Obelisk 6502 Guide

***Introduction***

My first computer, a BBC Microcomputer Model B, was powered by a 6502 processor and I spent many hours writing and debugging assembly language programs for it (I was the author of CodeKit, co-author of SupaStore and lots of other stuff). I managed to hold on to my BBC until 1989 when my wife insisted that I got rid of it before I bought my first Intel based PC.

Sometime in 1998 I found an emulator called 'pcBBC' on the web. It was very good BBC emulator and it reawakened my interest in 6502 assembly programming. Unfortunate pcBBC did not survive the various updates to Microsoft Windows and now I use BeebEm.

The assembler that I started with ([AS65](https://www.nesdev.org/obelisk-6502-guide/downloads.html#tools)) runs under DOS (or in a DOS window) and can assemble a 16Kb ROM almost instantly (on a Pentium III 500Mhz). When I was developing on my BBC I used to break for a cup of tea (or two) when the assembler was running!

More recently I have written my own Java based 6502 assembler which supports relocatable code and some simple structured programming constructs (e.g. if/then/else).

**Contents**

Originally this was all in one HTML file but it grew too big for any of my editors to handle without crashing so I had to cut it up into 8 sections.

* [Architecture](https://www.nesdev.org/obelisk-6502-guide/architecture.html) describes the few basic details of the processor.
* [Registers](https://www.nesdev.org/obelisk-6502-guide/registers.html) goes over each of the internal registers and their use.
* [Instructions](https://www.nesdev.org/obelisk-6502-guide/instructions.html) gives a summary of whole instruction set.
* [Addressing](https://www.nesdev.org/obelisk-6502-guide/addressing.html) describes each of the 6502 memory addressing modes.
* [Algorithms](https://www.nesdev.org/obelisk-6502-guide/algorithms.html) contains examples of basic 6502 coding.
* [Reference](https://www.nesdev.org/obelisk-6502-guide/reference.html) describes the complete instruction set in detail.
* [Downloads](https://www.nesdev.org/obelisk-6502-guide/downloads.html) contains some files useful for 6502 programming.
* [Links](https://www.nesdev.org/obelisk-6502-guide/links.html) contains some links to other sites worth visiting.

Click on any of the above links or the links at the bottom of each page to move between the sections.

## Basic Architecture

The 6502 microprocessor is a relatively simple 8 bit CPU with only a few internal registers capable of addressing at most 64Kb of memory via its 16 bit address bus. The processor is little endian and expects addresses to be stored in memory least significant byte first.

The first 256 byte page of memory ($0000-$00FF) is referred to as 'Zero Page' and is the focus of a number of special addressing modes that result in shorter (and quicker) instructions or allow indirect access to the memory. The second page of memory ($0100-$01FF) is reserved for the system stack and which cannot be relocated.

The only other reserved locations in the memory map are the very last 6 bytes of memory $FFFA to $FFFF which must be programmed with the addresses of the non-maskable interrupt handler ($FFFA/B), the power on reset location ($FFFC/D) and the BRK/interrupt request handler ($FFFE/F) respectively.

The 6502 does not have any special support of hardware devices so they must be mapped to regions of memory in order to exchange data with the hardware latches.

## The Registers

The 6502 has only a small number of registers compared to other processor of the same era. This makes it especially challenging to program as algorithms must make efficient use of both registers and memory.

### Program Counter

The program counter is a 16 bit register which points to the next instruction to be executed. The value of program counter is modified automatically as instructions are executed.

The value of the program counter can be modified by executing a jump, a relative branch or a subroutine call to another memory address or by returning from a subroutine or interrupt.

### Stack Pointer

The processor supports a 256 byte stack located between $0100 and $01FF. The stack pointer is an 8 bit register and holds the low 8 bits of the next free location on the stack. The location of the stack is fixed and cannot be moved.

Pushing bytes to the stack causes the stack pointer to be decremented. Conversely pulling bytes causes it to be incremented.

The CPU does not detect if the stack is overflowed by excessive pushing or pulling operations and will most likely result in the program crashing.

### Accumulator

The 8 bit accumulator is used all arithmetic and logical operations (with the exception of increments and decrements). The contents of the accumulator can be stored and retrieved either from memory or the stack.

Most complex operations will need to use the accumulator for arithmetic and efficient optimisation of its use is a key feature of time critical routines.

### Index Register X

The 8 bit index register is most commonly used to hold counters or offsets for accessing memory. The value of the X register can be loaded and saved in memory, compared with values held in memory or incremented and decremented.

The X register has one special function. It can be used to get a copy of the stack pointer or change its value.

### Index Register Y

The Y register is similar to the X register in that it is available for holding counter or offsets memory access and supports the same set of memory load, save and compare operations as wells as increments and decrements. It has no special functions.

### Processor Status

As instructions are executed a set of processor flags are set or clear to record the results of the operation. This flags and some additional control flags are held in a special status register. Each flag has a single bit within the register.

Instructions exist to test the values of the various bits, to set or clear some of them and to push or pull the entire set to or from the stack.

* Carry Flag

The carry flag is set if the last operation caused an overflow from bit 7 of the result or an underflow from bit 0. This condition is set during arithmetic, comparison and during logical shifts. It can be explicitly set using the 'Set Carry Flag' ([SEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEC)) instruction and cleared with 'Clear Carry Flag' ([CLC](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLC)).

* Zero Flag

The zero flag is set if the result of the last operation as was zero.

* Interrupt Disable

The interrupt disable flag is set if the program has executed a 'Set Interrupt Disable' ([SEI](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEI)) instruction. While this flag is set the processor will not respond to interrupts from devices until it is cleared by a 'Clear Interrupt Disable' ([CLI](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLI)) instruction.

* Decimal Mode

While the decimal mode flag is set the processor will obey the rules of Binary Coded Decimal (BCD) arithmetic during addition and subtraction. The flag can be explicitly set using 'Set Decimal Flag' ([SED](https://www.nesdev.org/obelisk-6502-guide/reference.html#SED)) and cleared with 'Clear Decimal Flag' ([CLD](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLD)).

