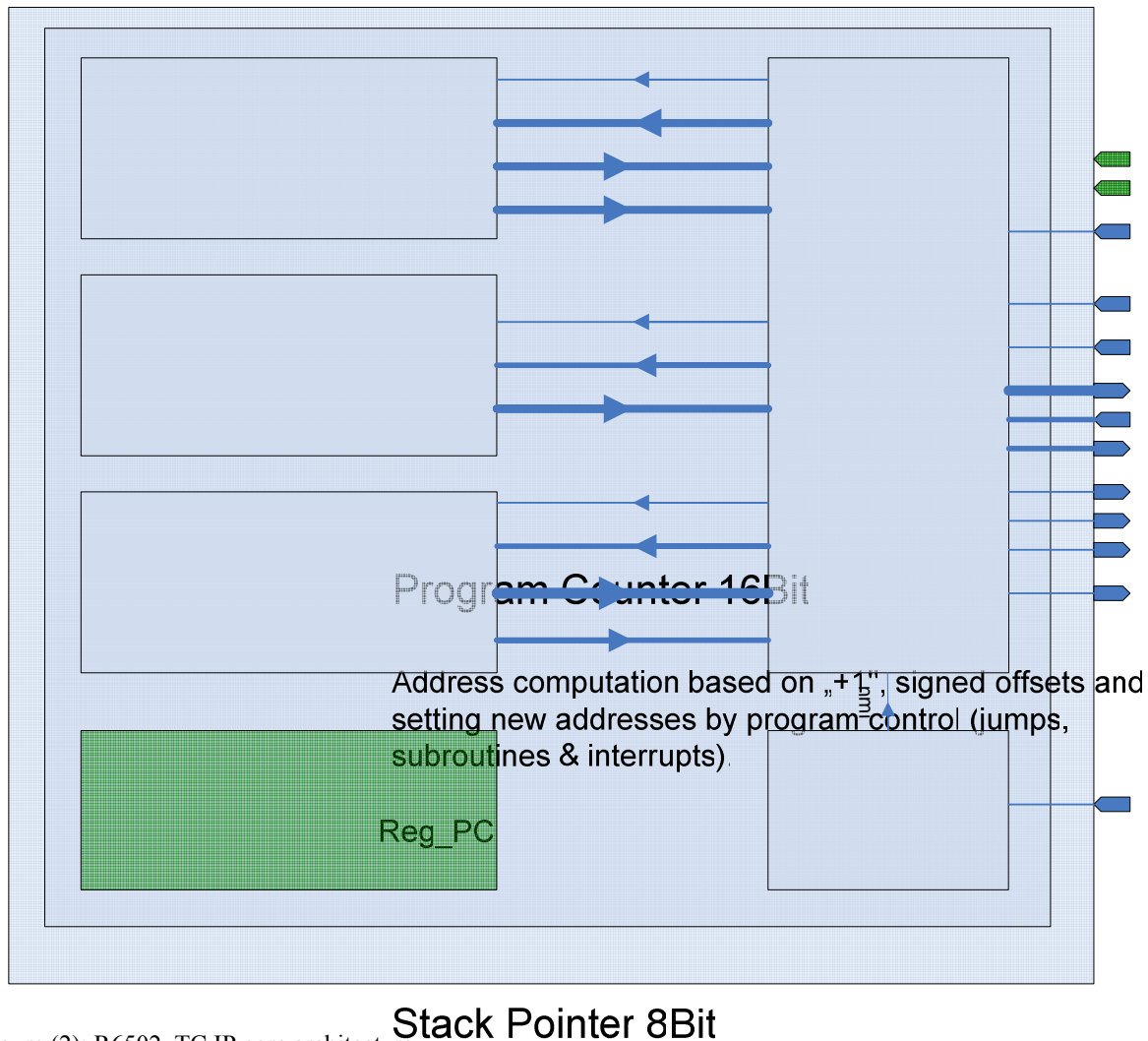


Figure (2): R6502\_TC IP core architecture

Address computation based on „-1“, „+1“ and setting new addresses by program control.  
Fixed to page 1 (0x01FF down to 0x0100).

Reg\_SP

## Registerbank for A, X and Y

Flipflops for main registers with internal paths for register-to-register transfers.

Rev 0.7 Preliminary

5 of 16

RegBank\_AXY



# 3 Operation

---

## ADC

Operation: Add memory or immediate value to accumulator A with carry ( $A + M + C \rightarrow A, C$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | ADC # Oper             | 69      | 2         | 2          | √ | √ | - | - | - | - | √ | √ |
| Zero Page             | ADC Oper               | 65      | 2         | 3          | √ | √ | - | - | - | - | √ | √ |
| Zero Page, X          | ADC Oper, X            | 75      | 2         | 4          | √ | √ | - | - | - | - | √ | √ |
| Absolute              | ADC Oper               | 6D      | 3         | 4          | √ | √ | - | - | - | - | √ | √ |
| Absolute, X           | ADC Oper, X            | 7D      | 3         | 4*         | √ | √ | - | - | - | - | √ | √ |
| Absolute, Y           | ADC Oper, Y            | 79      | 3         | 4*         | √ | √ | - | - | - | - | √ | √ |
| (Indirect, X)         | ADC (Oper, X)          | 61      | 2         | 6          | √ | √ | - | - | - | - | √ | √ |
| (Indirect), Y         | ADC (Oper), Y          | 71      | 2         | 5*         | √ | √ | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | ADC (Oper)             | 72      | 2         | 5          | √ | √ | - | - | - | - | √ | √ |

\* => Add 1 if page boundary is crossed

Table (1): ADC – Short Reference

Example Immediate (assumed A=\$FE, C=0):

| Address | Bytes | Mnemonic  |                                     |
|---------|-------|-----------|-------------------------------------|
| \$9000  | 69 03 | ADC #\$03 | ;                                   |
| \$9002  | ...   | next OP   | ; A is now \$01, C=1, N=0, Z=0, V=0 |

1111 1110 (\$FE)  
 ADC 0000 0011 (\$03)  
 0000 0001 (\$01), C=1, N=0, Z=0, V=0

## AND

Operation: “AND” memory or immediate value with accumulator A ( $A \cap M \rightarrow A$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | AND #Oper              | 29      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page             | AND Oper               | 25      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, X          | AND Oper, X            | 35      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute              | AND Oper               | 2D      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, X           | AND Oper, X            | 3D      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| Absolute, Y           | AND Oper, Y            | 39      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| (Indirect, X)         | AND (Oper, X)          | 21      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| (Indirect), Y         | AND (Oper), Y          | 31      | 2         | 5*         | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | AND (Oper)             | 32      | 2         | 5          | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (2): AND – Short Reference

Example Immediate (assumed A is \$C5):

| Address | Bytes | Mnemonic  |                           |
|---------|-------|-----------|---------------------------|
| \$9000  | 29 71 | AND #\$71 | ;                         |
| \$9002  | ...   | next OP   | ; A is now \$41, N=0, Z=0 |

```

1100 0101 ($C5)
AND 0111 0001 ($71)
0100 0001 ($41), N=0, Z=0

```

## ASL

Operation: Shift Left One Bit (Memory or Accumulator) ( $C \leftarrow \boxed{7} \boxed{6} \boxed{5} \boxed{4} \boxed{3} \boxed{2} \boxed{1} \boxed{0} \leftarrow 0$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator     | ASL A                  | 0A      | 1         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page       | ASL Oper               | 06      | 2         | 5          | √ | - | - | - | - | - | √ | √ |
| Zero Page, X    | ASL Oper, X            | 16      | 2         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute        | ASL Oper               | 0E      | 3         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute, X     | ASL Oper, X            | 1E      | 3         | 7          | √ | - | - | - | - | - | √ | √ |

Table (3): ASL – Short Reference

Example Accumulator (assumed A=\$55):

| Address | Bytes | Mnemonic |                           |
|---------|-------|----------|---------------------------|
| \$9000  | 0A    | ASL A    | ;                         |
| \$9001  | ...   | next OP  | ; A is now \$AA, N=1, Z=0 |

ASL 0101 0101 (\$55)  
 1010 1010 (\$AA), C=0, N=1, Z=0

## BBR

Operation: Branch on memory bit x is reset, where x=0-7 MSB of op code (Branch on Mx = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BBR(0-7) ZP,offs8      | 0F-7F   | 3         | 5*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (4): BBR – Short Reference

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes    | Mnemonic        |   |
|---------|----------|-----------------|---|
| \$9000  | 2F 32 10 | BBR2 \$32, \$10 | ; BBR to same page, - NO BRANCH - jumps to \$9003 |
| \$9003  | ...      | next OP         | ; M2(\$0032)=1 => no branch, 5 cycles             |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes    | Mnemonic        |  |
|---------|----------|-----------------|--|
| \$90FF  | 5F 32 10 | BBR5 \$32, \$10 | ; BBR to different page, - NO BRANCH – jumps to \$9102 |
| \$9102  | ...      | <i>next OP</i>  | ; M5(\$0032)=1 => no branch, 5 cycles (!!!)            |

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes    | Mnemonic        |   |
|---------|----------|-----------------|---|
| \$9000  | 0F 32 10 | BBR0 \$32, \$10 | ; BBR to same page, - BRANCH - jumps to \$9003 + \$10 |
| ...     |          |                 |   |
| \$9013  | ...      | <i>next OP</i>  | ; M0(\$0032)=0 => branch, 6 cycles                    |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes    | Mnemonic        |  |
|---------|----------|-----------------|--|
| \$90FF  | 0F 32 10 | BBR0 \$32, \$10 | ; BBR to different page, - BRANCH - jumps to \$9102 + \$10 |
| ...     |          |                 |  |
| \$9112  | ...      | <i>next OP</i>  | ; M0(\$0032)=0 => branch, 7 cycles                         |

## BBS

Operation: Branch on memory bit x is set, where x=0-7 MSB-8 of op code (Branch on Mx = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BBS(0-7) ZP,offs8      | 0F-7F   | 3         | 5*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (5): BBS – Short Reference

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes    | Mnemonic        |   |
|---------|----------|-----------------|---|
| \$9000  | 8F 32 10 | BBS0 \$32, \$10 | ; BBS to same page, - NO BRANCH - jumps to \$9003 |
| \$9003  | ...      | <i>next OP</i>  | ; M0(\$0032) =0 => no branch, 5 cycles            |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes    | Mnemonic        |   |
|---------|----------|-----------------|---|
| \$90FF  | FF 32 10 | BBS7 \$32, \$10 | ; BBS to different page, - NO BRANCH – jumps<br>; to \$9102 |
| \$9102  | ...      | <i>next OP</i>  | ; M7(\$0032)=0 => no branch, 5 cycles (!!!)                 |

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes    | Mnemonic        |  |
|---------|----------|-----------------|--|
| \$9000  | AF 32 10 | BBS2 \$32, \$10 | ; BBS to same page, - BRANCH - jumps to \$9003<br>; + \$10 |
| ...     |          |                 |  |
| \$9013  | ...      | <i>next OP</i>  | ; M2(\$0032)=1 => branch, 6 cycles                         |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes    | Mnemonic        |   |
|---------|----------|-----------------|---|
| \$90FF  | DF 32 10 | BBS5 \$32, \$10 | ; BBS to different page, - BRANCH - jumps to<br>; \$9102 + \$10 |
| ...     |          |                 |   |
| \$9112  | ...      | <i>next OP</i>  | ; M5(\$0032)=1 => branch, 7 cycles                              |

## BCC

Operation: Branch on Carry Clear (Branch on C = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BCC Oper               | 90      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (6): BCC – Short Reference

Example Relative (assumed C=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 90 10 | BCC \$10       | ; BCC to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; C=1 => no branch, 2 cycles                      |

Example Relative (assumed C=1, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | 90 10 | BCC \$10       | ; BCC to different page, - NO BRANCH - jumps to \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; C=1 => no branch, 2 cycles (!!!)                     |

Example Relative (assumed C=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 90 10 | BCC \$10       | ; BCC to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; C=0 => branch, 3 cycles                             |

Example Relative (assumed C=0, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | 90 10 | BCC \$10       | ; BCC to different page, - BRANCH - jumps to \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; C=0 => branch, 4 cycles                           |

## BCS

Operation: Branch on Carry Set (Branch on C = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BCS Oper               | B0      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (7): BCS – Short Reference

Example Relative (assumed C=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | B0 10 | BCS \$10       | ; BCS to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; C=0 => no branch, 2 cycles                      |

Example Relative (assumed C=0, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | B0 10 | BCS \$10       | ; BCS to different page, - NO BRANCH - jumps to \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; C=0 => no branch, 2 cycles (!!!)                     |

Example Relative (assumed C=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | B0 10 | BCS \$10       | ; BCS to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; C=1 => branch, 3 cycles                             |

Example Relative (assumed C=1, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | B0 10 | BCS \$10       | ; BCS to different page, - BRANCH - jumps to \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; C=1 => branch, 4 cycles                           |

## BEQ

Operation: Branch on result zero (Branch on Z = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BEQ Oper               | F0      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (8): BEQ – Short Reference

Example Relative (assumed Z=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | F0 10 | BEQ \$10       | ; BEQ to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; Z=0 => no branch, 2 cycles                      |



Example Relative (assumed Z=0, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | F0 10 | BEQ \$10       | ; BEQ to different page, - NO BRANCH - jumps to |
|         |       |                | ; \$9101  |
| \$9101  | ...   | <i>next OP</i> | ; Z=0 => no branch, 2 cycles (!!!)              |

Example Relative (assumed Z=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | F0 10 | BEQ \$10       | ; BEQ to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; Z=1 => branch, 3 cycles                             |

Example Relative (assumed Z=1, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | F0 10 | BEQ \$10       | ; BEQ to different page, - BRANCH - jumps to |
| ...     |       |                | ; \$9111                                     |
| \$9111  | ...   | <i>next OP</i> | ; Z=1 => branch, 4 cycles                    |

## BIT

Operation: Bit 6 and 7 are transferred to the status register. If the result of  $A \cap M$  then  $Z = 1$ , otherwise  $Z = 0$  ( $A \cap M \rightarrow A$ ,  $M7 \rightarrow N$ ,  $M6 \rightarrow V$ )

| Addressing Mode         | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate <i>NEW</i>    | BIT # Oper             | 89      | 2         | 2          | - | - | - | - | - | - | √ | - |
| Zero Page               | BIT Oper               | 24      | 2         | 3          | 7 | 6 | - | - | - | - | √ | - |
| Zero Page, X <i>NEW</i> | BIT Oper, X            | 34      | 2         | 4          | 7 | 6 | - | - | - | - | √ | - |
| Absolute                | BIT Oper               | 2C      | 3         | 4          | 7 | 6 | - | - | - | - | √ | - |
| Absolute, X <i>NEW</i>  | BIT Oper, X            | 3C      | 3         | 4*         | 7 | 6 | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (9): BIT – Short Reference

**In mode “Immediate” BIT leaves flags N&V unchanged. This is a special case on R65C02!**

## BMI

Operation: Branch on result minus (Branch on N = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BMI Oper               | 30      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (10): BMI – Short Reference

Example Relative (assumed Z=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 30 10 | BMI \$10       | ; BMI to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; N=0 => no branch, 2 cycles                      |

Example Relative (assumed Z=0, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | 30 10 | BMI \$10       | ; BMI to different page, - NO BRANCH - jumps to \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; N=0 => no branch, 2 cycles (!!!)                     |

Example Relative (assumed Z=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 30 10 | BMI \$10       | ; BMI to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; N=1 => branch, 3 cycles                             |

Example Relative (assumed Z=1, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | 30 10 | BMI \$10       | ; BMI to different page, - BRANCH - jumps to \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; N=1 => branch, 4 cycles                           |

## BNE

Operation: Branch on result not zero (Branch on  $Z = 0$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BNE Oper               | D0      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (11): BNE – Short Reference

Example Relative (assumed  $Z=1$ , same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | D0 10 | BNE \$10       | ; BNE to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; $Z=1 \Rightarrow$ no branch, 2 cycles           |

Example Relative (assumed  $Z=1$ , different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | D0 10 | BNE \$10       | ; BNE to different page, - NO BRANCH - jumps to \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; $Z=1 \Rightarrow$ no branch, 2 cycles (!!!)          |

Example Relative (assumed  $Z=0$ , same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | D0 10 | BNE \$10       | ; BNE to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; $Z=0 \Rightarrow$ branch, 3 cycles                  |

Example Relative (assumed  $Z=0$ , different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | D0 10 | BNE \$10       | ; BNE to different page, - BRANCH - jumps to \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; $Z=0 \Rightarrow$ branch, 4 cycles                |

## BPL

Operation: Branch on result plus (Branch on N = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BPL Oper               | 10      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (12): BPL – Short Reference

Example Relative (assumed N=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 10 10 | BPL \$10       | ; BPL to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; N=1 => no branch, 2 cycles                      |

Example Relative (assumed N=1, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | 10 10 | BPL \$10       | ; BPL to different page, - NO BRANCH - jumps to \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; N=1 => no branch, 2 cycles (!!!)                     |

Example Relative (assumed N=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 10 10 | BPL \$10       | ; BPL to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; N=0 => branch, 3 cycles                             |

Example Relative (assumed N=0, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | 10 10 | BPL \$10       | ; BPL to different page, - BRANCH - jumps to \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; N=0 => branch, 4 cycles                           |

## BRA

Operation: Branch allways (Branch allways)

| Addressing Mode     | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|---------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative <i>NEW</i> | BRA Oper               | 80      | 2         | 3*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to different page

Table (13): BRA – Short Reference

## BRK

Operation: Forced Interrupt (PC + 2 ↓, P ↓)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | BRK                    | 00      | 1         | 7          | - | - | - | - | - | 1 | - | - |

Table (14): BRK – Short Reference

## BVC

Operation: Branch on no overflow (Branch on V = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BVC Oper               | 50      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (15): BVC – Short Reference

Example Relative (assumed V=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 50 10 | BVC \$10       | ; BVC to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; V=1 => no branch, 2 cycles                      |

Example Relative (assumed V=1, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | 50 10 | BVC \$10       | ; BVC to different page, - NO BRANCH - jumps to ; \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; V=1 => no branch, 2 cycles (!!!)                       |

Example Relative (assumed V=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 50 10 | BVC \$10       | ; BVC to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; V=0 => branch, 3 cycles                             |

Example Relative (assumed V=0, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | 50 10 | BVC \$10       | ; BVC to different page, - BRANCH - jumps to ; \$9111 |
| ...     |       |                |   |
| \$9111  | ...   | <i>next OP</i> | ; V=0 => branch, 4 cycles                             |

