

sim6502 - 6502 Assembly Testing Framework

Barry Walker

January 28, 2026

Contents

1	Introduction	2
2	Processor Selection	2
2.1	Specifying Processor Type	2
2.2	65C02 Additional Instructions	2
2.2.1	Stack Operations	3
2.2.2	Store Zero	3
2.2.3	Branch Always	3
2.2.4	Accumulator Increment/Decrement	3
2.2.5	Bit Manipulation	3
2.2.6	Zero Page Indirect Addressing	3
2.3	6510 I/O Port	4
2.3.1	C64 Memory Banking	4
2.4	Processor Examples	4
2.4.1	6502 Example	4
2.4.2	6510 Example	4
2.4.3	65C02 Example	5
3	Test Suite Structure	5
4	License	5

1 Introduction

sim6502 is a tool to help you unit test your 6502 assembly language programs. It works by running your assembled programs with a 6502 simulator and then allowing you to make assertions on memory and CPU state.

2 Processor Selection

sim6502 supports multiple processor variants in the 65xx family:

- **6502** - Original NMOS MOS Technology 6502 (default)
- **6510** - Same ISA as 6502, adds I/O port at \$00-\$01 (Commodore 64)
- **65C02** - WDC CMOS variant with additional instructions

2.1 Specifying Processor Type

Use the `processor()` statement at the beginning of a suite:

```
suites {  
  suite("My 65C02 Tests") {  
    processor(65c02)  
  
    test("test-1", "Uses 65C02 opcodes") {  
      // PHX, PLX, STZ, BRA, etc. available  
    }  
  }  
}
```

If no `processor()` statement is specified, the default is 6502.

2.2 65C02 Additional Instructions

The 65C02 adds the following instructions not available on the 6502/6510:

- **PHX, PLX** - Push/Pull X register
- **PHY, PLY** - Push/Pull Y register
- **STZ** - Store Zero
- **BRA** - Branch Always
- **INC A, DEC A** - Increment/Decrement Accumulator
- **TRB, TSB** - Test and Reset/Set Bits
- Zero Page Indirect addressing mode: **LDA** (\$50)

2.2.1 Stack Operations

The 65C02 adds dedicated stack operations for the X and Y registers:

```
PHX      ; Push X register onto stack
PLX      ; Pull X register from stack
PHY      ; Push Y register onto stack
PLY      ; Pull Y register from stack
```

These instructions behave like PHA/PLA but operate on the X and Y registers respectively.

2.2.2 Store Zero

The STZ instruction stores zero to a memory location, supporting multiple addressing modes:

```
STZ $50      ; Zero page
STZ $50,X    ; Zero page, X
STZ $1234    ; Absolute
STZ $1234,X  ; Absolute, X
```

2.2.3 Branch Always

The BRA instruction provides an unconditional relative branch:

```
BRA label      ; Always branch to label
```

This is more efficient than using a conditional branch that always succeeds.

2.2.4 Accumulator Increment/Decrement

The 65C02 adds INC and DEC instructions that operate directly on the accumulator:

```
INC A      ; Increment accumulator
DEC A      ; Decrement accumulator
```

2.2.5 Bit Manipulation

TRB and TSB provide efficient bit testing and manipulation:

```
TRB $50      ; Test and Reset Bits (clear bits where A has 1s)
TSB $50      ; Test and Set Bits (set bits where A has 1s)
```

Both instructions set the Zero flag based on the result of A AND memory (before modification).

2.2.6 Zero Page Indirect Addressing

The 65C02 adds a new addressing mode that uses zero page indirect addressing without an index register:

```
LDA ($50)    ; Load A from address pointed to by $50-$51
STA ($50)    ; Store A to address pointed to by $50-$51
```

This mode is available for: LDA, STA, ORA, AND, EOR, ADC, CMP, and SBC.

2.3 6510 I/O Port

The 6510 (used in the Commodore 64) has an I/O port at addresses \$00 and \$01:

- **\$00** - Data Direction Register (DDR)
- **\$01** - Data Port

When using `processor(6510)`, reads and writes to these addresses access the I/O port registers instead of RAM. The DDR controls which bits are inputs (0) or outputs (1).

2.3.1 C64 Memory Banking

On the Commodore 64, the 6510 I/O port controls memory banking:

```
; Set DDR - bits 0-5 as outputs
LDA #$2F
STA $00

; Set data port - standard C64 configuration
LDA #$37
STA $01
```

2.4 Processor Examples

2.4.1 6502 Example

```
suites {
  suite("Standard 6502 Tests") {
    processor(6502) // Optional - this is the default

    test("basic-test", "Basic 6502 operation") {
      a = $42
      assert(a == $42, "Accumulator test")
    }
  }
}
```

2.4.2 6510 Example

```
suites {
  suite("C64 6510 Tests") {
    processor(6510)

    test("io-port", "Test 6510 I/O port") {
      $00 = $2F // Set DDR
      $01 = $37 // Set data port (C64 standard)
      assert($00 == $2F, "DDR configured")
      assert($01 == $37, "Port configured")
    }
  }
}
```

2.4.3 65C02 Example

```
suites {
  suite("WDC 65C02 Tests") {
    processor(65c02)

    test("enhanced-ops", "Test 65C02 enhancements") {
      a = $41
      x = $10
      y = $20

      // Use 65C02-specific instructions
      // These would be assembled in your .prg file

      assert(a == $42, "INC A worked")
      assert(x == $11, "PHX/PLX preserved X")
    }
  }
}
```

3 Test Suite Structure

For information on the complete DSL syntax, test structure, and other features, please refer to the project README.md file.

4 License

The 6502 Unit Test CLI and associated test suite are Copyright © 2020 by Barry Walker. All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.