* Break Command

The break command bit is set when a [BRK](https://www.nesdev.org/obelisk-6502-guide/reference.html#BRK) instruction has been executed and an interrupt has been generated to process it.

* Overflow Flag

The overflow flag is set during arithmetic operations if the result has yielded an invalid 2's complement result (e.g. adding to positive numbers and ending up with a negative result: 64 + 64 => -128). It is determined by looking at the carry between bits 6 and 7 and between bit 7 and the carry flag.

* Negative Flag

The negative flag is set if the result of the last operation had bit 7 set to a one.

## ****The Instruction Set****

The 6502 has a relatively basic set of instructions, many having similar functions (e.g. memory access, arithmetic, etc.). The following sections list the complete set of 56 instructions in functional groups.

### Load/Store Operations

These instructions transfer a single byte between memory and one of the registers. Load operations set the negative ([N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N)) and zero ([Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z)) flags depending on the value of transferred. Store operations do not affect the flag settings.

|  |  |  |
| --- | --- | --- |
| [LDA](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDA) | Load Accumulator | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [LDX](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDX) | Load X Register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [LDY](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDY) | Load Y Register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [STA](https://www.nesdev.org/obelisk-6502-guide/reference.html#STA) | Store Accumulator |  |
| [STX](https://www.nesdev.org/obelisk-6502-guide/reference.html#STX) | Store X Register |  |
| [STY](https://www.nesdev.org/obelisk-6502-guide/reference.html#STY) | Store Y Register |  |

### Register Transfers

The contents of the X and Y registers can be moved to or from the accumulator, setting the negative ([N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N)) and zero ([Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z)) flags as appropriate.

|  |  |  |
| --- | --- | --- |
| [TAX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAX) | Transfer accumulator to X | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [TAY](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAY) | Transfer accumulator to Y | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [TXA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXA) | Transfer X to accumulator | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [TYA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TYA) | Transfer Y to accumulator | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |

### Stack Operations

The 6502 microprocessor supports a 256 byte stack fixed between memory locations $0100 and $01FF. A special 8-bit register, S, is used to keep track of the next free byte of stack space. Pushing a byte on to the stack causes the value to be stored at the current free location (e.g. $0100,S) and then the stack pointer is post decremented. Pull operations reverse this procedure.

The stack register can only be accessed by transferring its value to or from the X register. Its value is automatically modified by push/pull instructions, subroutine calls and returns, interrupts and returns from interrupts.

|  |  |  |
| --- | --- | --- |
| [TSX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TSX) | Transfer stack pointer to X | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [TXS](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXS) | Transfer X to stack pointer |  |
| [PHA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHA) | Push accumulator on stack |  |
| [PHP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHP) | Push processor status on stack |  |
| [PLA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLA) | Pull accumulator from stack | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [PLP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLP) | Pull processor status from stack | All |

### Logical

The following instructions perform logical operations on the contents of the accumulator and another value held in memory. The BIT instruction performs a logical AND to test the presence of bits in the memory value to set the flags but does not keep the result.

|  |  |  |
| --- | --- | --- |
| [AND](https://www.nesdev.org/obelisk-6502-guide/reference.html#AND) | Logical AND | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [EOR](https://www.nesdev.org/obelisk-6502-guide/reference.html#EOR) | Exclusive OR | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [ORA](https://www.nesdev.org/obelisk-6502-guide/reference.html#ORA) | Logical Inclusive OR | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [BIT](https://www.nesdev.org/obelisk-6502-guide/reference.html#BIT) | Bit Test | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |

### Arithmetic

The arithmetic operations perform addition and subtraction on the contents of the accumulator. The compare operations allow the comparison of the accumulator and X or Y with memory values.

|  |  |  |
| --- | --- | --- |
| [ADC](https://www.nesdev.org/obelisk-6502-guide/reference.html#ADC) | Add with Carry | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [SBC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SBC) | Subtract with Carry | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [CMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#CMP) | Compare accumulator | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [CPX](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPX) | Compare X register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [CPY](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPY) | Compare Y register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |

### Increments & Decrements

Increment or decrement a memory location or one of the X or Y registers by one setting the negative ([N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N)) and zero ([Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z)) flags as appropriate,

|  |  |  |
| --- | --- | --- |
| [INC](https://www.nesdev.org/obelisk-6502-guide/reference.html#INC) | Increment a memory location | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [INX](https://www.nesdev.org/obelisk-6502-guide/reference.html#INX) | Increment the X register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [INY](https://www.nesdev.org/obelisk-6502-guide/reference.html#INY) | Increment the Y register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [DEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEC) | Decrement a memory location | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |
| [DEX](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEX) | Decrement the X register | [N](https://www.nesdev.org/obelisk-6502-guide/instructions.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/instructions.html#Z) |
| [DEY](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEY) | Decrement the Y register | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) |

### Shifts

Shift instructions cause the bits within either a memory location or the accumulator to be shifted by one bit position. The rotate instructions use the contents if the carry flag ([C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C)) to fill the vacant position generated by the shift and to catch the overflowing bit. The arithmetic and logical shifts shift in an appropriate 0 or 1 bit as appropriate but catch the overflow bit in the carry flag ([C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C)).

|  |  |  |
| --- | --- | --- |
| [ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL) | Arithmetic Shift Left | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR) | Logical Shift Right | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL) | Rotate Left | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR) | Rotate Right | [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N),[Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z),[C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |

### Jumps & Calls

The following instructions modify the program counter causing a break to normal sequential execution. The [JSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#JSR) instruction pushes the old [PC](https://www.nesdev.org/obelisk-6502-guide/registers.html#PC) onto the stack before changing it to the new location allowing a subsequent [RTS](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTS) to return execution to the instruction after the call.