## BVS

Operation: Branch on overflow (Branch on V = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative        | BVS Oper               | 70      | 2         | 2*         | - | - | - | - | - | - | - | - |

\* => Add 1 if branch occurs to same page

\* => Add 2 if branch occurs to different page

Table (16): BVS – Short Reference

Example Relative (assumed V=0, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 70 10 | BVS \$10       | ; BVS to same page, - NO BRANCH - jumps to \$9002 |
| \$9002  | ...   | <i>next OP</i> | ; V=0 => no branch, 2 cycles                      |

Example Relative (assumed V=0, different page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$90FF  | 70 10 | BVS \$10       | ; BVS to different page, - NO BRANCH - jumps to<br>; \$9101 |
| \$9101  | ...   | <i>next OP</i> | ; V=0 => no branch, 2 cycles (!!!)                          |

Example Relative (assumed V=1, same page):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 70 10 | BVS \$10       | ; BVS to same page, - BRANCH - jumps to \$9002 + \$10 |
| ...     |       |                |   |
| \$9012  | ...   | <i>next OP</i> | ; V=1 => branch, 3 cycles                             |

Example Relative (assumed V=1, different page):

| Address | Bytes | Mnemonic       |  |
|---------|-------|----------------|--|
| \$90FF  | 70 10 | BVS \$10       | ; BVS to different page, - BRANCH - jumps to<br>; \$9111 |
| ...     |       |                |  |
| \$9111  | ...   | <i>next OP</i> | ; V=1 => branch, 4 cycles                                |

## CLC

Operation: Clear Carry flag (0 → C)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | CLC                    | 18      | 1         | 2          | - | - | - | - | - | - | - | 0 |

Table (17): CLC – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | 18    | CLC            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; C is now 0 |

## CLD

Operation: Clear decimal flag (0 → D)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | CLD                    | D8      | 1         | 2          | - | - | - | - | 0 | - | - | - |

Table (18): CLD – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | D8    | CLD            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; D is now 0 |

## CLI

Operation: Clear interrupt disable flag (0 → I)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | CLI                    | 58      | 1         | 2          | - | - | - | - | - | 0 | - | - |

Table (19): CLI – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | 58    | CLI            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; I is now 0 |

## CLV

Operation: Clear overflow flag (0 → O)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | CLV                    | B8      | 1         | 2          | - | 0 | - | - | - | - | - | - |

Table (20): CLV – Short Reference



Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | B8    | CLV            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; V is now 0 |

## CMP

Operation: Compare Memory and Accumulator (A - M)

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | CMP # Oper             | C9      | 2         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page             | CMP Oper               | C5      | 2         | 3          | √ | - | - | - | - | - | √ | √ |
| Zero Page, X          | CMP Oper, X            | D5      | 2         | 4          | √ | - | - | - | - | - | √ | √ |
| Absolute              | CMP Oper               | CD      | 3         | 4          | √ | - | - | - | - | - | √ | √ |
| Absolute, X           | CMP Oper, X            | DD      | 3         | 4*         | √ | - | - | - | - | - | √ | √ |
| Absolute, Y           | CMP Oper, Y            | D9      | 3         | 4*         | √ | - | - | - | - | - | √ | √ |
| (Indirect, X)         | CMP (Oper, X)          | C1      | 2         | 6          | √ | - | - | - | - | - | √ | √ |
| (Indirect), Y         | CMP (Oper), Y          | D1      | 2         | 5*         | √ | - | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | CMP (Oper)             | D2      | 2         | 5          | √ | - | - | - | - | - | √ | √ |

\* => Add 1 if page boundary is crossed

Table (21): CMP – Short Reference

## CPX

Operation: Compare Memory and Index X (X - M)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate       | CPX # Oper             | E0      | 2         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page       | CPX Oper               | E4      | 2         | 3          | √ | - | - | - | - | - | √ | √ |
| Absolute        | CPX Oper               | EC      | 3         | 4          | √ | - | - | - | - | - | √ | √ |

Table (22): CPX – Short Reference

## CPY

Operation: Compare Memory and Index Y ( $Y - M$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate       | CPY # Oper             | C0      | 2         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page       | CPY Oper               | C4      | 2         | 3          | √ | - | - | - | - | - | √ | √ |
| Absolute        | CPY Oper               | CC      | 3         | 4          | √ | - | - | - | - | - | √ | √ |

Table (23): CPY – Short Reference

## DEC

Operation: Decrement Memory by one ( $M - 1 \rightarrow M$ )

| Addressing Mode        | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator <i>NEW</i> | DEC A                  | 3A      | 1         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page              | DEC Oper               | C6      | 2         | 5          | √ | - | - | - | - | - | √ | - |
| Zero Page, X           | DEC Oper, X            | D6      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| Absolute               | DEC Oper               | CE      | 3         | 6          | √ | - | - | - | - | - | √ | - |
| Absolute, X            | DEC Oper, X            | DE      | 3         | 7          | √ | - | - | - | - | - | √ | - |

Table (24): DEC – Short Reference

## DEX

Operation: Decrement index X by one ( $X - 1 \rightarrow X$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | DEX                    | CA      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (25): DEX – Short Reference

Example Implied (assume  $X = \$01$ ):

| Address | Bytes | Mnemonic |                           |
|---------|-------|----------|---------------------------|
| \$9000  | CA    | DEX      | ;                         |
| \$9001  | ...   | next OP  | ; X is now \$00, N=0, Z=1 |

DEX 0000 0001 (\$01)  
0000 0000 (\$00), N=0, Z=1

## DEY

Operation: Decrement index Y by one ( $Y - 1 \rightarrow Y$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | DEY                    | 88      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (26): DEX – Short Reference

Example Implied (assume Y=\$00):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | 88    | DEY            | ;                         |
| \$9001  | ...   | <i>next OP</i> | ; Y is now \$FF, N=1, Z=0 |

DEY 0000 0000 (\$00)  
1111 1111 (\$FF), N=1, Z=0

## EOR

Operation: “Exclusive-Or” memory or immediate value with accumulator A ( $A \nabla M \rightarrow A$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | EOR # Oper             | 49      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page             | EOR Oper               | 45      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, X          | EOR Oper, X            | 55      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute              | EOR Oper               | 4D      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, X           | EOR Oper, X            | 5D      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| Absolute, Y           | EOR Oper, Y            | 59      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| (Indirect, X)         | EOR (Oper, X)          | 41      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| (Indirect), Y         | EOR (Oper), Y          | 51      | 2         | 5*         | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | EOR (Oper)             | 52      | 2         | 5          | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (27): EOR – Short Reference

Example Immediate (assumed A is \$23):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | 49 63 | EOR #\$63      | ;                         |
| \$9002  | ...   | <i>next OP</i> | ; A is now \$40, N=0, Z=0 |

0110 0011 (\$63)  
 EOR 0010 0011 (\$23)  
0100 0000 (\$40), N=0, Z=0

## INC

Operation: Increment Memory by one ( $M + 1 \rightarrow M$ )

| Addressing Mode        | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator <i>NEW</i> | INC A                  | 1A      | 1         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page              | INC Oper               | E6      | 2         | 5          | √ | - | - | - | - | - | √ | - |
| Zero Page, X           | INC Oper, X            | F6      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| Absolute               | INC Oper               | EE      | 3         | 6          | √ | - | - | - | - | - | √ | - |
| Absolute, X            | INC Oper, X            | FE      | 3         | 7          | √ | - | - | - | - | - | √ | - |

Table (28): INC – Short Reference

## INX

Operation: Increment index X by one ( $X + 1 \rightarrow X$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | INX                    | E8      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (29): INX – Short Reference

Example Implied (assume X=\$0C):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | E8    | INX            | ;                         |
| \$9001  | ...   | <i>next OP</i> | ; X is now \$0D, N=0, Z=0 |

INX 0000 1100 (\$0C)  
0000 1101 (\$0D), N=0, Z=0

## INY

Operation: Increment index Y by one ( $Y + 1 \rightarrow Y$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | INY                    | C8      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (30): INY – Short Reference

Example Implied (assume  $Y = \$FF$ ):

| Address | Bytes | Mnemonic       |
|---------|-------|----------------|
| \$9000  | C8    | INY            |
| \$9001  | ...   | <i>next OP</i> |

INY 1111 1111 (\$FF)  
0000 0000 (\$00), N=0, Z=1

## JMP

Operation: Jump to new location ( $(PC + 1) \rightarrow PCL$ ,  $(PC + 2) \rightarrow PCH$ )

| Addressing Mode          | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Absolute                 | JMP Oper               | 4C      | 3         | 3          | - | - | - | - | - | - | - | - |
| (Absolute, X) <i>NEW</i> | JMP (Oper, X)          | 7C      | 3         | 6          | - | - | - | - | - | - | - | - |
| Indirect                 | JMP (Oper)             | 6C      | 3         | 6          | - | - | - | - | - | - | - | - |

Table (31): JMP – Short Reference

In mode “Indirect” JMP fetches the high-order byte of the effective address from the first byte of the next page, if the indirect address location spans a page boundary. The 6502 fetches it from the first byte of the current page.

R65C02: JMP (\$02FF) gets ADL from \$02FF, ADH from \$0300 (6 cycles)  
6502: JMP (\$02FF) gets ADL from \$02FF, ADH from \$0200 (5 cycles)

## JSR

Operation: Jump to subroutine saving return address ( $PC + 2 \downarrow$ ,  $(PC + 1) \rightarrow PCH$ ,  $(PC + 2) \rightarrow PCL$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Absolute        | JSR Oper               | 20      | 3         | 6          | - | - | - | - | - | - | - | - |

Table (32): JSR – Short Reference

Example:

| Address | Bytes    | Mnemonic     |
|---------|----------|--------------|
| \$9000  | 20 72 F0 | JSR \$F072 ; |
| F072    | ...      | next OP      |

## LDA

Operation: Load accumulator A with memory or immediate value ( $M \rightarrow A$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | LDA # Oper             | A9      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page             | LDA Oper               | A5      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, X          | LDA Oper, X            | B5      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute              | LDA Oper               | AD      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, X           | LDA Oper, X            | BD      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| Absolute, Y           | LDA Oper, Y            | B9      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| (Indirect, X)         | LDA (Oper, X)          | A1      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| (Indirect, Y)         | LDA (Oper), Y          | B1      | 2         | 5*         | √ | - | - | - | - | - | √ | - |
| (Indirect) <b>NEW</b> | LDA (Oper)             | B2      | 2         | 5          | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (33): LDA – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic                          |
|---------|-------|-----------------------------------|
| \$9000  | A9 00 | LDA #\$00 ;                       |
| \$9002  | ...   | next OP ; A is now \$00, N=0, Z=1 |

## LDX

Operation: Load index X with memory or immediate value ( $M \rightarrow X$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate       | LDX # Oper             | A2      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page       | LDX Oper               | A6      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, Y    | LDX Oper, Y            | B6      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute        | LDX Oper               | AE      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, Y     | LDX Oper, Y            | BE      | 3         | 4*         | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (34): LDX – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic                          |
|---------|-------|-----------------------------------|
| \$9000  | A2 8F | LDX #\$8F ;                       |
| \$9002  | ...   | next OP ; X is now \$8F, N=1, Z=0 |

## LDY

Operation: Load index Y with memory or immediate value ( $M \rightarrow Y$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate       | LDY # Oper             | A0      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page       | LDY Oper               | A4      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, X    | LDY Oper, X            | B4      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute        | LDY Oper               | AC      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, X     | LDY Oper, X            | BC      | 3         | 4*         | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (35): LDY – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic                          |
|---------|-------|-----------------------------------|
| \$9000  | A0 02 | LDY #\$02 ;                       |
| \$9002  | ...   | next OP ; Y is now \$02, N=0, Z=0 |

## LSR

Operation: Shift Right One Bit (Memory or Accumulator) (0 → 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

 → C)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator     | LSR A                  | 4A      | 1         | 2          | 0 | - | - | - | - | - | √ | √ |
| Zero Page       | LSR Oper               | 46      | 2         | 5          | 0 | - | - | - | - | - | √ | √ |
| Zero Page, X    | LSR Oper, X            | 56      | 2         | 6          | 0 | - | - | - | - | - | √ | √ |
| Absolute        | LSR Oper               | 4E      | 3         | 6          | 0 | - | - | - | - | - | √ | √ |
| Absolute, X     | LSR Oper, X            | 5E      | 3         | 7          | 0 | - | - | - | - | - | √ | √ |

Table (36): LSR – Short Reference

## NOP

Operation: No Operation

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | NOP                    | EA      | 1         | 2          | - | - | - | - | - | - | - | - |

Table (37): NOP – Short Reference

Example:

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | EA    | NOP            | ; |
| \$9001  | ...   | <i>next OP</i> | ; |



## ORA

Operation: “Or” memory or immediate value with accumulator A ( $A \vee M \rightarrow A$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | OR # Oper              | 09      | 2         | 2          | √ | - | - | - | - | - | √ | - |
| Zero Page             | OR Oper                | 05      | 2         | 3          | √ | - | - | - | - | - | √ | - |
| Zero Page, X          | OR Oper, X             | 15      | 2         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute              | OR Oper                | 0D      | 3         | 4          | √ | - | - | - | - | - | √ | - |
| Absolute, X           | OR Oper, X             | 1D      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| Absolute, Y           | OR Oper, Y             | 19      | 3         | 4*         | √ | - | - | - | - | - | √ | - |
| (Indirect, X)         | OR (Oper, X)           | 01      | 2         | 6          | √ | - | - | - | - | - | √ | - |
| (Indirect), Y         | OR (Oper), Y           | 11      | 2         | 5*         | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | OR (Oper)              | 12      | 2         | 5          | √ | - | - | - | - | - | √ | - |

\* => Add 1 if page boundary is crossed

Table (38): OR – Short Reference

Example Immediate (assumed A is \$8A):

| Address | Bytes | Mnemonic |                           |
|---------|-------|----------|---------------------------|
| \$9000  | 09 11 | OR #\$11 | ;                         |
| \$9002  | ...   | next OP  | ; A is now \$9B, N=1, Z=0 |

1000 1010 (\$8A)  
 OR 0001 0001 (\$11)  
 1001 1011 (\$9B), N=1, Z=0

## PHA

Operation: Push accumulator on stack ( $A \downarrow$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | PHA                    | 48      | 1         | 3          | - | - | - | - | - | - | - | - |

Table (39): PHA – Short Reference

## PHP

Operation: Push processor status on stack (P ↓)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | PHP                    | 08      | 1         | 3          | - | - | - | - | - | - | - | - |

Table (40): PHP – Short Reference

## PHX

Operation: Push index register X on stack (X ↓)

| Addressing Mode    | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PHX                    | DA      | 1         | 3          | - | - | - | - | - | - | - | - |

Table (41): PHX – Short Reference

## PHY

Operation: Push index register Y on stack (Y ↓)

| Addressing Mode    | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PHY                    | 5A      | 1         | 3          | - | - | - | - | - | - | - | - |

Table (42): PHY – Short Reference

## PLA

Operation: Pull accumulator from stack (A ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | PLA                    | 68      | 1         | 4          | √ | - | - | - | - | - | √ | - |

Table (43): PLA – Short Reference

## PLP

Operation: Pull processor status from stack (P ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N          | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|------------|---|---|---|---|---|---|---|
| Implied         | PLP                    | 28      | 1         | 4          | From Stack |   |   |   |   |   |   |   |

Table (44): PLP – Short Reference

## PLX

Operation: Pull index register X from stack (X ↑)

| Addressing Mode    | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PLX                    | FA      | 1         | 4          | √ | - | - | - | - | - | √ | - |

Table (45): PLX – Short Reference

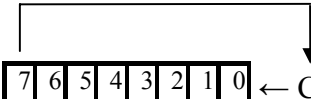
## PLY

Operation: Pull index register Y from stack (Y ↑)

| Addressing Mode    | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PLY                    | 7A      | 1         | 4          | √ | - | - | - | - | - | √ | - |

Table (46): PLY – Short Reference

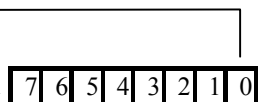
## ROL

Operation: Rotate one bit left (memory or accumulator) (  )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator     | ROL A                  | 2A      | 1         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page       | ROL Oper               | 26      | 2         | 5          | √ | - | - | - | - | - | √ | √ |
| Zero Page, X    | ROL Oper, X            | 36      | 2         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute        | ROL Oper               | 2E      | 3         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute, X     | ROL Oper, X            | 3E      | 3         | 7          | √ | - | - | - | - | - | √ | √ |

Table (47): ROL – Short Reference

## ROR

Operation: Rotate one bit right (memory or accumulator) (C →  )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator     | ROR A                  | 6A      | 1         | 2          | √ | - | - | - | - | - | √ | √ |
| Zero Page       | ROR Oper               | 66      | 2         | 5          | √ | - | - | - | - | - | √ | √ |
| Zero Page, X    | ROR Oper, X            | 76      | 2         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute        | ROR Oper               | 6E      | 3         | 6          | √ | - | - | - | - | - | √ | √ |
| Absolute, X     | ROR Oper, X            | 7E      | 3         | 7          | √ | - | - | - | - | - | √ | √ |

Table (48): ROR – Short Reference

## RMB

Operation: Reset memory bit x. Where x = 0-7 MSB of op code (0 → Mx)

| Addressing Mode      | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | RMB(0-7) Oper          | 07-77   | 2         | 5          | - | - | - | - | - | - | - | - |

Table (49): RMB – Short Reference

Example Zero Page:

| Address | Bytes | Mnemonic       |                      |
|---------|-------|----------------|----------------------|
| \$9000  | 37 14 | RMB3 \$14      | ; \$0014 = \$C9      |
| \$9002  | ...   | <i>next OP</i> | ; \$0014 is now \$C1 |

## RTI

Operation: Return from interrupt (P ↑, PC ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N          | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|------------|---|---|---|---|---|---|---|
| Implied         | RTI                    | 40      | 1         | 6          | From Stack |   |   |   |   |   |   |   |

Table (50): RTI – Short Reference

## RTS

Operation: Return from subroutine (PC ↑, PC + 1 → PC)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | RTS                    | 60      | 1         | 6          | - | - | - | - | - | - | - | - |

Table (51): RTS – Short Reference

## SBC

Operation: Subtract memory or immediate value from accumulator with borrow ( $A - M - \overline{C} \rightarrow A, C$ )