|  |  |  |
| --- | --- | --- |
| [JMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#JMP) | Jump to another location |  |
| [JSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#JSR) | Jump to a subroutine |  |
| [RTS](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTS) | Return from subroutine |  |

### Branches

Branch instructions break the normal sequential flow of execution by changing the program counter if a specified condition is met. All the conditions are based on examining a single bit within the processor status.

|  |  |  |
| --- | --- | --- |
| [BCC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCC) | Branch if carry flag clear |  |
| [BCS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCS) | Branch if carry flag set |  |
| [BEQ](https://www.nesdev.org/obelisk-6502-guide/reference.html#BEQ) | Branch if zero flag set |  |
| [BMI](https://www.nesdev.org/obelisk-6502-guide/reference.html#BMI) | Branch if negative flag set |  |
| [BNE](https://www.nesdev.org/obelisk-6502-guide/reference.html#BNE) | Branch if zero flag clear |  |
| [BPL](https://www.nesdev.org/obelisk-6502-guide/reference.html#BPL) | Branch if negative flag clear |  |
| [BVC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVC) | Branch if overflow flag clear |  |
| [BVS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVS) | Branch if overflow flag set |  |

Branch instructions use relative address to identify the target instruction if they are executed. As relative addresses are stored using a signed 8 bit byte the target instruction must be within 126 bytes before the branch or 128 bytes after the branch.

### Status Flag Changes

The following instructions change the values of specific status flags.

|  |  |  |
| --- | --- | --- |
| [CLC](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLC) | Clear carry flag | [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [CLD](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLD) | Clear decimal mode flag | [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) |
| [CLI](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLI) | Clear interrupt disable flag | [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) |
| [CLV](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLV) | Clear overflow flag | [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) |
| [SEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEC) | Set carry flag | [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) |
| [SED](https://www.nesdev.org/obelisk-6502-guide/reference.html#SED) | Set decimal mode flag | [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) |
| [SEI](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEI) | Set interrupt disable flag | [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) |

### System Functions

The remaining instructions perform useful but rarely used functions.

|  |  |  |
| --- | --- | --- |
| [BRK](https://www.nesdev.org/obelisk-6502-guide/reference.html#BRK) | Force an interrupt | [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) |
| [NOP](https://www.nesdev.org/obelisk-6502-guide/reference.html#NOP) | No Operation |  |
| [RTI](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTI) | Return from Interrupt | All |

## Addressing Modes

The 6502 processor provides several ways in which memory locations can be addressed. Some instructions support several different modes while others may only support one. In addition the two index registers can not always be used interchangeably. This lack of orthogonality in the instruction set is one of the features that makes the 6502 trickier to program well.

### Implicit

For many 6502 instructions the source and destination of the information to be manipulated is implied directly by the function of the instruction itself and no further operand needs to be specified. Operations like 'Clear Carry Flag' ([CLC](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLC)) and 'Return from Subroutine' ([RTS](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTS)) are implicit.

### Accumulator

Some instructions have an option to operate directly upon the accumulator. The programmer specifies this by using a special operand value, 'A'. For example:

LSR A ;Logical shift right one bit

ROR A ;Rotate right one bit

### Immediate

Immediate addressing allows the programmer to directly specify an 8 bit constant within the instruction. It is indicated by a '#' symbol followed by an numeric expression. For example:

LDA #10 ;Load 10 ($0A) into the accumulator

LDX #LO LABEL ;Load the LSB of a 16 bit address into X

LDY #HI LABEL ;Load the MSB of a 16 bit address into Y

### Zero Page

An instruction using zero page addressing mode has only an 8 bit address operand. This limits it to addressing only the first 256 bytes of memory (e.g. $0000 to $00FF) where the most significant byte of the address is always zero. In zero page mode only the least significant byte of the address is held in the instruction making it shorter by one byte (important for space saving) and one less memory fetch during execution (important for speed).

An assembler will automatically select zero page addressing mode if the operand evaluates to a zero page address and the instruction supports the mode (not all do).

LDA $00 ;Load accumulator from $00

ASL ANSWER ;Shift labelled location ANSWER left

### Zero Page,X

The address to be accessed by an instruction using indexed zero page addressing is calculated by taking the 8 bit zero page address from the instruction and adding the current value of the X register to it. For example if the X register contains $0F and the instruction LDA $80,X is executed then the accumulator will be loaded from $008F (e.g. $80 + $0F => $8F).

**NB**:  
The address calculation wraps around if the sum of the base address and the register exceed $FF. If we repeat the last example but with $FF in the X register then the accumulator will be loaded from $007F (e.g. $80 + $FF => $7F) and not $017F.

STY $10,X ;Save the Y register at location on zero page

AND TEMP,X ;Logical AND accumulator with a zero page value

### Zero Page,Y

The address to be accessed by an instruction using indexed zero page addressing is calculated by taking the 8 bit zero page address from the instruction and adding the current value of the Y register to it. This mode can only be used with the [LDX](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDX) and [STX](https://www.nesdev.org/obelisk-6502-guide/reference.html#STX) instructions.

LDX $10,Y ;Load the X register from a location on zero page

STX TEMP,Y ;Store the X register in a location on zero page

### Relative

Relative addressing mode is used by branch instructions (e.g. [BEQ](https://www.nesdev.org/obelisk-6502-guide/reference.html#BEQ), [BNE](https://www.nesdev.org/obelisk-6502-guide/reference.html#BNE), etc.) which contain a signed 8 bit relative offset (e.g. -128 to +127) which is added to program counter if the condition is true. As the program counter itself is incremented during instruction execution by two the effective address range for the target instruction must be with -126 to +129 bytes of the branch.

BEQ LABEL ;Branch if zero flag set to LABEL

BNE \*+4 ;Skip over the following 2 byte instruction

### Absolute

Instructions using absolute addressing contain a full 16 bit address to identify the target location.