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate             | SBC # Oper             | E9      | 2         | 2          | √ | √ | - | - | - | - | √ | √ |
| Zero Page             | SBC Oper               | E5      | 2         | 3          | √ | √ | - | - | - | - | √ | √ |
| Zero Page, X          | SBC Oper, X            | F5      | 2         | 4          | √ | √ | - | - | - | - | √ | √ |
| Absolute              | SBC Oper               | ED      | 3         | 4          | √ | √ | - | - | - | - | √ | √ |
| Absolute, X           | SBC Oper, X            | FD      | 3         | 4*         | √ | √ | - | - | - | - | √ | √ |
| Absolute, Y           | SBC Oper, Y            | F9      | 3         | 4*         | √ | √ | - | - | - | - | √ | √ |
| (Indirect, X)         | SBC (Oper, X)          | E1      | 2         | 6          | √ | √ | - | - | - | - | √ | √ |
| (Indirect), Y         | SBC (Oper), Y          | F1      | 2         | 5*         | √ | √ | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | SBC (Oper)             | F2      | 2         | 5          | √ | √ | - | - | - | - | √ | √ |

\* => Add 1 if page boundary is crossed

Table (52): SBC – Short Reference

## SEC

Operation: Set carry flag (1 → C)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | SEC                    | 38      | 1         | 2          | - | - | - | - | - | - | - | 1 |

Table (53): SEC – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | 38    | SEC            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; C is now 1 |

## SED

Operation: Set decimal flag (1 → D)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | SED                    | F8      | 1         | 2          | - | - | - | - | 1 | - | - | - |

Table (54): CLD – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | F8    | SED            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; D is now 1 |

## SEI

Operation: Set interrupt disable flag (1 → I)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | SEI                    | 78      | 1         | 2          | - | - | - | - | - | 1 | - | - |

Table (55): SEI – Short Reference

Example:

| Address | Bytes | Mnemonic       |              |
|---------|-------|----------------|--------------|
| \$9000  | 78    | SEI            | ;            |
| \$9001  | ...   | <i>next OP</i> | ; I is now 1 |

## SMB

Operation: Set memory bit x. Where x = 0-7 (MSB – 8) of op code (1 → Mx)

| Addressing Mode      | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | SMB(0-7) Oper          | 87-F7   | 2         | 5          | - | - | - | - | - | - | - | - |

Table (56): SMB – Short Reference

Example Zero Page:

| Address | Bytes | Mnemonic       |                      |
|---------|-------|----------------|----------------------|
| \$9000  | F7 14 | SMB \$F0       | ; \$00F0 = \$43      |
| \$9002  | ...   | <i>next OP</i> | ; \$00F0 is now \$C3 |

## STA

Operation: Store accumulator in memory (A → M)

| Addressing Mode       | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page             | STA Oper               | 85      | 2         | 3          | - | - | - | - | - | - | - | - |
| Zero Page, X          | STA Oper, X            | 95      | 2         | 4          | - | - | - | - | - | - | - | - |
| Absolute              | STA Oper               | 8D      | 3         | 4          | - | - | - | - | - | - | - | - |
| Absolute, X           | STA Oper, X            | 9D      | 3         | 5          | - | - | - | - | - | - | - | - |
| Absolute, Y           | STA Oper, Y            | 99      | 3         | 5          | - | - | - | - | - | - | - | - |
| (Indirect, X)         | STA (Oper, X)          | 81      | 2         | 6          | - | - | - | - | - | - | - | - |
| (Indirect), Y         | STA (Oper), Y          | 91      | 2         | 6          | - | - | - | - | - | - | - | - |
| (Indirect) <i>NEW</i> | STA (Oper)             | 92      | 2         | 5          | - | - | - | - | - | - | - | - |

Table (57): STA – Short Reference

## STX

Operation: Store index X in memory ( $X \rightarrow M$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page       | STX Oper               | 86      | 2         | 3          | - | - | - | - | - | - | - | - |
| Zero Page, Y    | STX Oper, Y            | 96      | 2         | 4          | - | - | - | - | - | - | - | - |
| Absolute        | STX Oper               | 8E      | 3         | 4          | - | - | - | - | - | - | - | - |

Table (58): STX – Short Reference

## STY

Operation: Store index Y in memory ( $Y \rightarrow M$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page       | STY Oper               | 84      | 2         | 3          | - | - | - | - | - | - | - | - |
| Zero Page, X    | STY Oper, X            | 94      | 2         | 4          | - | - | - | - | - | - | - | - |
| Absolute        | STY Oper               | 8C      | 3         | 4          | - | - | - | - | - | - | - | - |

Table (59): STY – Short Reference

## STZ

Operation: Store value #00 in memory ( $\#00 \rightarrow M$ )

| Addressing Mode         | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i>    | STZ Oper               | 64      | 2         | 3          | - | - | - | - | - | - | - | - |
| Zero Page, X <i>NEW</i> | STZ Oper, X            | 74      | 2         | 4          | - | - | - | - | - | - | - | - |
| Absolute <i>NEW</i>     | STZ Oper               | 9C      | 3         | 4          | - | - | - | - | - | - | - | - |
| Absolute, X <i>NEW</i>  | STZ Oper, X            | 9E      | 3         | 5          | - | - | - | - | - | - | - | - |

Table (60): STZ – Short Reference



## TAX

Operation: Transfer accumulator A to index X ( $A \rightarrow X$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TAX                    | AA      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (61): TAX – Short Reference

Example Implied (assume A=\$5D):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | AA    | TAX            | ;                         |
| \$9001  | ...   | <i>next OP</i> | ; X is now \$5D, N=0, Z=0 |

## TAY

Operation: Transfer accumulator A to index Y ( $A \rightarrow Y$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TAY                    | A8      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (62): TAY – Short Reference

Example Implied (assume A=\$89):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | A8    | TAY            | ;                         |
| \$9001  | ...   | <i>next OP</i> | ; Y is now \$89, N=1, Z=0 |

## TRB

Operation: Test memory with accumulator A and reset memory bits. If the result of  $A \cap M = 0$  then Z = 1, otherwise Z = 0 ( $\overline{A \cap M} \rightarrow Z$ )

| Addressing Mode      | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | TRB Oper               | 14      | 2         | 5          | - | - | - | - | - | - | √ | - |
| Absolute <i>NEW</i>  | TRB Oper               | 1C      | 3         | 6          | - | - | - | - | - | - | √ | - |

Table (63): TRB – Short Reference

Example Zero Page (assume A=\$03):

| Address | Bytes | Mnemonic                          |
|---------|-------|-----------------------------------|
| \$9000  | 14 D2 | TRB \$D2 ; \$00D2 = \$61          |
| \$9002  | ...   | next OP ; \$00D2 is now \$60, Z=0 |

```

0000 0011 (A=$03)
NOT 1111 1100
AND 0110 0001 (M=$61)
0110 0000 ($60) New value for memory

```

```

0000 0011 (A=$03)
AND 0110 0001 (M=$61)
0000 0001 Z=0 New value for flag Z

```

## TSB

Operation: Test memory with accumulator A and set memory bits. If the result of  $A \cap M = 0$  then  $Z = 1$ , otherwise  $Z = 0$  ( $A \cup M \rightarrow M$ )

| Addressing Mode      | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | TSB Oper               | 04      | 2         | 5          | - | - | - | - | - | - | √ | - |
| Absolute <i>NEW</i>  | TSB Oper               | 0C      | 3         | 6          | - | - | - | - | - | - | √ | - |

Table (64): TSB – Short Reference

Example Zero Page (assume A=\$0C):

| Address | Bytes | Mnemonic                          |
|---------|-------|-----------------------------------|
| \$9000  | 04 D2 | TSB \$D2 ; \$00D2 = \$11          |
| \$9002  | ...   | next OP ; \$00D2 is now \$1D, Z=1 |

```

0000 1100 (A=$0C)
OR 0001 0001 (M=$11)
0001 1101 ($1D) New value for memory

```

```

0000 1100 (A=$0C)
AND 0001 0001 (M=$11)
0000 0000 Z=1 New value for flag Z

```

## TSX

Operation: Transfer stack pointer SP to index X ( $SP \rightarrow X$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TSX                    | BA      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (65): TSX – Short Reference

Example Implied (assume  $SP = \$F2$ ):

| Address | Bytes | Mnemonic                                 |
|---------|-------|--|
| \$9000  | BA    | TSX                                      |
| \$9001  | ...   | <i>next OP</i> ; X is now \$F2, N=1, Z=0 |

## TXA

Operation: Transfer index X to accumulator A ( $X \rightarrow A$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TXA                    | 8A      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (66): TXA – Short Reference

Example Implied (assume  $X = \$00$ ):

| Address | Bytes | Mnemonic                                 |
|---------|-------|--|
| \$9000  | 8A    | TXA                                      |
| \$9001  | ...   | <i>next OP</i> ; A is now \$00, N=0, Z=1 |

## TXS

Operation: Transfer index X to stack pointer SP ( $X \rightarrow SP$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TXS                    | 9A      | 1         | 2          | - | - | - | - | - | - | - | - |

Table (67): TXS – Short Reference

Example Implied (assume X=\$E0):

| Address | Bytes | Mnemonic       |   |
|---------|-------|----------------|---|
| \$9000  | 9A    | TXS            |   |
| \$9001  | ...   | <i>next OP</i> | ; SP is now \$E0 (all flags are uneffected) |

## TYA

Operation: Transfer index Y to accumulator A ( $Y \rightarrow A$ )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied         | TYA                    | 98      | 1         | 2          | √ | - | - | - | - | - | √ | - |

Table (68): TYA – Short Reference

Example Implied (assume Y=\$14):

| Address | Bytes | Mnemonic       |                           |
|---------|-------|----------------|---------------------------|
| \$9000  | 98    | TYA            | ;                         |
| \$9001  | ...   | <i>next OP</i> | ; A is now \$14, N=0, Z=0 |

# 4

## Registers

This section specifies all internal registers. It should completely cover the interface between the core and the host as seen from the software view.

### List of Registers

| Name | Address | Width | Access | Description |
|------|---------|-------|--------|-------------|
|      |         |       |        |             |
|      |         |       |        |             |
|      |         |       |        |             |

Table 1: List of registers

### Register 1 – Description

(You shall choose the style of register you prefer. Do not use both options in one and the same document.)

| Bit # | Access | Description |
|-------|--------|-------------|
|       |        |             |
|       |        |             |
|       |        |             |

*Reset Value:*

Reg\_Name: 0000h

| 31 | 30 | 29 | 28 | ... | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|-----|---|---|---|---|---|---|---|---|---|
|    |    |    |    |     |   |   |   |   |   |   |   |   |   |
|    |    |    |    |     |   |   |   |   |   |   |   |   |   |

Table 2: Description of registers

*Reset Value:*

Reg\_Name: 0000h

# 5

## Clocks

This section specifies all the clocks. All clocks, clock domain passes and the clock relations should be described.

| Name | Source | Rates (MHz) |     |            | Remarks | Description |
|------|--------|-------------|-----|------------|---------|-------------|
|      |        | Max         | Min | Resolution |         |             |
|      |        |             |     |            |         |             |
|      |        |             |     |            |         |             |
|      |        |             |     |            |         |             |

Table 3: List of clocks

# Appendix A

---

## Timing Diagrams

## ADC, SBC

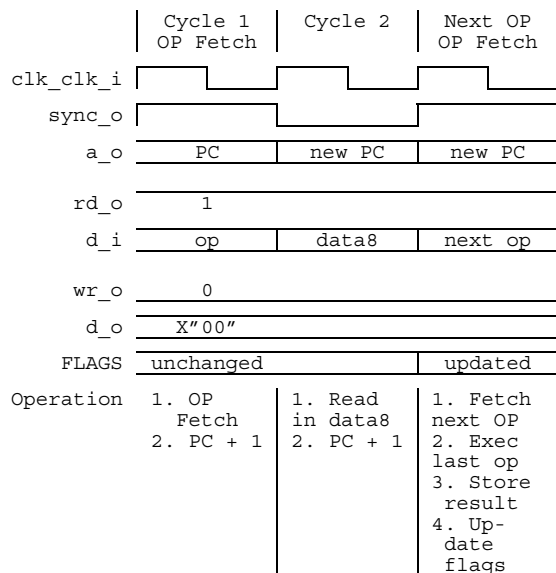


Figure (3): ADC, SBC – Timing Diagram “Immediate”

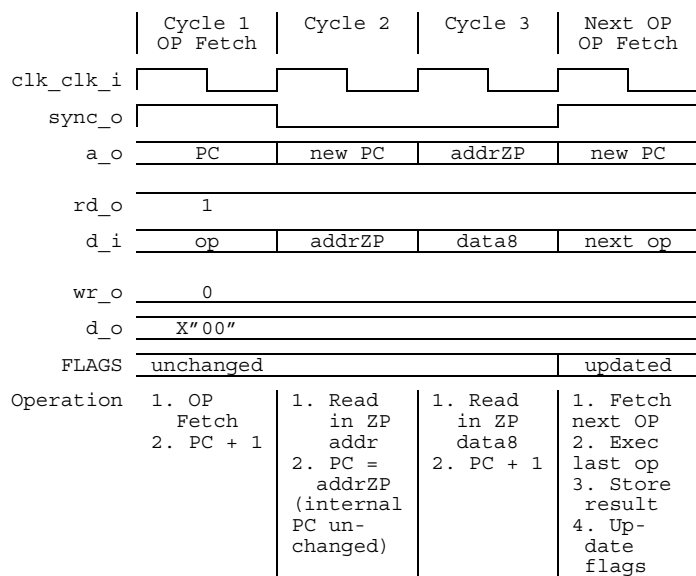


Figure (4): ADC, SBC – Timing Diagram “Zero Page”



## ADC, SBC (cont.)

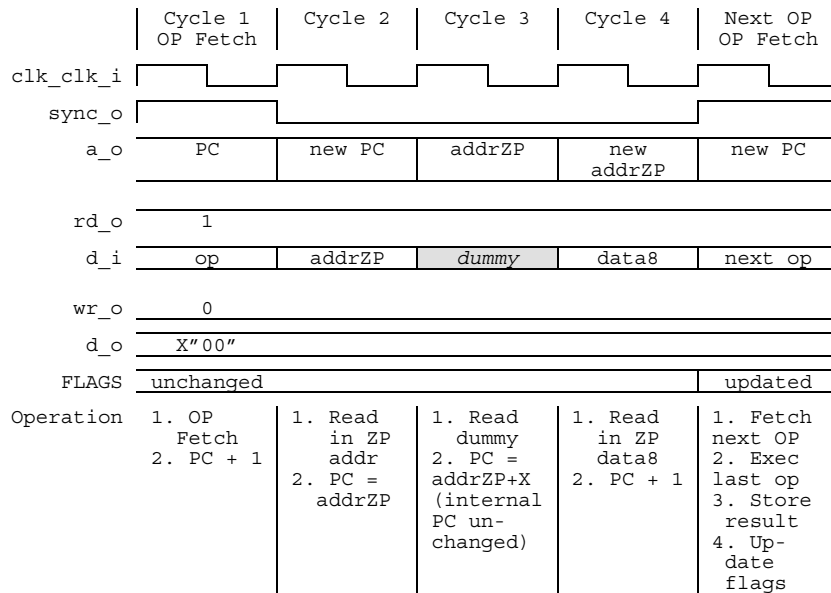


Figure (5): ADC, SBC – Timing Diagram “Zero Page, X”

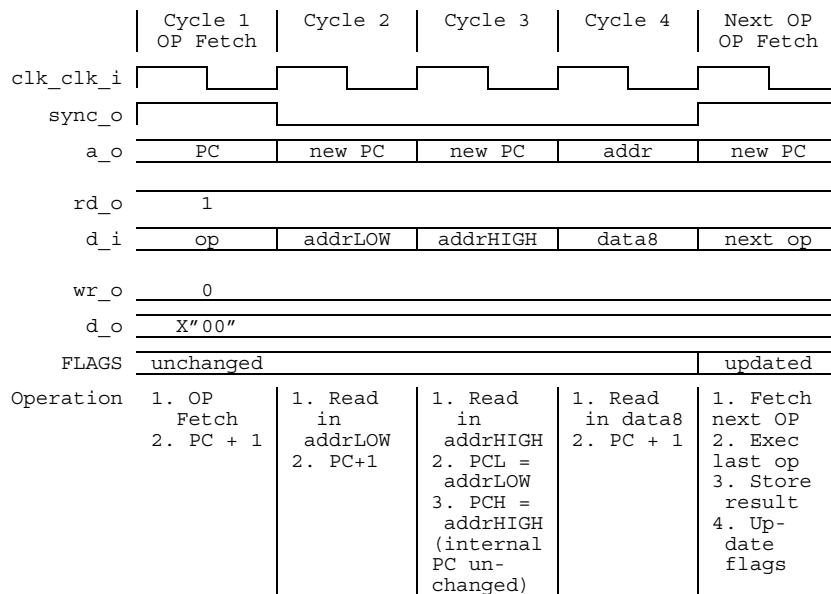


Figure (6): ADC, SBC – Timing Diagram “Absolute”