JMP $1234 ;Jump to location $1234

JSR WIBBLE ;Call subroutine WIBBLE

### Absolute,X

The address to be accessed by an instruction using X register indexed absolute addressing is computed by taking the 16 bit address from the instruction and added the contents of the X register. For example if X contains $92 then an STA $2000,X instruction will store the accumulator at $2092 (e.g. $2000 + $92).

STA $3000,X ;Store accumulator between $3000 and $30FF

ROR CRC,X ;Rotate right one bit

### Absolute,Y

The Y register indexed absolute addressing mode is the same as the previous mode only with the contents of the Y register added to the 16 bit address from the instruction.

AND $4000,Y ;Perform a logical AND with a byte of memory

STA MEM,Y ;Store accumulator in memory

### Indirect

[JMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#JMP) is the only 6502 instruction to support indirection. The instruction contains a 16 bit address which identifies the location of the least significant byte of another 16 bit memory address which is the real target of the instruction.

For example if location $0120 contains $FC and location $0121 contains $BA then the instruction JMP ($0120) will cause the next instruction execution to occur at $BAFC (e.g. the contents of $0120 and $0121).

JMP ($FFFC) ;Force a power on reset

JMP (TARGET) ;Jump via a labelled memory area

### Indexed Indirect

Indexed indirect addressing is normally used in conjunction with a table of address held on zero page. The address of the table is taken from the instruction and the X register added to it (with zero page wrap around) to give the location of the least significant byte of the target address.

LDA ($40,X) ;Load a byte indirectly from memory

STA (MEM,X) ;Store accumulator indirectly into memory

### Indirect Indexed

Indirect indirect addressing is the most common indirection mode used on the 6502. In instruction contains the zero page location of the least significant byte of 16 bit address. The Y register is dynamically added to this value to generated the actual target address for operation.

LDA ($40),Y ;Load a byte indirectly from memory

STA (DST),Y ;Store accumulator indirectly into memory

## ****Coding Algorithms****

As you can see from the preceding descriptions the instruction set of the 6502 is quite basic, having only simple 8 bit operations. Complex operations such as 16 or 32 bit arithmetic and memory transfers have to be performed by executing a sequence of simpler operations. This sections describes how to build these algorithms and is based on code taken from my macro library (available from the [download section](https://www.nesdev.org/obelisk-6502-guide/downloads.html#source)).

If you find any bugs in the code, have routines to donate to the library, or can suggest improvements then please mail me.

### Standard Conventions

The 6502 processor expects addresses to be stored in 'little endian' order, with the least significant byte first and the most significant byte second. If the value stored was just a number (e.g. game score, etc.) then we could write code to store and manipulate it in 'big endian' order if we wished, however the algorithms presented here always use 'little endian' order so that they may be applied either to simple numeric values or addresses without modification.

*The terms 'big endian' and 'little endian' come from Gulliver's Travels. The people of Lilliput and Blefuscu have been fighting a war over which end of an boiled egg one should crack to eat it. In computer terms it refers to whether the most or least significant  portion of a binary number is stored in the lower memory address.*

To be safe the algorithms usually start by setting processor flags and registers to safe initial values. If you need to squeeze a few extra bytes or cycles out of the routine you might be able to remove some of these initializations depending on the preceding instructions.

### Simple Memory Operations

Probably the most fundamental memory operation is clearing an area of memory to an initial value, such as zero. As the 6502 cannot directly move values to memory clearing even a small region of memory requires the use of a register. Any of A, X or Y could be used to hold the initial value, but in practice A is normally used because it can be quickly saved and restored (with PHA and PLA) leaving X and Y free for application use.

; Clearing 16 bits of memory

\_CLR16 LDA #0 ;Load constant zero into A

STA MEM+0 ;Then clear the least significant byte

STA MEM+1 ;... followed by the most significant

; Clearing 32 bits of memory

\_CLR32 LDA #0 ;Load constant zero into A

STA MEM+0 ;Clear from the least significant byte

STA MEM+1 ;... up

STA MEM+2 ;... to

STA MEM+3 ;... the most significant

Moving a small quantity of data requires a register to act as a temporary container during the transfer. Again any of A, X, or Y may be used, but as before using A as the temporary register is often the most practical.

; Moving 16 bits of memory

\_XFR16 LDA SRC+0 ;Move the least significant byte

STA DST+0

LDA SRC+1 ;Then the most significant

STA DST+1

; Moving 32 bits of memory

\_XFR32 LDA SRC+0 ;Move from least significant byte

STA DST+0

LDA SRC+1 ;... up

STA DST+1

LDA SRC+2 ;... to

STA DST+2

LDA SRC+3 ;... the most significant

STA DST+3

Provided the source and destination areas do not overlap then the order in which the bytes are moved is irrelevant, but it usually pays to be consistent in your approach to make mistakes easier to spot.

All of the preceding examples can be extended to apply to larger memory areas but will generate increasingly larger code as the number of bytes involved grows. Algorithms that iterate using a counter and use index addressing to access memory will result in smaller code but will be slightly slower to execute.

This trade off between speed and size is a common issue in assembly language programming and there are times when one approach is clearly better than the other (e.g. when trying to squeeze code into a fixed size ROM - SIZE, or manipulate data during a video blanking period - SPEED).

; Clear 32 bits of memory iteratively

\_CLR32C LDX #3

LDA #0

\_LOOP STA MEM,X

DEX

BPL \_LOOP

; Move 32 bits of memory iteratively

\_XFR32C LDX #3

\_LOOP LDA SRC,X

STA DST,X

DEX

BPL \_LOOP

Another basic operation is setting a 16 bit word to an initial constant value. The easiest way to do this is to load the low and high portions into A one at a time and store them.

; Setting a 16 bit constant

\_SET16I LDA #LO NUM ;Set the least significant byte of the constant

STA MEM+0

LDA #HI NUM ;... then the most significant byte

STA MEM+1

### Logical Operations

The simplest forms of operation on binary values are the logical AND, logical OR and exclusive OR illustrated by the following truth tables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Logical AND (AND)*** |  | ***Logical OR (ORA)*** |  | ***Exclusive OR (EOR)*** |
| |  |  |  | | --- | --- | --- | |  | 0 | 1 | | 0 | 0 | 0 | | 1 | 0 | 1 | |  | |  |  |  | | --- | --- | --- | |  | 0 | 1 | | 0 | 0 | 1 | | 1 | 1 | 1 | |  | |  |  |  | | --- | --- | --- | |  | 0 | 1 | | 0 | 0 | 1 | | 1 | 1 | 0 | |

These results can be summarized in English as:

* The result of a logical AND is true (1) if and only if both inputs are true, otherwise it is false (0).
* The result of a logical OR is true (1) if either of the inputs its true, otherwise it is false (0).
* The result of an exclusive OR is true (1) if and only if one input is true and the other is false, otherwise it is false (0).

The tables show result of applying these operations on two one-bit values but as the 6502 comprises of eight bit registers and memory each instruction will operate on two eight bit values simultaneously as shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Logical AND (AND)*** |  | ***Logical OR (ORA)*** |  | ***Exclusive OR (EOR)*** |
| ***Value 1*** | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| ***Value 2*** | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| ***Result*** | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | |

It is important to understand the properties and practical applications of each of these operations as they are extensively used in other algorithms.

* Logical AND operates as a filter and is often used to select a subset of bits from a value (e.g. the status flags from a peripheral control chip).
* Logical OR allows bits to be inserted into an existing value (e.g. to set control flags in a peripheral control chip).
* Exclusive OR allows selected bits to be set or inverted.

In the 6502 these operations are implemented by the [AND](https://www.nesdev.org/obelisk-6502-guide/reference.html#AND), [ORA](https://www.nesdev.org/obelisk-6502-guide/reference.html#ORA) and [EOR](https://www.nesdev.org/obelisk-6502-guide/reference.html#EOR) instructions. One of the values to be operated on will be the current contents of the accumulator, the other is in memory either as an immediate value or at a specified location. The result of the operation is placed in the accumulator and the zero and negative flags are set accordingly.

; Example logical operations

AND #$0F ;Filter out all but the least 4 bits

ORA BITS,X ;Insert some bits from a table

EOR (DATA),Y ;EOR against some data

A very common use of the EOR instruction is to calculate the 'complement' (or logical NOT) of a value. This involves inverting every bit in the value and is most easily calculated by exclusively ORing against an all ones value.

; Calculate the complement

EOR #$FF

The macro library contains reference code for 16 and 32 bit AND, ORA, EOR and NOT operations although there is very little use for them outside of interpreters.

### Shifts & Rotates

The shift and rotate instructions allow the bits within either the accumulator or a memory location to be moved by one place either up (left) or down (right). When the bits are moved a new value will be needed to fill the vacant position created at one end of the value, and similarly the bit displaced at the opposite end will need to be caught and stored.

Both shifts and rotates catch the displaced bit in the carry flag but they differ in how they fill the vacant position; shifts will always fill the vacant bit with a zero whilst a rotate will fill it with the value of the carry flag as it was at the start of the instruction.

For example the following diagram shows the result of applying an 'Arithmetic Shift Left' ([ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL)) to the value $4D to give $9A.

+---+---+---+---+---+---+---+---+

Initial: | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

+---+---+---+---+---+---+---+---+

| | | | | | | |

/ / / / / / / /

/ / / / / / / / 0

/ / / / / / / / /

/ | | | | | | | |

/ v v v v v v v v

+---+---+---+---+---+---+---+---+

Result: C=0  | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

+---+---+---+---+---+---+---+---+

Whist the following shows the result of applying a 'Rotate Left' ([ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL)) to the same value, but assuming that the carry contained the value one.

+---+---+---+---+---+---+---+---+

Initial: | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | C=1

+---+---+---+---+---+---+---+---+

| | | | | | | | /

/ / / / / / / / /

/ / / / / / / / /

/ / / / / / / / /

/ | | | | | | | |

/ v v v v v v v v

+---+---+---+---+---+---+---+---+

Result: C=0  | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

+---+---+---+---+---+---+---+---+

Shifting the bits within a value (and introducing a zero as the least significant bit) has the effect of multiplying its value by two. In order to apply this multiplication to a value larger than a single byte we use ASL to shift the first byte and then ROL all the subsequent bytes as necessary using the carry flag to temporarily hold the displaced bits as they are moved from one byte to the next.

; Shift a 16bit value by one place left (e.g. multiply by two)

\_ASL16 ASL MEM+0 ;Shift the LSB

ROL MEM+1 ;Rotate the MSB

The behavior of the right shift as rotates follows the same pattern. For example we can apply a 'Logical Shift Right' ([LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR) ) to the value $4D to give $26.

+---+---+---+---+---+---+---+---+

Initial: | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

+---+---+---+---+---+---+---+---+

| | | | | | | |

\ \ \ \ \ \ \ \

0 \ \ \ \ \ \ \ \

\ \ \ \ \ \ \ \ \

| | | | | | | | \

v v v v v v v v \

+---+---+---+---+---+---+---+---+

Result:   | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | C=1

+---+---+---+---+---+---+---+---+

Or a 'Rotate Right' ([ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR)) of the same value, but assuming that the carry contained the value one to give $A6.

+---+---+---+---+---+---+---+---+

Initial: C=1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

+---+---+---+---+---+---+---+---+

\ | | | | | | | |

\ \ \ \ \ \ \ \ \

\ \ \ \ \ \ \ \ \

\ \ \ \ \ \ \ \ \

| | | | | | | | \

v v v v v v v v \

+---+---+---+---+---+---+---+---+

Result:   | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | C=1

+---+---+---+---+---+---+---+---+

Not surprisingly if left shifts multiply a value by two then right shifts do an unsigned division by two. Again if we are applying the division to a multi-byte value we will typically use LSR on the first byte (the MSB this time) and ROR on all subsequent bytes.