## ADC, SBC (cont.)

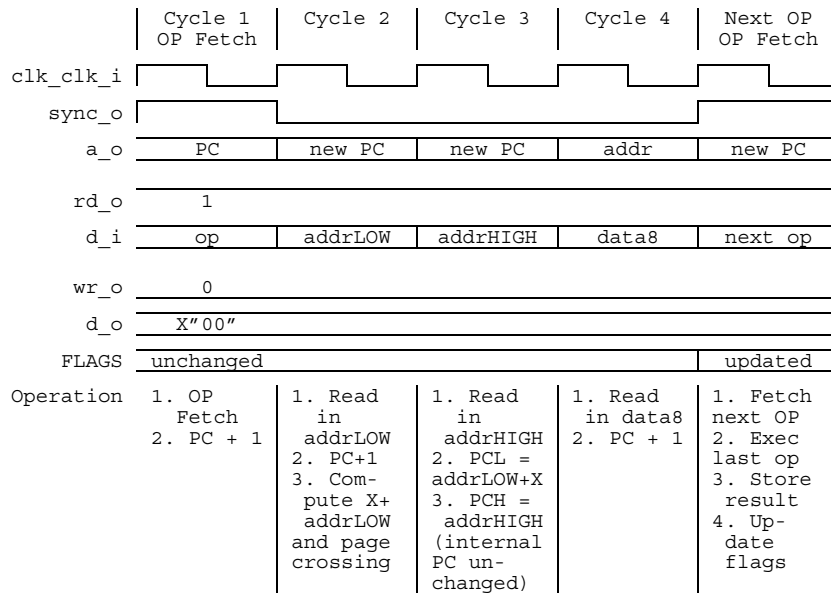


Figure (7): ADC, SBC – Timing Diagram “Absolute, X” – no page crossing

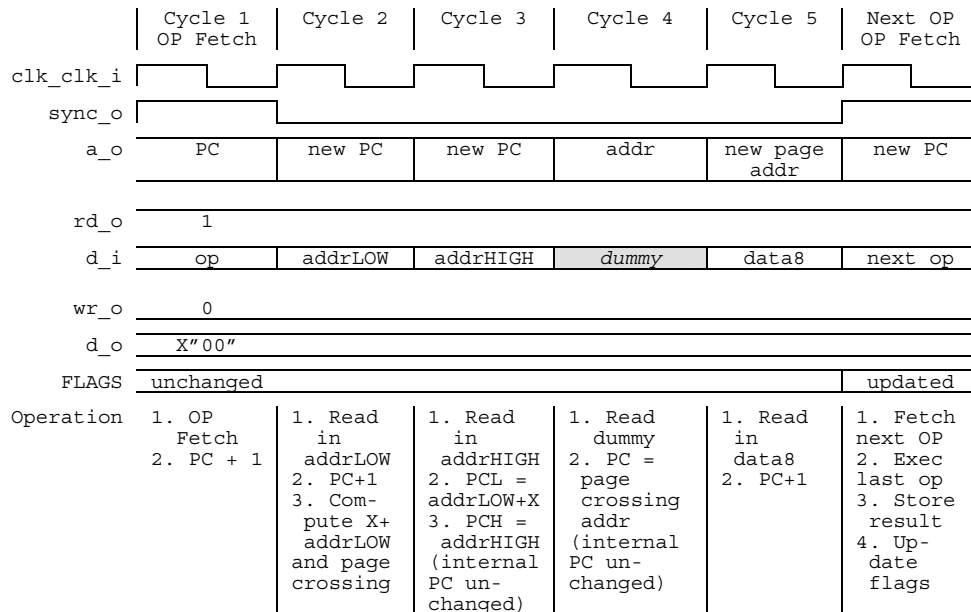


Figure (8): ADC, SBC – Timing Diagram “Absolute, X” – page crossing

## ADC, SBC (cont.)

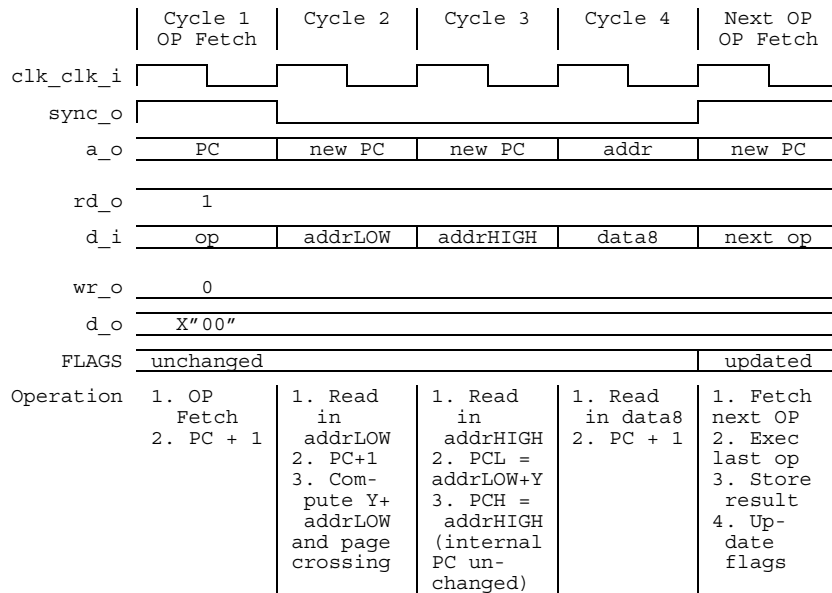


Figure (9): ADC, SBC – Timing Diagram “Absolute, Y” – no page crossing

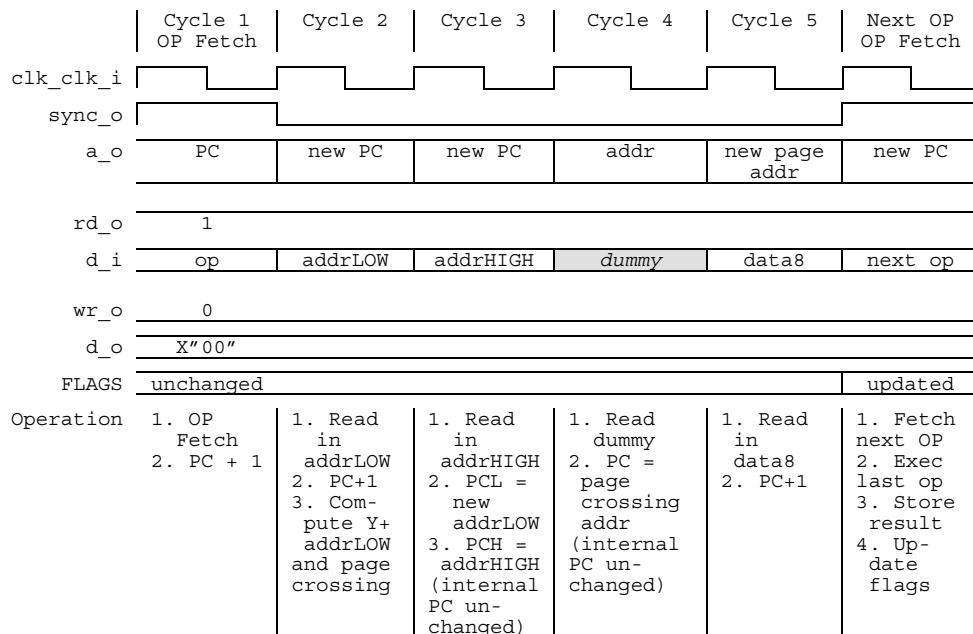


Figure (10): ADC, SBC – Timing Diagram “Absolute, Y” – page crossing

## ADC, SBC (cont.)

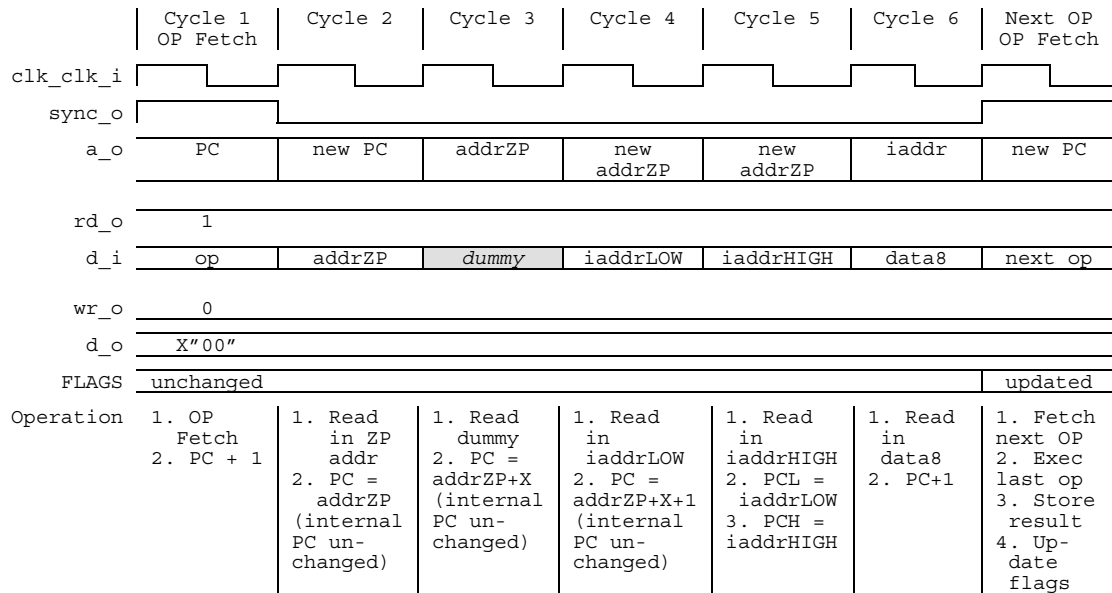


Figure (11): ADC, SBC – Timing Diagram “(Indirect, X)”

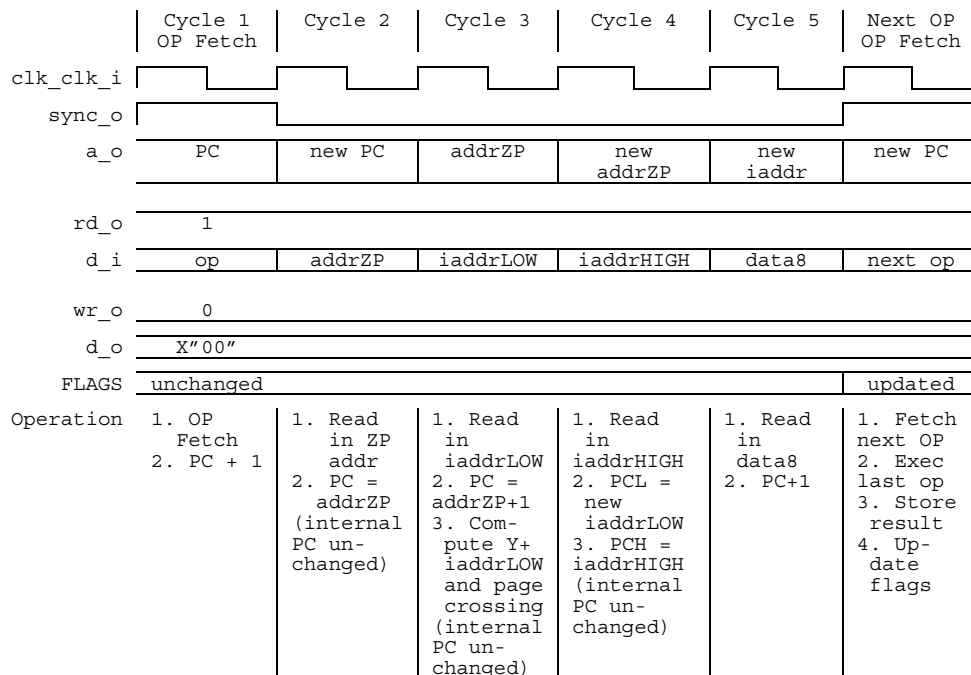


Figure (12): ADC, SBC – Timing Diagram “(Indirect, Y)” – no page crossing

## ADC, SBC (cont.)

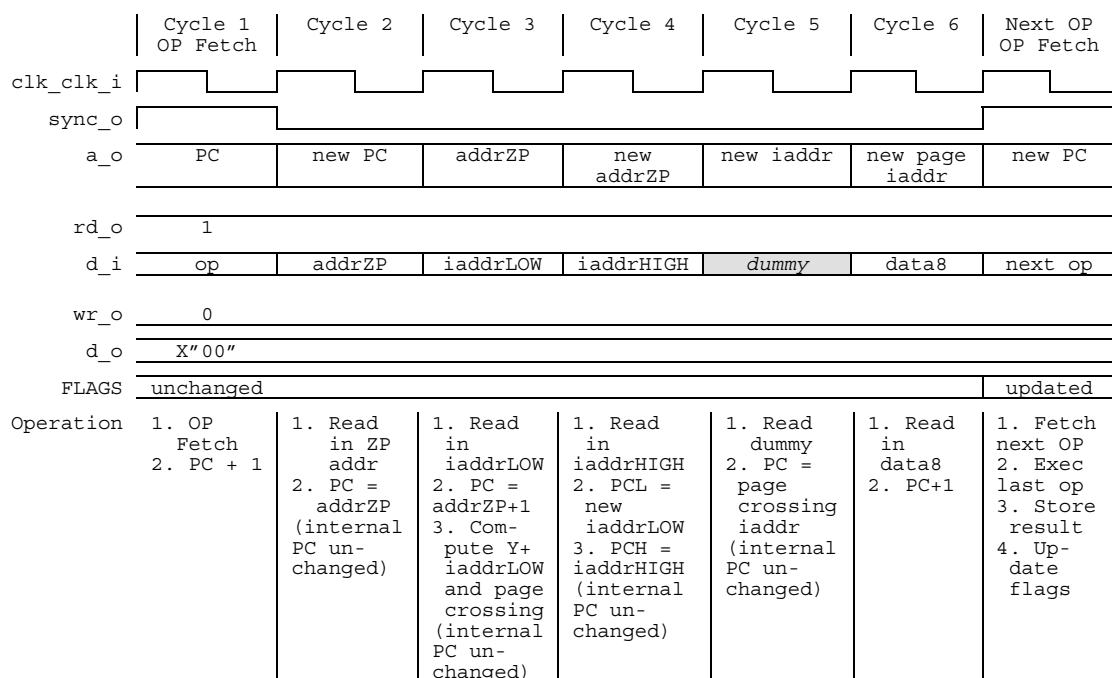


Figure (13): ADC, SBC – Timing Diagram “(Indirect), Y” – page crossing

## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA

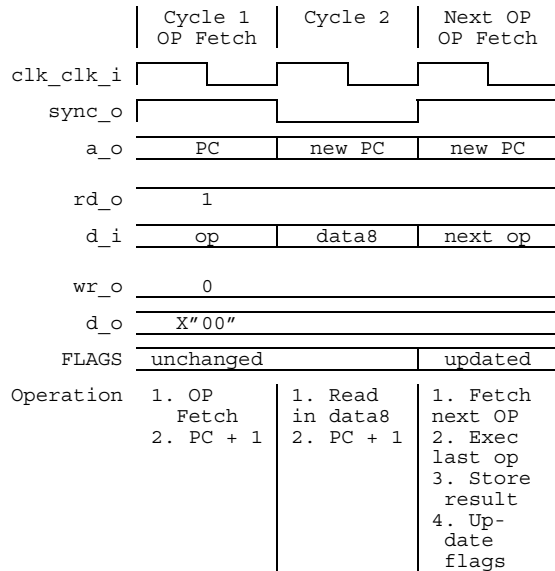


Figure (14): AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Immediate”

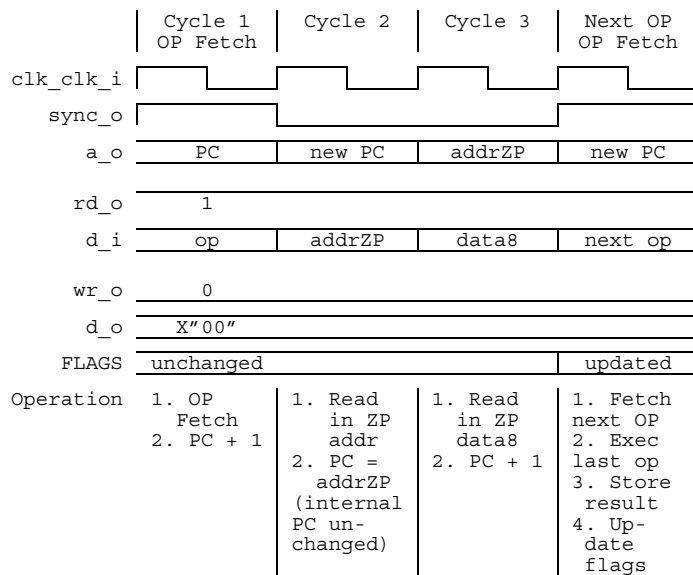


Figure (15): AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Zero Page”

## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA (cont.)

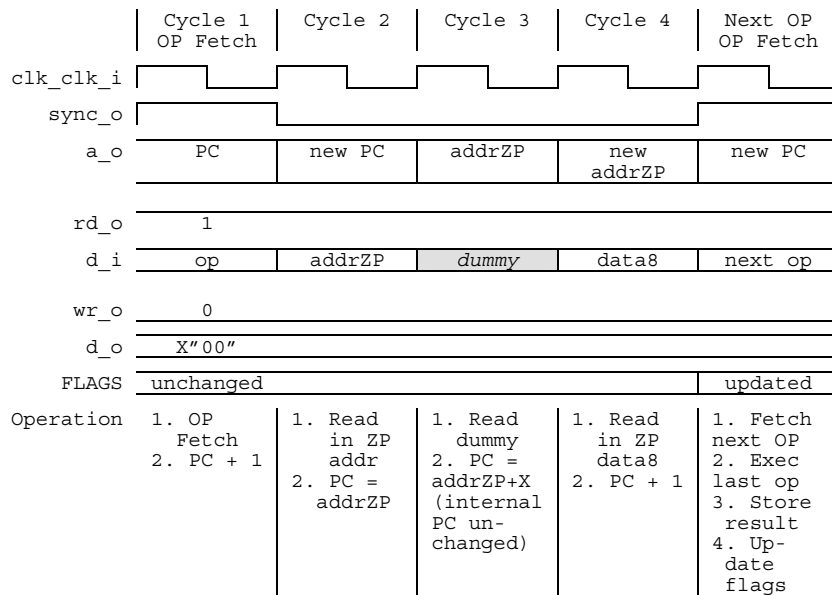


Figure (16): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Zero Page, X”

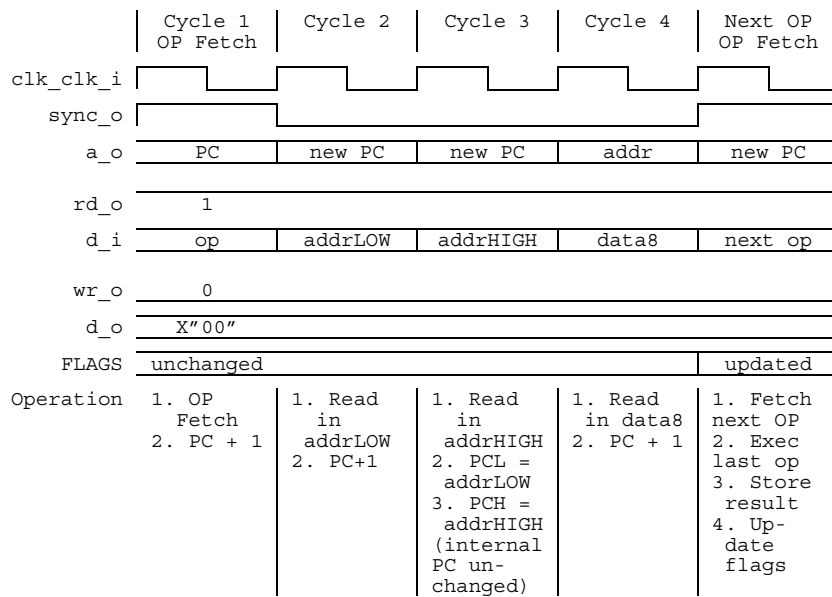


Figure (17): AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA – Timing Diagram “Absolute”

## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA (cont.)