; Shift a 16 bit value by one place right (e.g. divide by two)

\_LSR16 LSR MEM+1 ;Shift the MSB

ROR MEM+0 ;Rotate the LSB

There are a number of applications for shifts and rotates, not least the coding of generic multiply and divide algorithms which are discussed later.

As was pointed out earlier right shifting a value two divide it by two only works on unsigned values. This is because the LSR is will always place a zero in the most significant bit of the MSB. To make this algorithm work for all two complement coded values we need to ensure that value of this bit is copied back into itself to keep the value the same sign. We can use another shift to achieve this.

; Divide a signed 16 bit value by two

\_DIV2 LDA MEM+1 ;Load the MSB

ASL A ;Copy the sign bit into C

ROR MEM+1 ;And back into the MSB

ROR MEM+0 ;Rotate the LSB as normal

### Addition & Subtraction

The 6502 processor provides 8 bit addition and subtraction instructions and a carry/borrow flag that is used to propagate the carry bit between operations.

To implement a 16 bit addition the programmer must code two pairs of additions; one for the least significant bytes and one for the most significant bytes. The carry flag must be cleared before the first addition to ensure that an additional increment isn't performed.

; 16 bit Binary Addition

CLC ;Ensure carry is clear

LDA VLA+0 ;Add the two least significant bytes

ADC VLB+0

STA RES+0 ;... and store the result

LDA VLA+1 ;Add the two most significant bytes

ADC VLB+1 ;... and any propagated carry bit

STA RES+1 ;... and store the result

Subtraction follows the same pattern but the carry must be set before the first pair of bytes are subtracted to get the correct result.

; 16 bit Binary Subtraction

SEC ;Ensure carry is set

LDA VLA+0 ;Subtract the two least significant bytes

SBC VLB+0

STA RES+0 ;... and store the result

LDA VLA+1 ;Subtract the two most significant bytes

SBC VLB+1 ;... and any propagated borrow bit

STA RES+1 ;... and store the result

Both the addition and subtraction algorithm can be extended to 32 bits by repeating the LDA/ADC/STA or LDA/SBC/STA pattern for two further bytes worth of data.

### Negation

The traditional approach to negating a twos complement number is to reverse all the bits (by EORing with $FF) and add one as shown below.

; 8 bit Binary Negation

CLC ;Ensure carry is clear

EOR #$FF ;Invert all the bits

ADC #1 ;... and add one

This technique works well with a single byte already held in the accumulator but not with bigger numbers. With these it is easier just to subtract them from zero.

; 16 bit Binary Negation

SEC ;Ensure carry is set

LDA #0 ;Load constant zero

SBC SRC+0 ;... subtract the least significant byte

STA DST+0 ;... and store the result

LDA #0 ;Load constant zero again

SBC SRC+1 ;... subtract the most significant byte

STA DST+1 ;... and store the result

### Decimal Arithmetic

The behavior of the ADC and SBC instructions can be modified by setting or clearing the decimal mode flag in the processor status register. Normally decimal mode is disabled and ADC/SBC perform simple binary arithmetic (e.g. $99 + $01 => $9A Carry = 0), but if the flag is set with a SED instruction the processor will perform binary coded decimal arithmetic instead (e.g. $99 + $01 => $00 Carry = 1).

To make the 16 bit addition/subtraction code work in decimal mode simply include an SED at the start and a CLD at the end (to restore the processor to normal).

; 16 bit Binary Code Decimal Addition

SED ;Set decimal mode flag

CLC ;Ensure carry is clear

LDA VLA+0 ;Add the two least significant bytes

ADC VLB+0

STA RES+0 ;... and store the result

LDA VLA+1 ;Add the two most significant bytes

ADC VLB+1 ;... and any propagated carry bit

STA RES+1 ;... and store the result

CLD ;Clear decimal mode

Binary coded values are more easily converted to displayable digits and are useful for holding numbers such as high scores.

; Print the BCD value in A as two ASCII digits

PHA ;Save the BCD value

LSR A ;Shift the four most significant bits

LSR A ;... into the four least significant

LSR A

LSR A

ORA #'0' ;Make an ASCII digit

JSR PRINT ;... and print it

PLA ;Recover the BCD value

AND #$0F ;Mask out all but the bottom 4 bits

ORA #'0' ;Make an ASCII digit

JSR PRINT ;... and print it

Another use for BCD is in the conversion of binary values to decimal ones. Some algorithms perform this conversion by counting the number of times that 10000's, 1000's, 100's, 10's and 1's can be subtracted from the binary value before it underflows, but I normally use a simple fixed loop that shifts the bits out of the binary value one at a time and adds it to an intermediate result that is being doubled (in BCD) on each iteration.

; Convert an 16 bit binary value into a 24bit BCD value

BIN2BCD LDA #0 ;Clear the result area

STA RES+0

STA RES+1

STA RES+2

LDX #16 ;Setup the bit counter

SED ;Enter decimal mode

\_LOOP ASL VAL+0 ;Shift a bit out of the binary

ROL VAL+1 ;... value

LDA RES+0 ;And add it into the result, doubling

ADC RES+0 ;... it at the same time

STA RES+0

LDA RES+1

ADC RES+1

STA RES+1

LDA RES+2

ADC RES+2

STA RES+2

DEX ;More bits to process?

BNE \_LOOP

CLD ;Leave decimal mode

One final odd use of decimal arithmetic is the conversion of hexadecimal digits to printable ASCII characters. The usual way to perform this conversion is to add $30 to the digit ($00 - $0F) to make an intermediate result which is then examined to see if it is greater than or equal to $3A. If it is then an additional $06 is added to make the result fall in the range $41 - $46 (e.g. 'A' - 'F').

; Convert a hex digit ($00-$0F) to ASCII ('0'-'9' or 'A'-'F')

HEX2ASC ORA #$30 ;Form the basic character code

CMP #$3A ;Does the result need adjustment?

BCC .+4

ADC #$05 ;Add 6 (5 and the carry) if needed

It turns out that in decimal mode the processor does basically the same correction after an addition and with the right arguments we can convert the digit to its ASCII character without performing any comparisons as shown in the following code.

; Convert a hex digit ($00-$0F) to ASCII ('0'-'9' or 'A'-'F')

HEX2ASC SED ;Enter BCD mode

CLC ;Ensure the carry is clear

ADC #$90 ;Produce $90-$99 (C=0) or $00-$05 (C=1)

ADC #$40 ;Produce $30-$39 or $41-$46

CLD ;Leave BCD mode

### Increments & Decrements

Assembly programs frequently use memory based counters that occasionally need incrementing or decrementing by one. One way to achieve this would be to load the LSB and MSB in turn and add or subtract one with the [ADC](https://www.nesdev.org/obelisk-6502-guide/reference.html#ADC)/[SBC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SBC) instructions, but the 6502 has a more efficient way to do this using [INC](https://www.nesdev.org/obelisk-6502-guide/reference.html#INC) and [DEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEC).

Incrementing is straight forward, we just increment the least significant byte until the result becomes zero. This indicates that the calculation has wrapped round (e.g. $FF + $01 => $00) and an increment to the most significant byte is needed.

; Increment a 16 bit value by one

\_INC16 INC MEM+0 ;Increment the LSB

BNE \_DONE ;If the result was not zero we're done

INC MEM+1 ;Increment the MSB if LSB wrapped round

\_DONE EQU \*

Decrementing is a little trickier because we need to know when the least significant byte is about to underflow from $00 to $FF. The answer is to test it first by loading it into the accumulator to set the processor flags.

; Decrement a 16 bit value by one

\_DEC16 LDA MEM+0 ;Test if the LSB is zero

BNE \_SKIP ;If it isn't we can skip the next instruction

DEC MEM+1 ;Decrement the MSB when the LSB will underflow

\_SKIP DEC MEM+0 ;Decrement the LSB

### Complex Memory Transfers

Moving data from one place to another is a common operation. If the amount of data to moved is 256 bytes or less and the source and target locations of the data are fixed then a simple loop around an indexed LDA followed by an indexed STA is the most efficient. Note that whilst both the X and Y registers can be used in indexed addressing modes  an asymmetry in the 6502's instruction means that X is the better register to use if one or both of the memory areas resides on zero page.

; Move 256 bytes or less in a forward direction

LDX #0 ;Start with the first byte

\_LOOP LDA SRC,X ;Move it

STA DST,X

INX ;Then bump the index ...

CPX #LEN ;... until we reach the limit

BNE \_LOOP

The corresponding code moving the last byte first is as follows:

; Move 256 bytes or less in a reverse direction

LDX #LEN ;Start with the last byte

\_LOOP DEX ;Bump the index

LDA SRC,X ;Move a byte

STA DST,X

CPX #0 ;... until all bytes have moved

BNE \_LOOP

If the amount is even smaller (128 bytes or less) then we can eliminate the comparison against the limit and use the settings of the flags after a DEX to determine if the loop has finished.

; Move 128 bytes or less in a reverse direction

LDX #LEN-1 ;Start with the last byte

\_LOOP LDA SRC,X ;Move it

STA DST,X

DEX ;Then bump the index ...

BPL \_LOOP ;... until all bytes have moved

To create a completely generic memory transfer we must change to using indirect indexed addressing to access memory and use all the registers. The following code shows a forward transferring algorithm which first moves complete pages of 256 bytes followed by any remaining fragments of smaller size.

\_MOVFWD LDY #0 ;Initialise the index

LDX LEN+1 ;Load the page count

BEQ \_FRAG ;... Do we only have a fragment?

\_PAGE LDA (SRC),Y ;Move a byte in a page transfer

STA (DST),Y

INY ;And repeat for the rest of the

BNE \_PAGE ;... page

INC SRC+1 ;Then bump the src and dst addresses

INC DST+1 ;... by a page

DEX ;And repeat while there are more

BNE \_PAGE ;... pages to move

\_FRAG CPY LEN+0 ;Then while the index has not reached

BEQ \_DONE ;... the limit

LDA (SRC),Y ;Move a fragment byte

STA (DST),Y

INY ;Bump the index and repeat

BNE \_FRAG\?

\_DONE EQU \* ;All done

### Character Classification

The standard C library provides a set of functions for classifying (e.g. is letter, is digit, is ASCII, is upper case, etc.) and modifying (e.g. to upper case and to lower case) characters defined in a header called <ctype.h>. This section describes how a similar set of functions can be coded in 6502 assembler. There are two techniques that can be applied to solve this problem, namely, comparisons or look up tables.

*Note: These functions will be restricted to just the normal ASCII character range $00-$7F.*

The look up table required to implement character classification  needs a byte per character. Bits within the look up table indicate how the character is to be classified (e.g. control character, printable character, white space, decimal digit, hexadecimal digit, punctuation, upper case latter or lower case letter). To test a character for a specific classification you load its description byte from the table and test for the presence of certain bits (e.g. with AND).