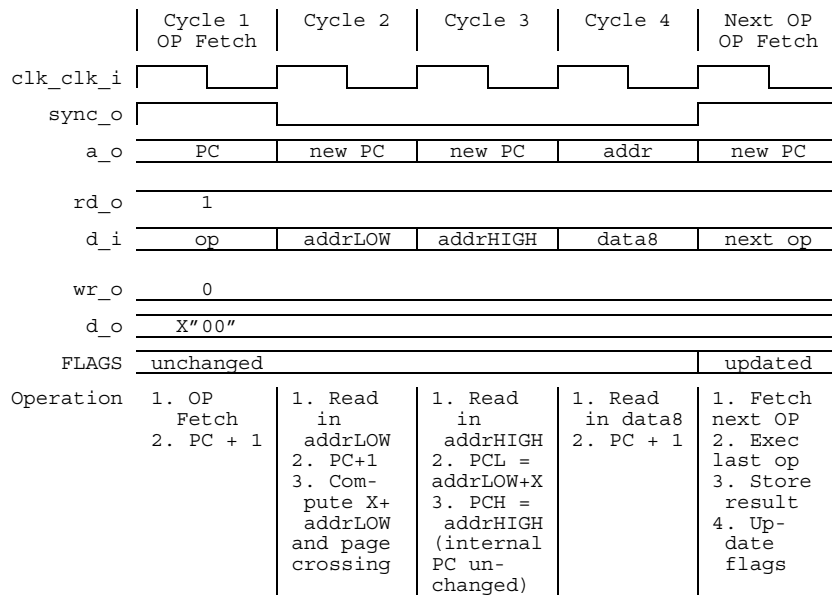


Figure (18): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Absolute, X” – no page crossing

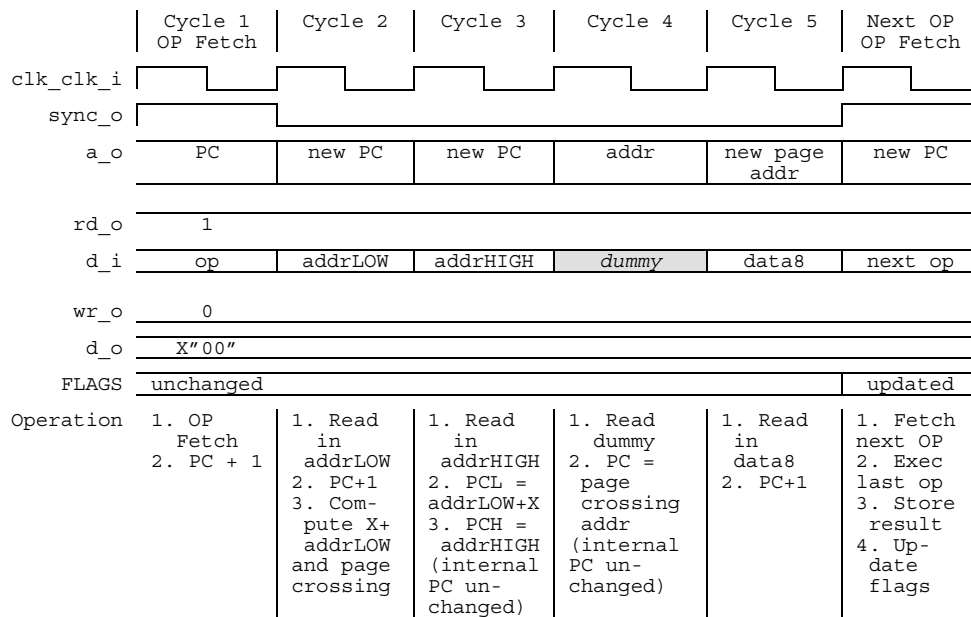


Figure (19): AND, CMP, EOR, LDA, LDY, ORA – Timing Diagram “Absolute, X” – page crossing



## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA (cont.)

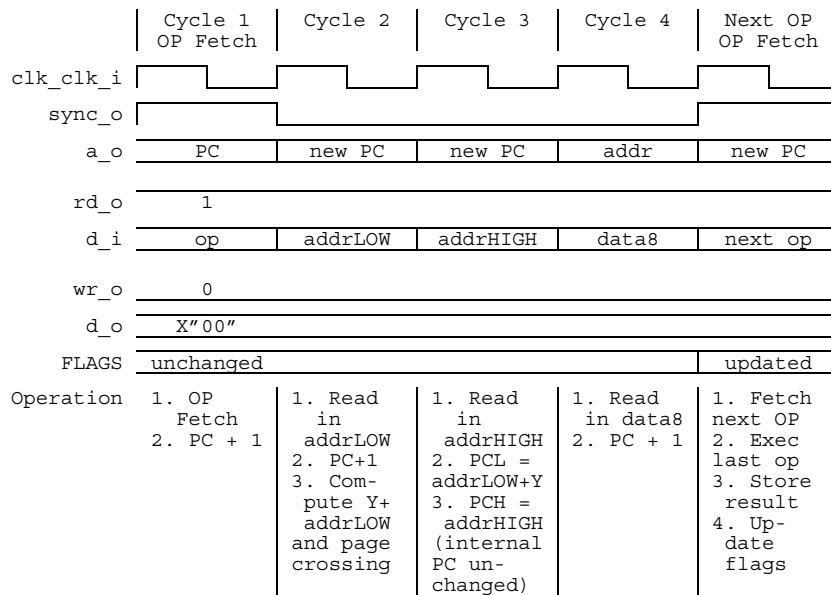


Figure (20): AND, CMP, EOR, LDA, LDX, ORA – Timing Diagram “Absolute, Y” – no page crossing

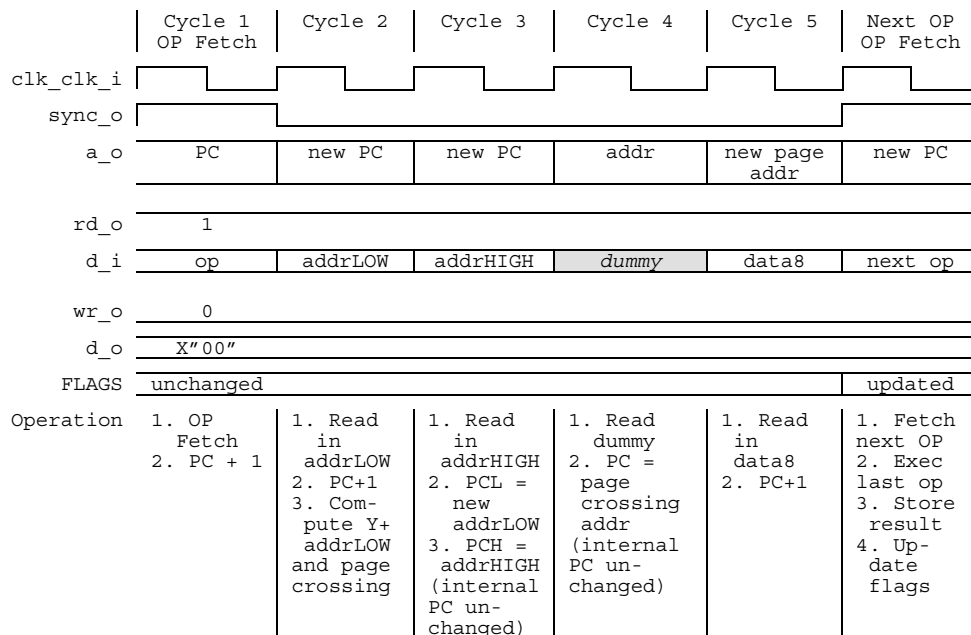


Figure (21): AND, CMP, EOR, LDA, LDX, ORA – Timing Diagram “Absolute, Y” – page crossing

## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA (cont.)

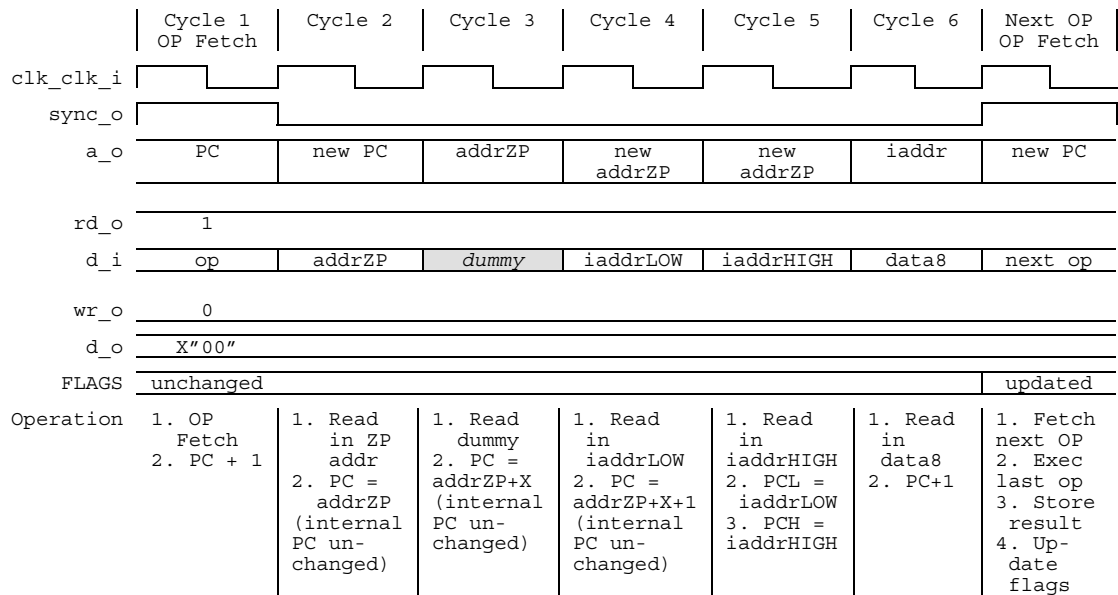


Figure (22): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect, X)”

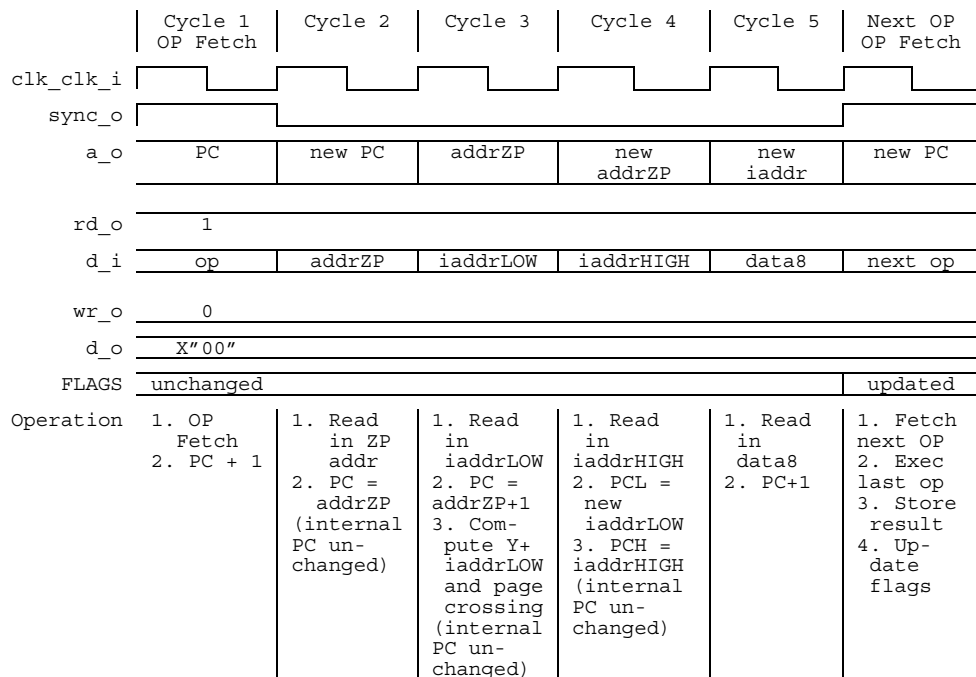


Figure (23): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect, Y)” – no page crossing

## AND, BIT, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA (cont.)

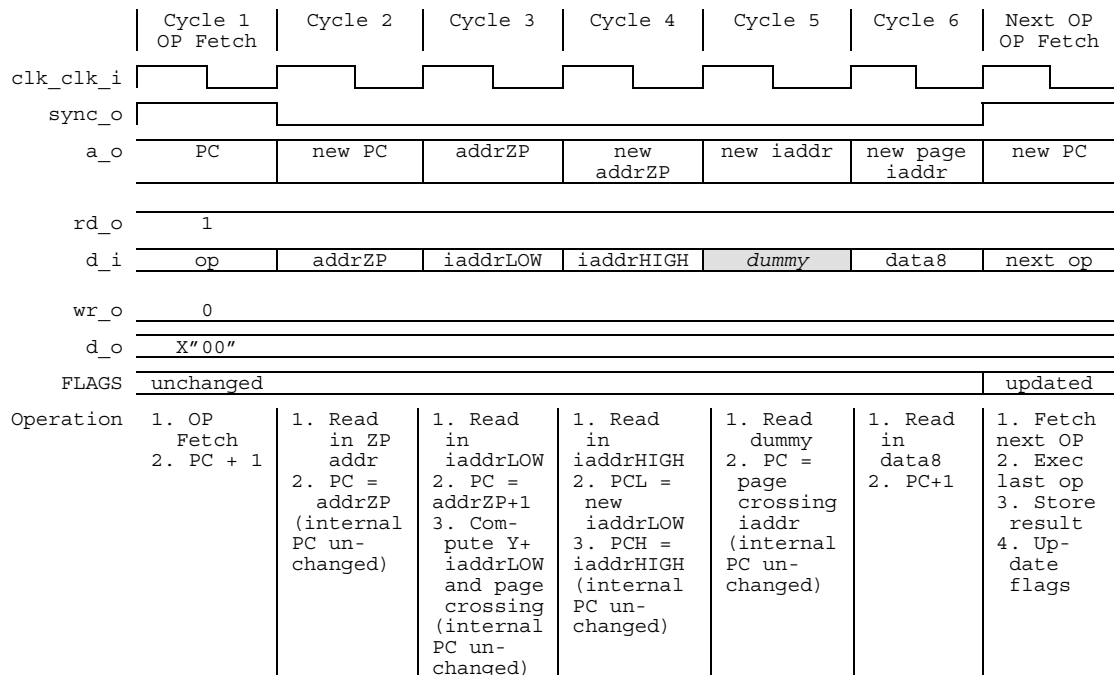


Figure (24): AND, CMP, EOR, LDA, ORA – Timing Diagram “(Indirect, Y” – page crossing

## BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE

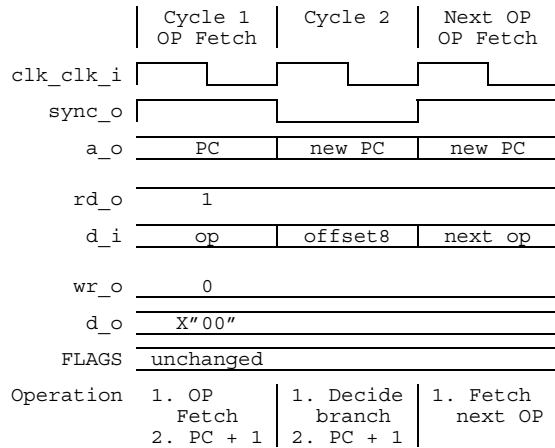


Figure (25): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “no branch, same page”

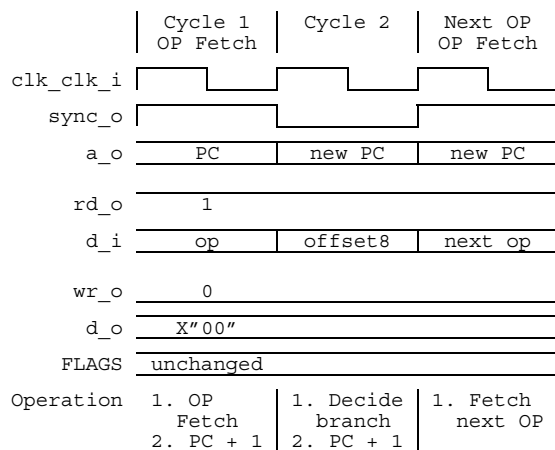


Figure (26): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “no branch, different page”

## BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE (cont.)

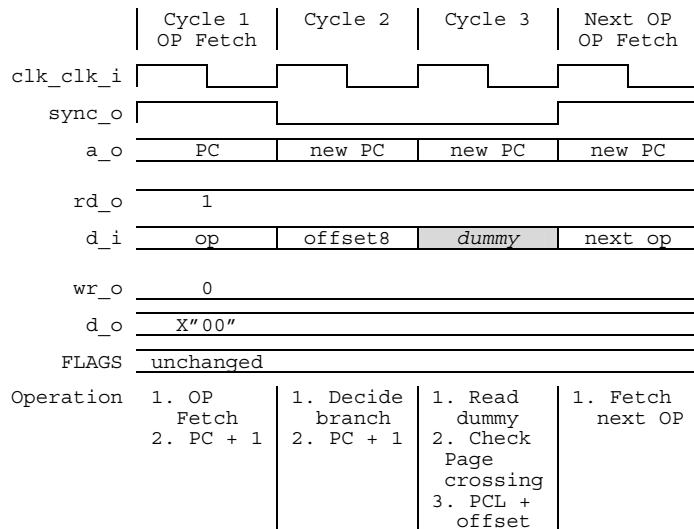


Figure (27): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “branch, same page”

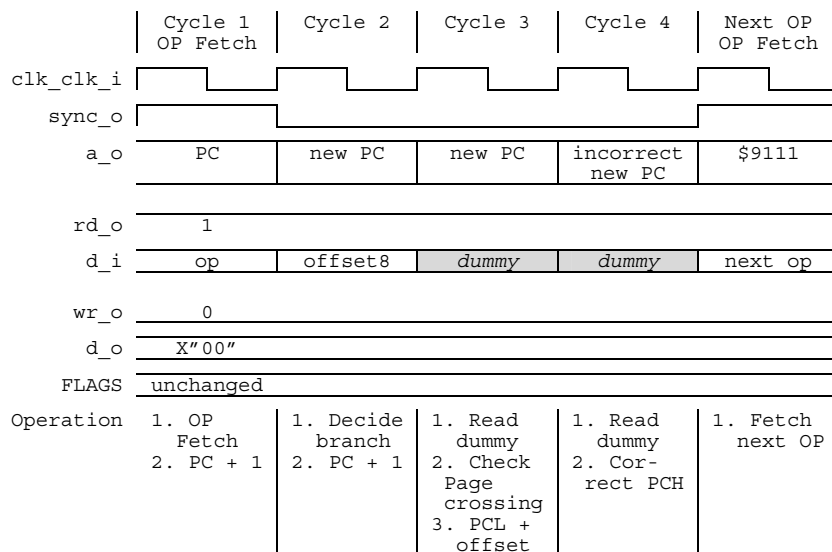


Figure (28): BCC, BCS, BVC, BVS, BPL, BMI, BEQ, BNE – Timing Diagram “branch, different page”

## BRK

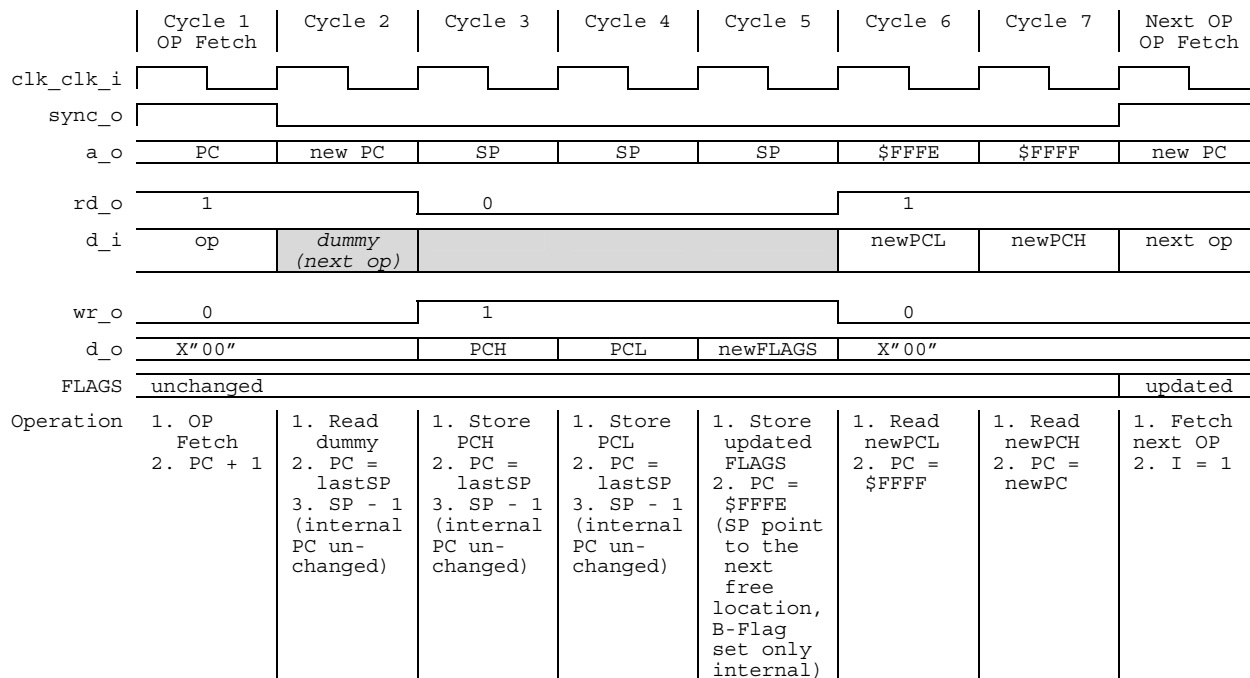


Figure (29): BRK – Timing Diagram “Implied”

## ASL, LSR, ROL, ROR, DEC, INC

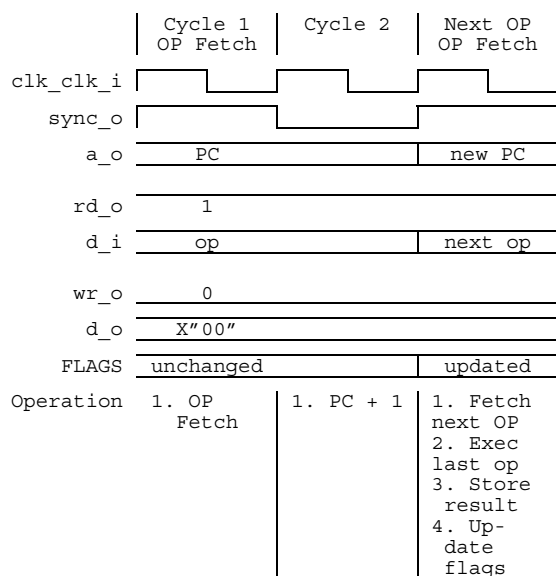


Figure (30): ASL A, LSR A, ROL A, ROR A – Timing Diagram “Immediate”

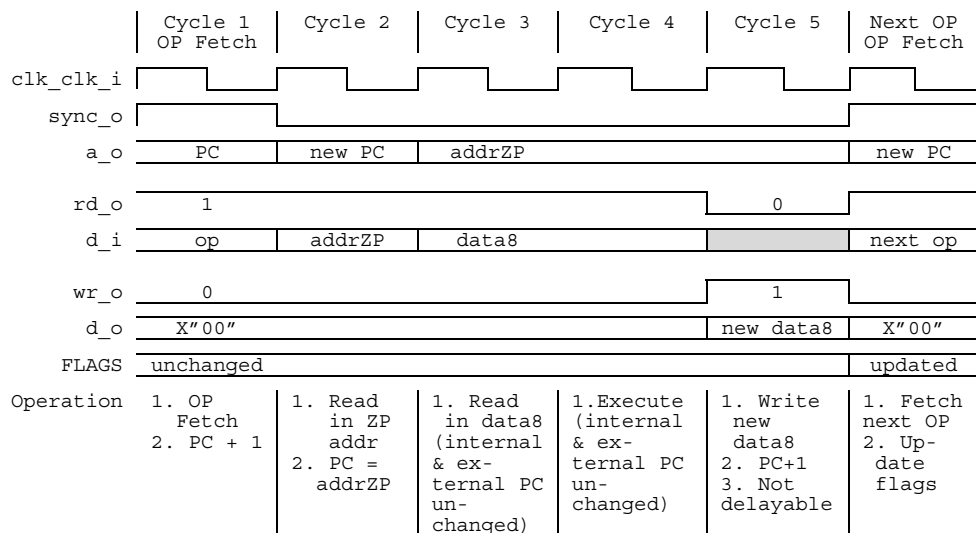


Figure (31): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Zero Page”

## ASL, LSR, ROL, ROR, DEC, INC (cont.)

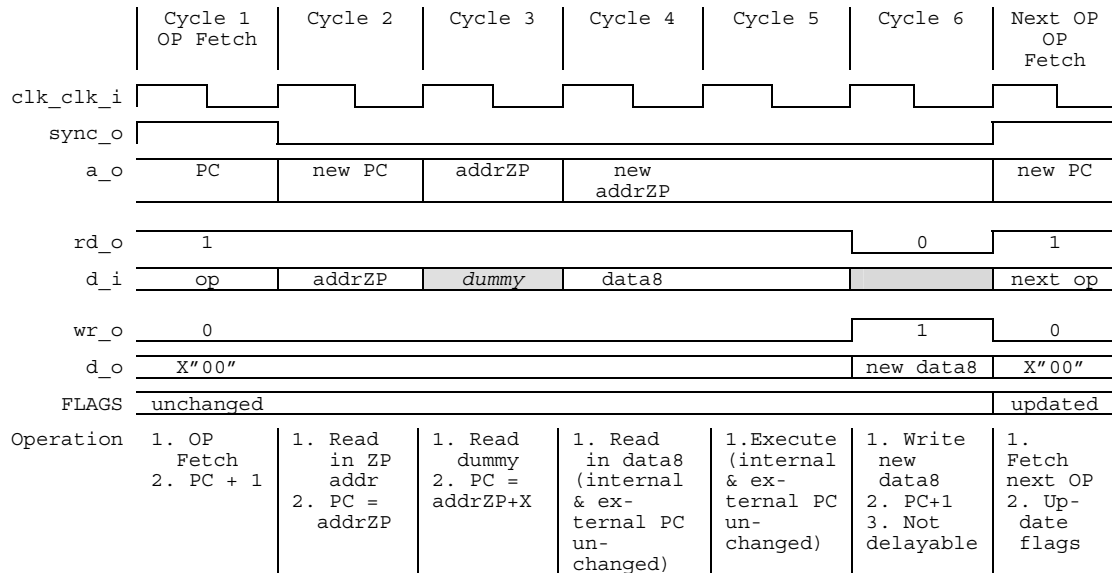


Figure (32): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Zero Page, X”

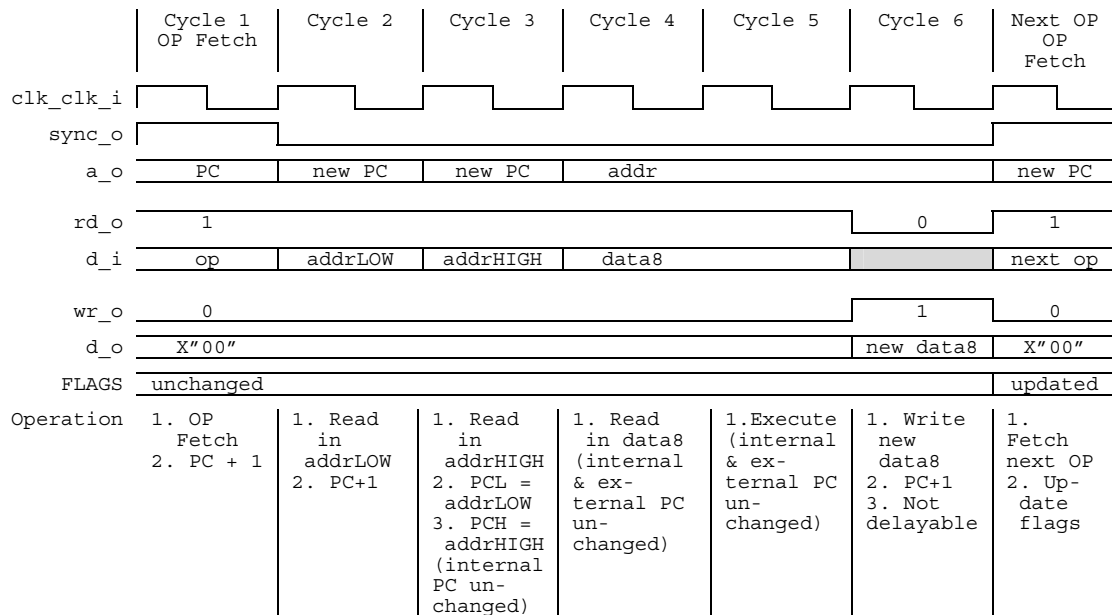


Figure (33): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Absolute”



## ASL, LSR, ROL, ROR, DEC, INC (cont.)

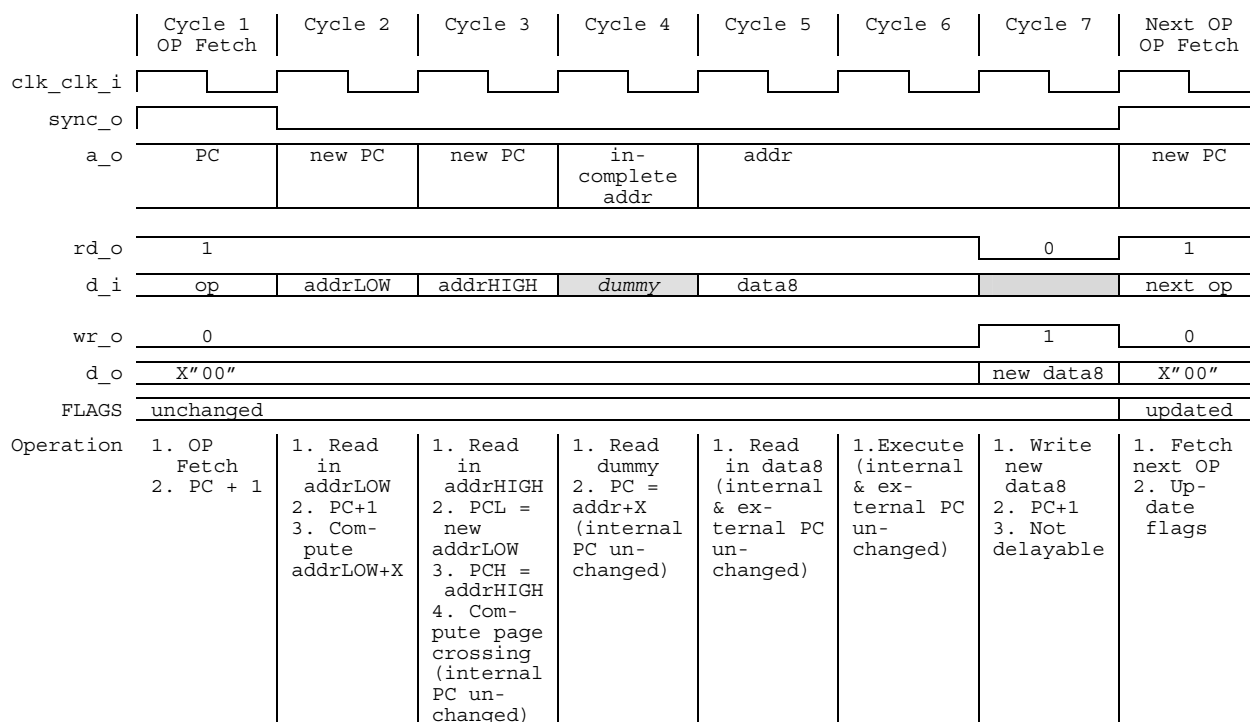


Figure (34): ASL, LSR, ROL, ROR, DEC, INC – Timing Diagram “Absolute, X”

# CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, SEC, SED, SEI, TAX, TAY, TSX, TXA, TYA

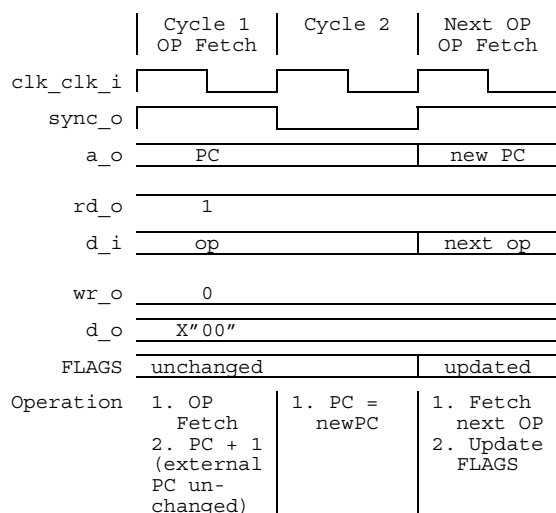


Figure (35): CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, SEC, SED, SEI, TAX, TAY, TSX, TXA, TYA – Timing Diagram

## JMP

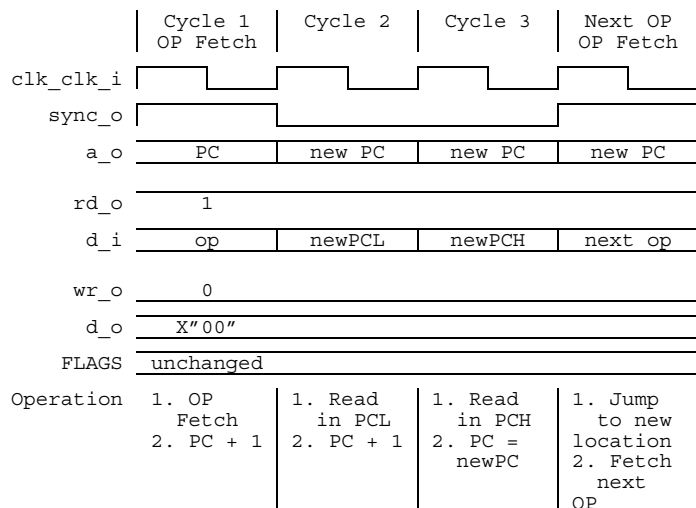


Figure (36): JMP – Timing Diagram “Absolute”

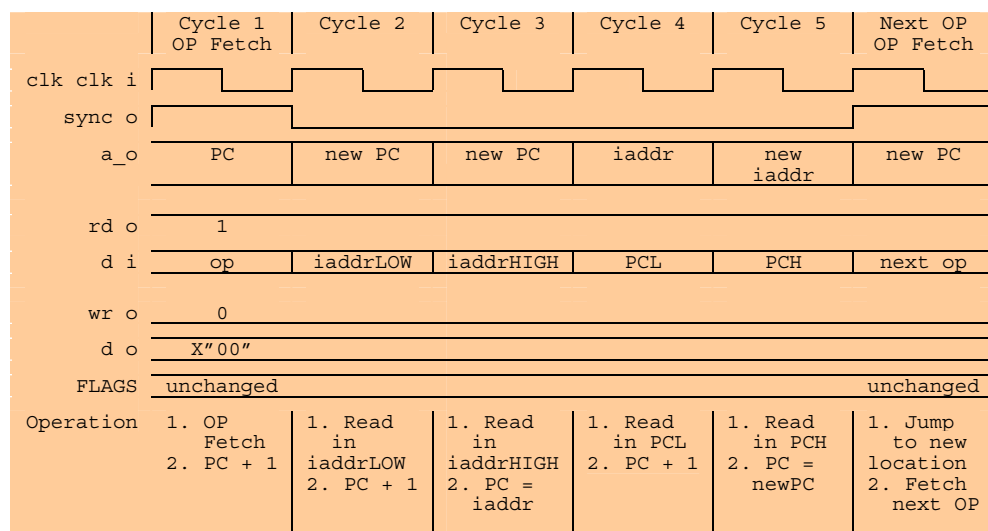


Figure (37): JMP – Timing Diagram “Indirect” (must be corrected)

## JSR

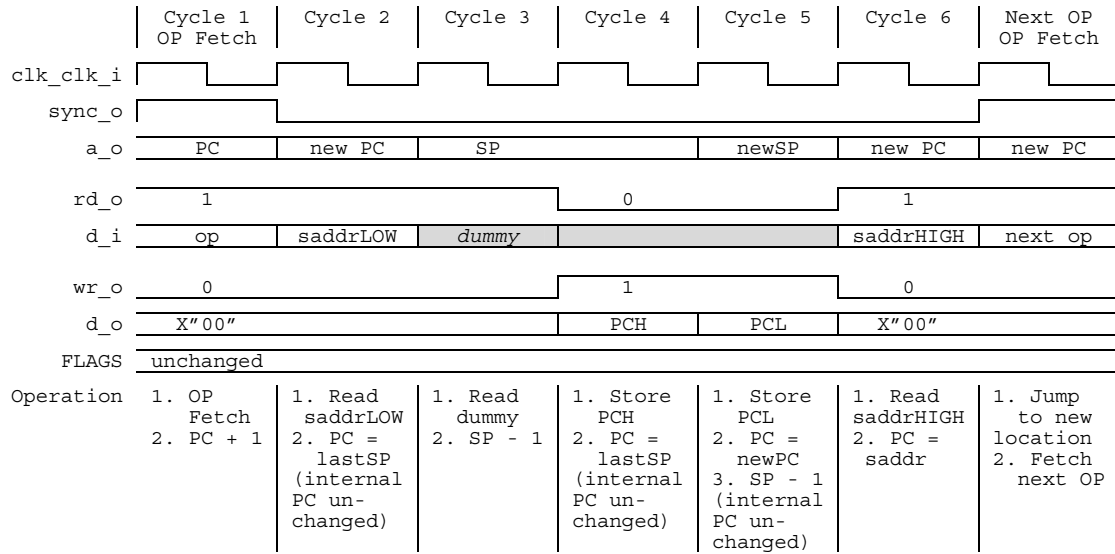


Figure (38): JSR – Timing Diagram

## NOP, TXS

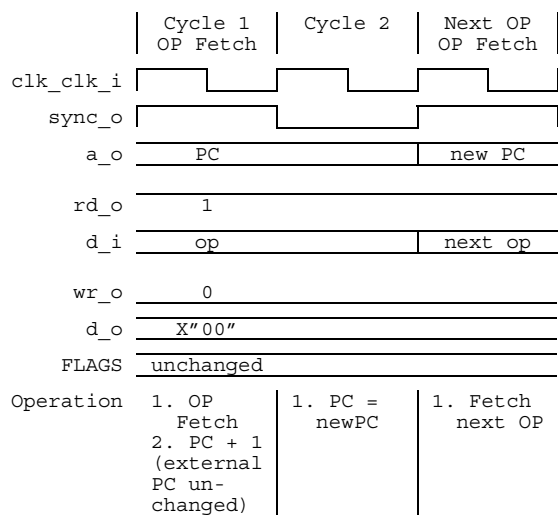


Figure (39): NOP, TXS – Timing Diagram

## PLA, PLP

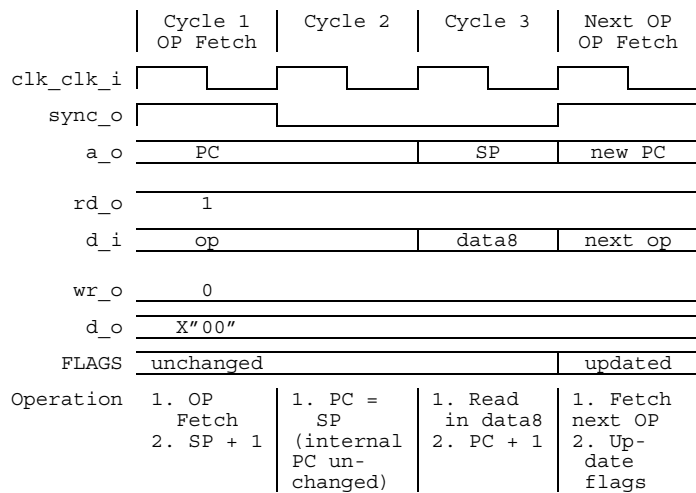


Figure (40): PLA, PLP – Timing Diagram “Implied”

## PHA, PHP

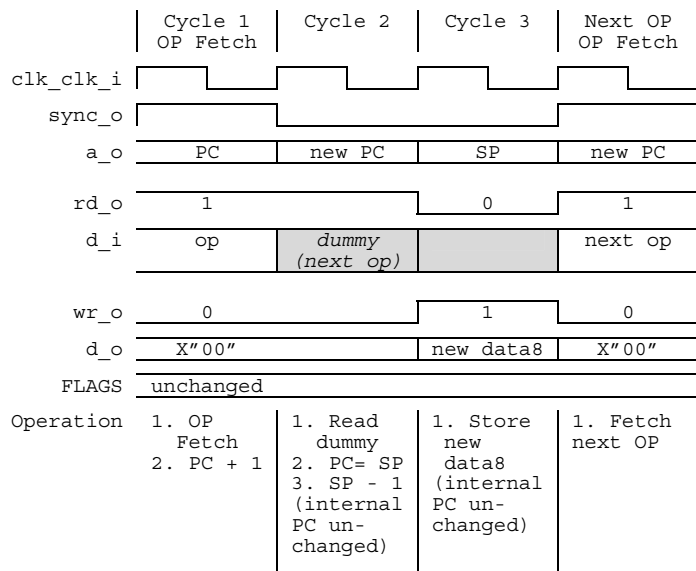


Figure (41): PHA, PHP – Timing Diagram “Implied”

## RTI

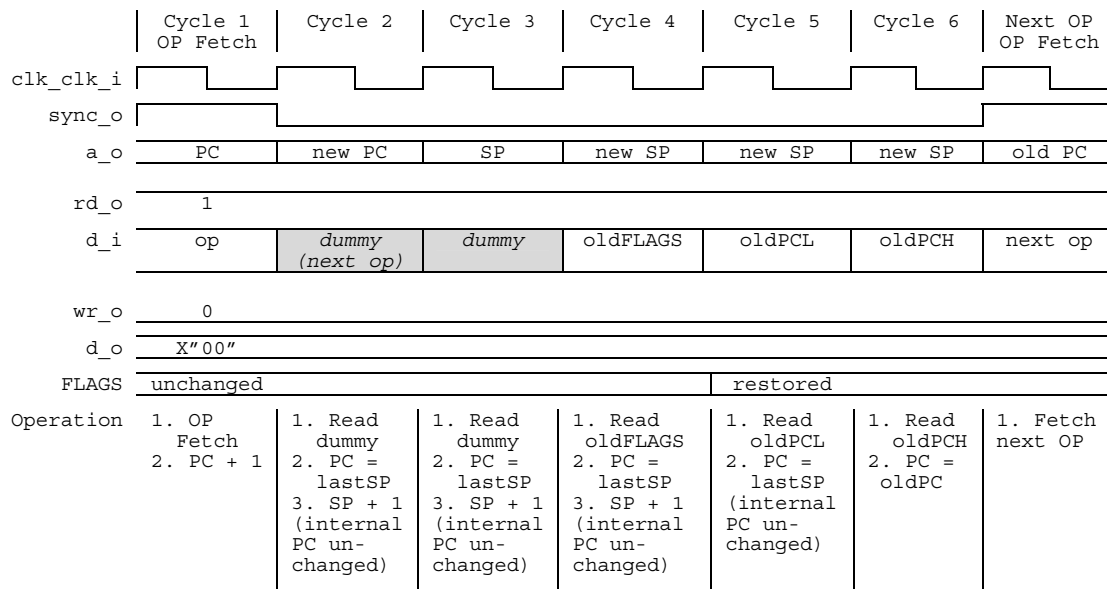


Figure (42): RTI – Timing Diagram “Implied”



## RTS

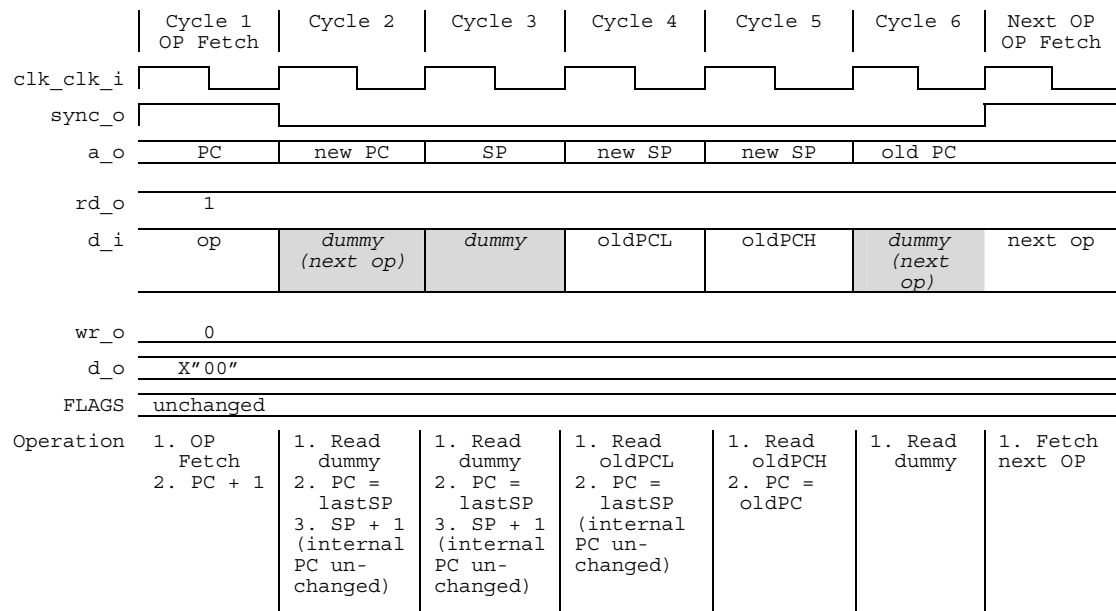


Figure (43): RTS – Timing Diagram “Implied”

## STA, STX, STY

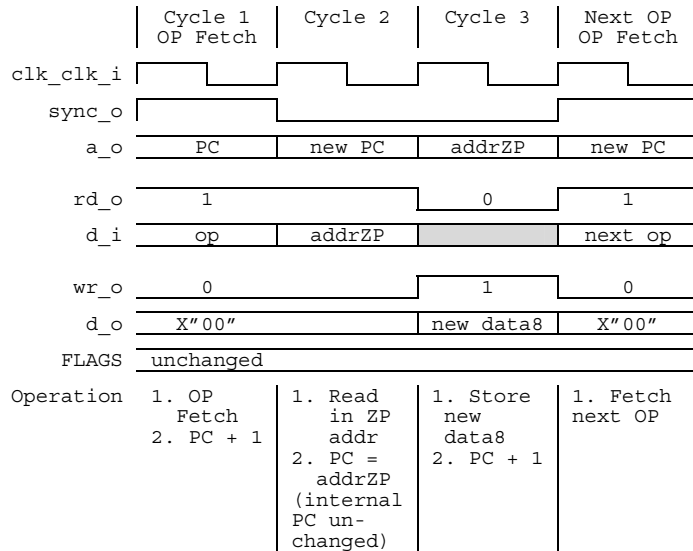


Figure (44): STA, STX, STY – Timing Diagram “Zero Page”

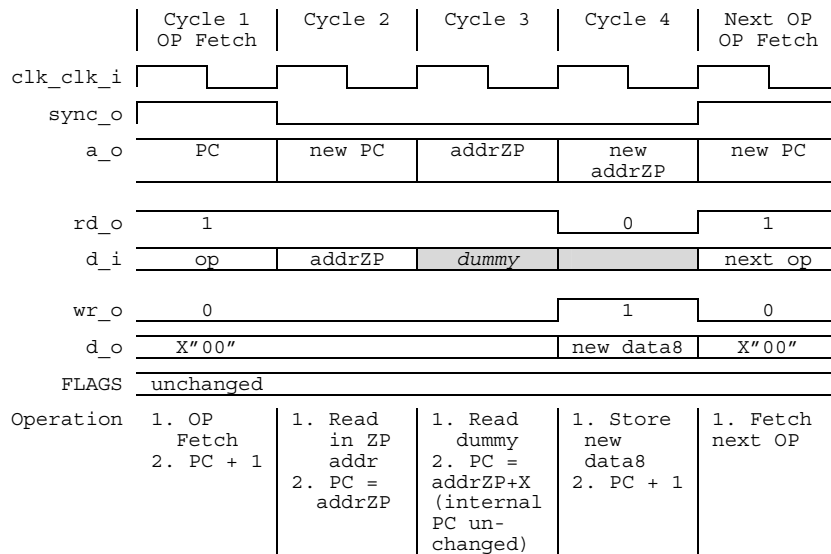


Figure (45): STA, STY – Timing Diagram “Zero Page, X”

## STA, STX, STY (cont.)

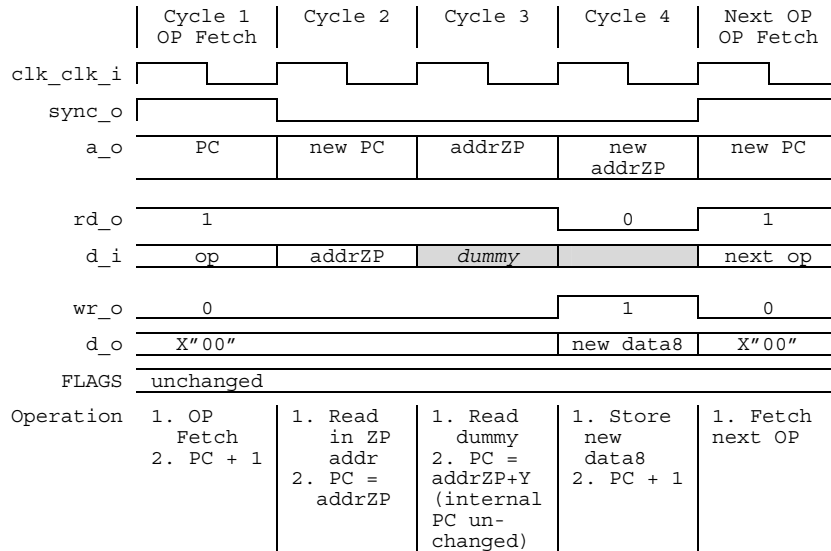


Figure (46): STX – Timing Diagram “Zero Page, Y”

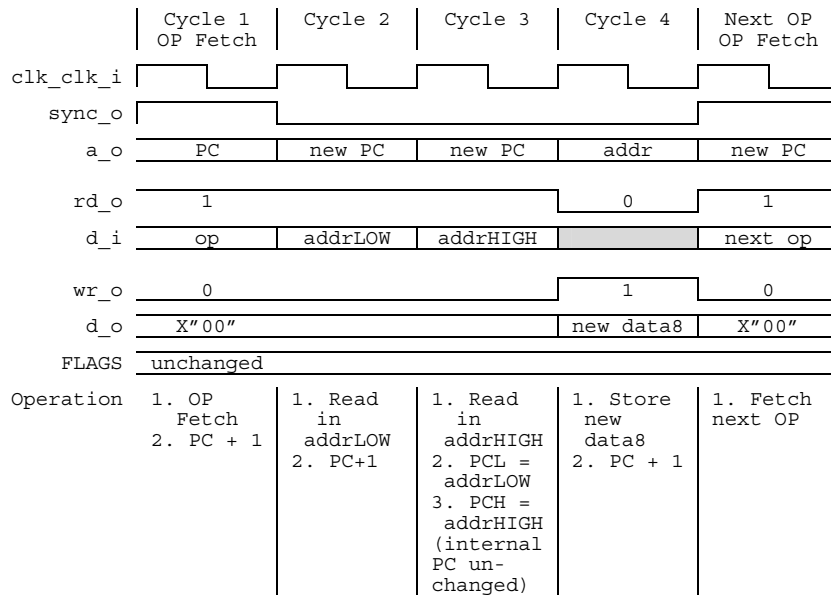


Figure (47): STA, STX, STY – Timing Diagram “Absolute”

## STA, STX, STY (cont.)

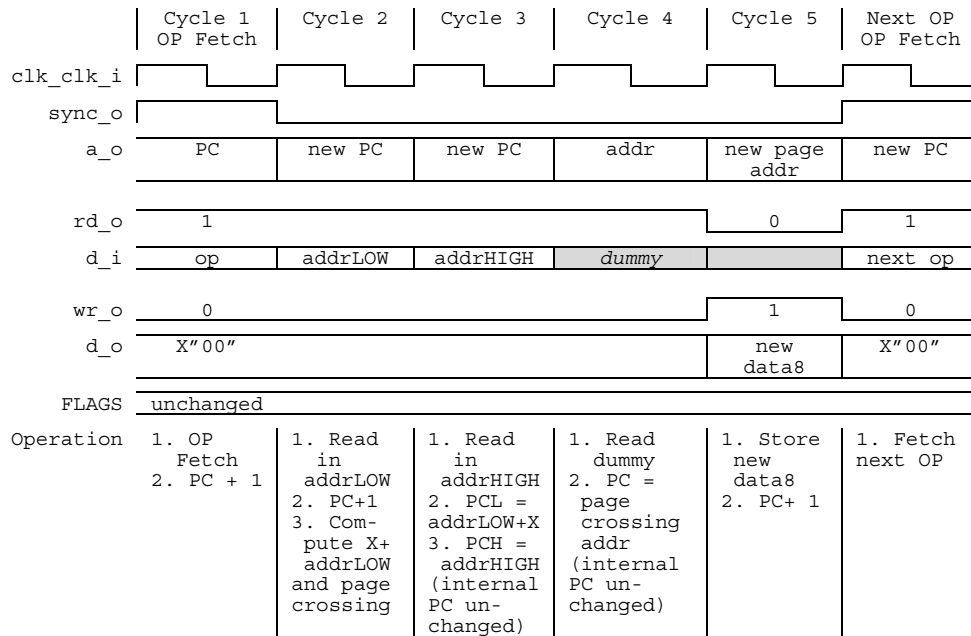


Figure (48): STA – Timing Diagram “Absolute, X”

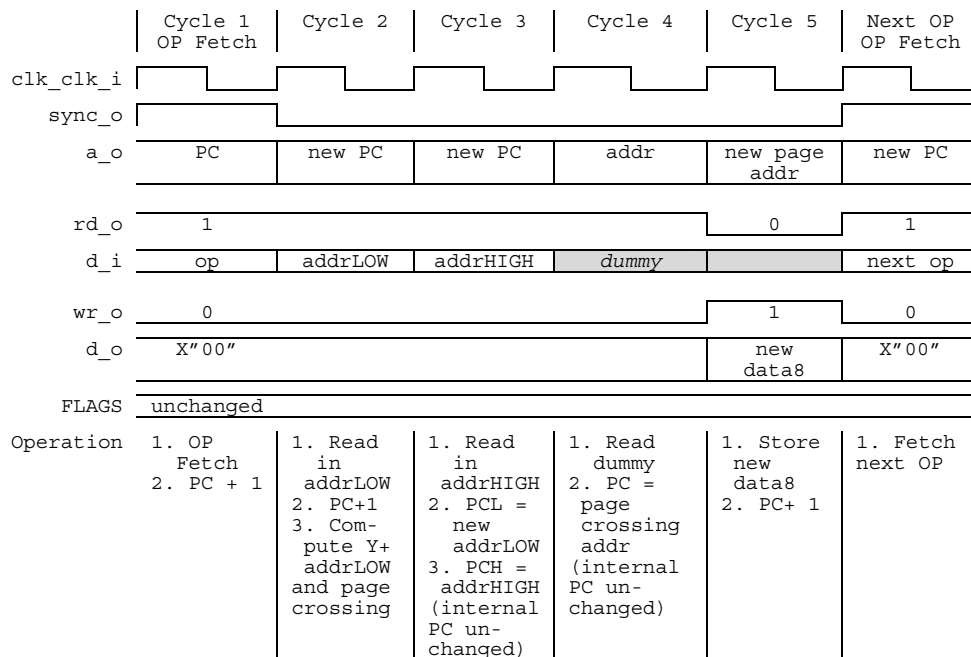


Figure (49): STA – Timing Diagram “Absolute, Y”

## STA, STX, STY (cont.)

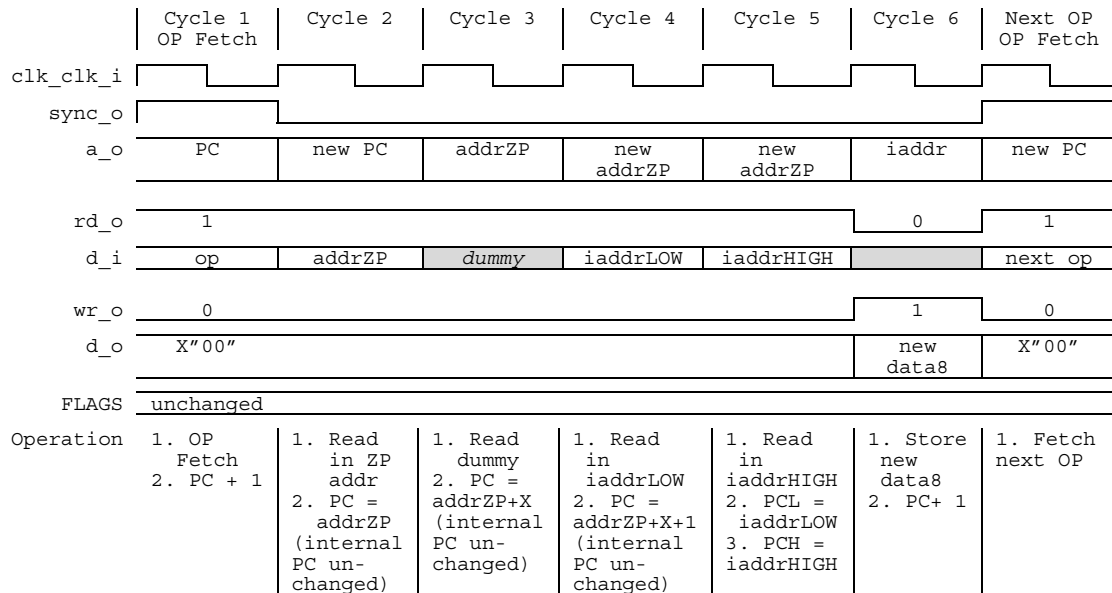


Figure (50): STA – Timing Diagram “(Indirect, X)”

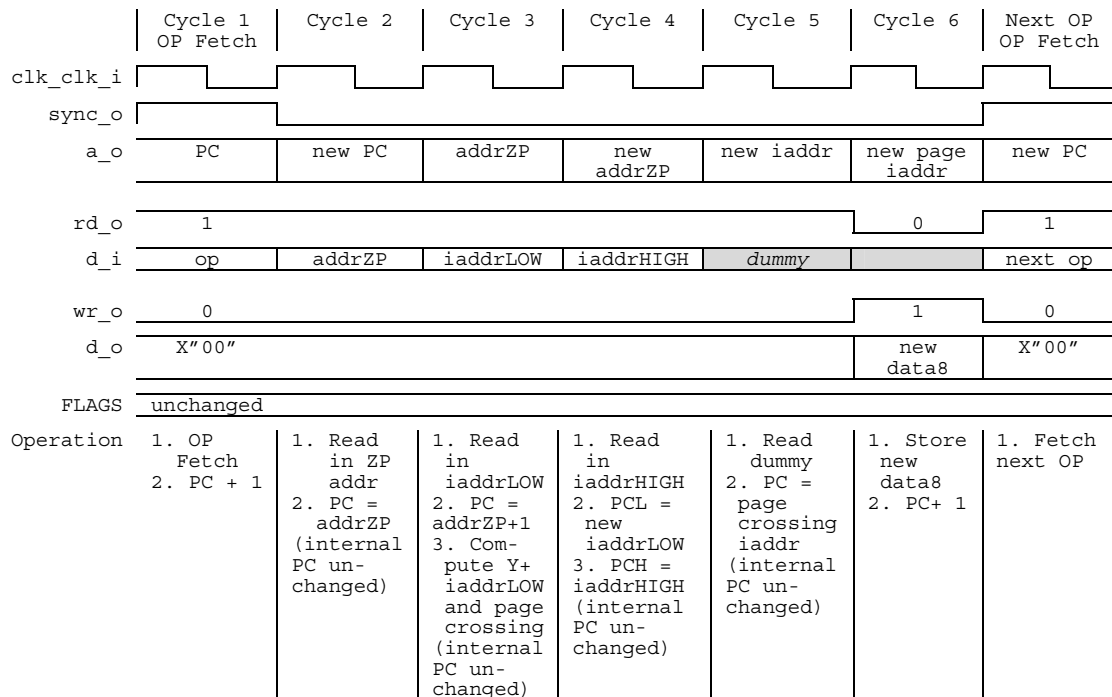


Figure (51): STA – Timing Diagram “(Indirect), Y”



# Instruction Table

| LSB |                                       |                                      |                     |   |                    |                                 |                    |                   |                       |                                    |                       |   |                                    |                                    |                                    |                                   |
|-----|---------------------------------------|--------------------------------------|---------------------|---|--------------------|---------------------------------|--------------------|-------------------|-----------------------|------------------------------------|-----------------------|---|------------------------------------|------------------------------------|------------------------------------|-----------------------------------|
|     | 0                                     | 1                                    | 2                   | 3 | 4                  | 5                               | 6                  | 7                 | 8                     | 9                                  | A                     | B | C                                  | D                                  | E                                  | F                                 |
| 0   | BRK<br>Implied<br>1 7                 | ORA<br>(IND,X)<br>2 6                |                     |   | TSB<br>ZP<br>2 5   | ORA<br>ZP<br>2 3                | ASL<br>ZP<br>2 5   | RMB0<br>ZP<br>2 5 | PHP<br>Implied<br>1 3 | ORA<br>IMM<br>2 2                  | ASL<br>Accum<br>1 2   |   | TSB<br>ABS<br>3 6                  | ORA<br>ABS<br>3 4                  | ASL<br>ABS<br>3 6                  | BBR0<br>ZP<br>3 5 <sup>one</sup>  |
| 1   | BPL<br>Relative<br>2 2 <sup>one</sup> | ORA<br>(IND),Y<br>2 5 <sup>one</sup> | ORA<br>(IND)<br>2 5 |   | TRB<br>ZP<br>2 5   | ORA<br>ZP,X<br>2 4              | ASL<br>ZP,X<br>2 6 | RMB1<br>ZP<br>2 5 | CLC<br>Implied<br>1 3 | ORA<br>ABS,Y<br>3 4 <sup>one</sup> | INC<br>Accum<br>1 2   |   | TRB<br>ABS<br>3 6                  | ORA<br>ABS,X<br>3 4 <sup>one</sup> | ASL<br>ABS,X<br>3 7                | BBR1<br>ZP<br>3 5 <sup>one</sup>  |
| 2   | JSR<br>ABS<br>3 6                     | AND<br>(IND,X)<br>2 6                |                     |   | BIT<br>ZP<br>2 3   | AND<br>ZP<br>2 3                | ROL<br>ZP<br>2 5   | RMB2<br>ZP<br>2 5 | PLP<br>Implied<br>1 4 | IMM<br>2 2                         | ROL<br>Accum<br>1 2   |   | BIT<br>ABS<br>3 4                  | AND<br>ABS<br>3 4                  | ROL<br>ABS<br>3 6                  | BBR2<br>ZP<br>3 5 <sup>one</sup>  |
| 3   | BMI<br>Relative<br>2 2 <sup>one</sup> | AND<br>(IND),Y<br>2 5 <sup>one</sup> | AND<br>(IND)<br>2 5 |   | BIT<br>ZP,X<br>2 4 | AND<br>ZP,X<br>2 4              | ROL<br>ZP,X<br>2 6 | RMB3<br>ZP<br>2 5 | SEC<br>Implied<br>1 3 | ABS,Y<br>2 4 <sup>one</sup>        | DEC<br>Accum<br>1 2   |   | BIT<br>ABS,X<br>3 4 <sup>one</sup> | AND<br>ABS,X<br>3 4 <sup>one</sup> | ROL<br>ABS,X<br>3 7                | BBR3<br>ZP<br>3 5 <sup>one</sup>  |
| 4   | RTI<br>Implied<br>1 6                 | EOR<br>(IND,X)<br>2 6                |                     |   |                    | EOR<br>ZP<br>2 3                | LSR<br>ZP<br>2 5   | RMB4<br>ZP<br>2 5 | PHA<br>Implied<br>1 3 | EOR<br>IMM<br>2 2                  | LSR<br>Accum<br>1 2   |   | JMP<br>ABS<br>3 3                  | EOR<br>ABS<br>3 4                  | LSR<br>ABS<br>3 6                  | BBR4<br>ZP<br>3 5 <sup>one</sup>  |
| 5   | BVC<br>Relative<br>2 2 <sup>one</sup> | EOR<br>(IND),Y<br>2 5 <sup>one</sup> | EOR<br>(IND)<br>2 5 |   |                    | EOR<br>ZP,X<br>2 4              | LSR<br>ZP,X<br>2 6 | RMB5<br>ZP<br>2 5 | CLI<br>Implied<br>1 3 | EOR<br>ABS,Y<br>3 4 <sup>one</sup> | PHY<br>Implied<br>1 3 |   |                                    | EOR<br>ABS,X<br>3 4 <sup>one</sup> | LSR<br>ABS,X<br>3 7                | BBR5<br>ZP<br>3 5 <sup>one</sup>  |
| 6   | RTS<br>Implied<br>1 6                 | ADC<br>(IND,X)<br>2 6                |                     |   | STZ<br>ZP<br>2 3   | ADC<br>ZP<br>2 3 <sup>+</sup>   | ROR<br>ZP<br>2 5   | RMB6<br>ZP<br>2 5 | PLA<br>Implied<br>1 4 | ADC<br>IMM<br>2 2 <sup>+</sup>     | ROR<br>Accum<br>1 2   |   | JMP<br>(ABS)<br>3 6                | ADC<br>ABS<br>3 4 <sup>+</sup>     | ROR<br>ABS<br>3 6                  | BBR6<br>ZP<br>3 5 <sup>one</sup>  |
| 7   | BVS<br>Relative<br>2 2 <sup>one</sup> | ADC<br>(IND),Y<br>2 5 <sup>one</sup> | ADC<br>(IND)<br>2 5 |   | STZ<br>ZP,X<br>2 4 | ADC<br>ZP,X<br>2 4 <sup>+</sup> | ROR<br>ZP,X<br>2 6 | RMB7<br>ZP<br>2 5 | SEI<br>Implied<br>1 3 | ADC<br>3 4 <sup>one</sup>          | PLY<br>Implied<br>1 4 |   | JMP<br>(ABS,X)<br>3 6              | ADC<br>ABS,X<br>3 4 <sup>one</sup> | ROR<br>ABS,X<br>3 7                | BBR7<br>ZP<br>3 5 <sup>one</sup>  |
| 8   | BRA<br>Relative<br>2 3 <sup>one</sup> | STA<br>(IND,X)<br>2 6                |                     |   | STY<br>ZP<br>2 3   | STA<br>ZP<br>2 3                | STX<br>ZP<br>2 3   | SNB0<br>ZP<br>2 5 | DEY<br>Implied<br>1 3 | BIT<br>IMM<br>2 2                  | TXA<br>Implied<br>1 2 |   | STY<br>ABS<br>3 4                  | STA<br>ABS<br>3 4                  | STX<br>ABS<br>3 4                  | BBR8<br>ZP<br>3 5 <sup>one</sup>  |
| 9   | BCC<br>Relative<br>2 2 <sup>one</sup> | STA<br>(IND),Y<br>2 6                | STA<br>(IND)<br>2 5 |   | STY<br>ZP,X<br>2 4 | STA<br>ZP,X<br>2 4              | STX<br>ZP,Y<br>2 4 | SNB1<br>ZP<br>2 5 | TYA<br>Implied<br>1 3 | STA<br>ABS,Y<br>3 5                | TXS<br>Implied<br>1 2 |   | STZ<br>ABS<br>3 4                  | STA<br>ABS,X<br>3 5                | STZ<br>ABS,X<br>3 5                | BBR9<br>ZP<br>3 5 <sup>one</sup>  |
| A   | LDY<br>IMM<br>2 2                     | LDA<br>(IND,X)<br>2 6                | LDX<br>IMM<br>2 2   |   | LDY<br>ZP<br>2 3   | LDA<br>ZP<br>2 3                | LDX<br>ZP<br>2 3   | SNB2<br>ZP<br>2 5 | TAY<br>Implied<br>1 3 | IMM<br>2 2                         | TAX<br>Implied<br>1 2 |   | LDY<br>ABS<br>3 4                  | LDA<br>ABS<br>3 4                  | LDX<br>ABS<br>3 4                  | BBR10<br>ZP<br>3 5 <sup>one</sup> |
| B   | BCS<br>Relative<br>2 2 <sup>one</sup> | LDA<br>(IND),Y<br>2 5 <sup>one</sup> | LDA<br>(IND)<br>2 5 |   | LDY<br>ZP,X<br>2 4 | LDA<br>ZP,X<br>2 4              | LDX<br>ZP,Y<br>2 4 | SNB3<br>ZP<br>2 5 | CLV<br>Implied<br>1 3 | LDA<br>3 4 <sup>one</sup>          | TSX<br>Implied<br>1 2 |   | LDY<br>ABS,X<br>3 4 <sup>one</sup> | LDA<br>ABS,X<br>3 4 <sup>one</sup> | LDX<br>ABS,X<br>3 4 <sup>one</sup> | BBR11<br>ZP<br>3 5 <sup>one</sup> |
| C   | CPY<br>IMM<br>2 2                     | CMP<br>(IND,X)<br>2 6                |                     |   | CPY<br>ZP<br>2 3   | CMP<br>ZP<br>2 3                | DEC<br>ZP<br>2 5   | SNB4<br>ZP<br>2 5 | INY<br>Implied<br>1 3 | CMP<br>IMM<br>2 2                  | DEX<br>Implied<br>1 2 |   | CPY<br>ABS<br>3 4                  | CMP<br>ABS<br>3 4                  | DEC<br>ABS<br>3 6                  | BBR12<br>ZP<br>3 5 <sup>one</sup> |
| D   | BNE<br>Relative<br>2 2 <sup>one</sup> | CMP<br>(IND),Y<br>2 5 <sup>one</sup> | CMP<br>(IND)<br>2 5 |   |                    | CMP<br>ZP,X<br>2 4              | DEC<br>ZP,X<br>2 6 | SNB5<br>ZP<br>2 5 | CLD<br>Implied<br>1 3 | CMP<br>ABS,Y<br>3 4 <sup>one</sup> | PHX<br>Implied<br>1 3 |   |                                    | CMP<br>ABS,X<br>3 4 <sup>one</sup> | DEC<br>ABS,X<br>3 7                | BBR13<br>ZP<br>3 5 <sup>one</sup> |
| E   | CPX<br>IMM<br>2 2                     | SBC<br>(IND,X)<br>2 6 <sup>+</sup>   |                     |   | CPX<br>ZP<br>2 3   | SBC<br>ZP<br>2 3 <sup>+</sup>   | INC<br>ZP<br>2 5   | SNB6<br>ZP<br>2 5 | INX<br>Implied<br>1 3 | SBC<br>IMM<br>2 2 <sup>+</sup>     | NOP<br>Implied<br>1 2 |   | CPX<br>ABS<br>3 4                  | SBC<br>ABS<br>3 4 <sup>+</sup>     | INC<br>ABS<br>3 6                  | BBR14<br>ZP<br>3 5 <sup>one</sup> |
| F   | BEQ<br>Relative<br>2 2 <sup>one</sup> | SBC<br>(IND),Y<br>2 5 <sup>one</sup> | SBC<br>(IND)<br>2 5 |   |                    | SBC<br>ZP,X<br>2 4 <sup>+</sup> | INC<br>ZP,X<br>2 6 | SNB7<br>ZP<br>2 5 | SED<br>Implied<br>1 3 | SBC<br>ABS,Y<br>3 4 <sup>one</sup> | PLX<br>Implied<br>1 4 |   |                                    | SBC<br>3 4 <sup>one</sup>          | INC<br>ABS,X<br>3 7                | BBR15<br>ZP<br>3 5 <sup>one</sup> |
|     | 0                                     | 1                                    | 2                   | 3 | 4                  | 5                               | 6                  | 7                 | 8                     | 9                                  | A                     | B | C                                  | D                                  | E                                  | F                                 |

+ Add 1 to N if in decimal mode, \* ADD 1 to N if page boundary is crossed  
\*\*\* Add 1 to N if branch occurs to same page - Add 2 to N if branch occurs to different page

# Index

---

This section contains an alphabetical list of helpful document entries with their corresponding page numbers.