; Constants describing the role of each classification bit

\_CTL EQU $80

\_PRN EQU $40

\_WSP EQU $20

\_PCT EQU $10

\_UPR EQU $08

\_LWR EQU $04

\_DGT EQU $02

\_HEX EQU $01

; Test if the character in A is a control character

ISCNTRL TAX

LDA #\_CTL

BNE TEST

; Test if the character in A is printable

ISPRINT TAX

LDA #\_PRN

BNE TEST

; Test if the character in A is punctation

ISPUNCT TAX

LDA #\_PCT

BNE TEST

; Test if the character in A is upper case

ISUPPER TAX

LDA #\_UPR

BNE TEST

; Test if the character in A is lower case

ISLOWER TAX

LDA #\_LWR

BNE TEST

; Test if the character in A is a letter

ISALPHA TAX

LDA #\_UPR|\_LWR

BNE TEST

; Test if the character in A is a decimal digit

ISDIGIT TAX

LDA #\_DGT

BNE TEST

; Test if the character in A is a hexadecimal digit

ISXDIGIT TAX

LDA #\_HEX

BNE TEST

; Test if the character in A is letter or a digit

ISALNUM TAX

LDA #\_DGT|\_UPR|\_LWR

; Tests for the required bits in the look up table value

TEST AND CTYPE,X

BEQ FAIL

; Set the carry flag if any target bits were found

PASS TXA

SEC

RTS

; Test if the character in A is in the ASCII range $00-$7F

ISASCII TAX

BPL PASS

; Clear the carry flag if no target bits were found

FAIL TXA

CLC

RTS

; If A contains a lower case letter convert it to upper case

TOUPPER JSR ISLOWER

BCC \*+4

AND #$DF

RTS

; If A contains an upper case letter convert it to lower case

TOLOWER JSR ISUPPER

BCC \*+4

ORA #$20

RTS

; The lookup table of character descriptions

CTYPE DB \_CTL ; NUL

DB \_CTL ; SOH

DB \_CTL ; STX

DB \_CTL ; ETX

DB \_CTL ; EOT

DB \_CTL ; ENQ

DB \_CTL ; ACK

DB \_CTL ; BEL

DB \_CTL ; BS

DB \_CTL|\_WSP ; TAB

DB \_CTL|\_WSP ; LF

DB \_CTL|\_WSP ; VT

DB \_CTL|\_WSP ; FF

DB \_CTL|\_WSP ; CR

DB \_CTL ; SO

DB \_CTL ; SI

DB \_CTL ; DLE

DB \_CTL ; DC1

DB \_CTL ; DC2

DB \_CTL ; DC3

DB \_CTL ; DC4

DB \_CTL ; NAK

DB \_CTL ; SYN

DB \_CTL ; ETB

DB \_CTL ; CAN

DB \_CTL ; EM

DB \_CTL ; SUB

DB \_CTL ; ESC

DB \_CTL ; FS

DB \_CTL ; GS

DB \_CTL ; RS

DB \_CTL ; US

DB \_PRN|\_WSP ; SPACE

DB \_PRN|\_PCT ; !

DB \_PRN|\_PCT ; "

DB \_PRN|\_PCT ; #

DB \_PRN|\_PCT ; $

DB \_PRN|\_PCT ; %

DB \_PRN|\_PCT ; &

DB \_PRN|\_PCT ; '

DB \_PRN|\_PCT ; (

DB \_PRN|\_PCT ; )

DB \_PRN|\_PCT ; \*

DB \_PRN|\_PCT ; +

DB \_PRN|\_PCT ; ,

DB \_PRN|\_PCT ; -

DB \_PRN|\_PCT ; .

DB \_PRN|\_PCT ; /

DB \_PRN|\_DGT|\_HEX ; 0

DB \_PRN|\_DGT|\_HEX ; 1

DB \_PRN|\_DGT|\_HEX ; 2

DB \_PRN|\_DGT|\_HEX ; 3

DB \_PRN|\_DGT|\_HEX ; 4

DB \_PRN|\_DGT|\_HEX ; 5

DB \_PRN|\_DGT|\_HEX ; 6

DB \_PRN|\_DGT|\_HEX ; 7

DB \_PRN|\_DGT|\_HEX ; 8

DB \_PRN|\_DGT|\_HEX ; 9

DB \_PRN|\_PCT ; :

DB \_PRN|\_PCT ; ;

DB \_PRN|\_PCT ; <

DB \_PRN|\_PCT ; =

DB \_PRN|\_PCT ; >

DB \_PRN|\_PCT ; ?

DB \_PRN|\_PCT ; @

DB \_PRN|\_UPR|\_HEX ; A

DB \_PRN|\_UPR|\_HEX ; B

DB \_PRN|\_UPR|\_HEX ; C

DB \_PRN|\_UPR|\_HEX ; D

DB \_PRN|\_UPR|\_HEX ; E

DB \_PRN|\_UPR|\_HEX ; F

DB \_PRN|\_UPR ; G

DB \_PRN|\_UPR ; H

DB \_PRN|\_UPR ; I

DB \_PRN|\_UPR ; J

DB \_PRN|\_UPR ; K

DB \_PRN|\_UPR ; L

DB \_PRN|\_UPR ; M

DB \_PRN|\_UPR ; N

DB \_PRN|\_UPR ; O

DB \_PRN|\_UPR ; P

DB \_PRN|\_UPR ; Q

DB \_PRN|\_UPR ; R

DB \_PRN|\_UPR ; S

DB \_PRN|\_UPR ; T

DB \_PRN|\_UPR ; U

DB \_PRN|\_UPR ; V

DB \_PRN|\_UPR ; W

DB \_PRN|\_UPR ; X

DB \_PRN|\_UPR ; Y

DB \_PRN|\_UPR ; Z

DB \_PRN|\_PCT ; [

DB \_PRN|\_PCT ; \

DB \_PRN|\_PCT ; ]

DB \_PRN|\_PCT ; ^

DB \_PRN|\_PCT ; \_

DB \_PRN|\_PCT ; `

DB \_PRN|\_LWR|\_HEX ; a

DB \_PRN|\_LWR|\_HEX ; b

DB \_PRN|\_LWR|\_HEX ; c

DB \_PRN|\_LWR|\_HEX ; d

DB \_PRN|\_LWR|\_HEX ; e

DB \_PRN|\_LWR|\_HEX ; f

DB \_PRN|\_LWR ; g

DB \_PRN|\_LWR ; h

DB \_PRN|\_LWR ; i

DB \_PRN|\_LWR ; j

DB \_PRN|\_LWR ; k

DB \_PRN|\_LWR ; l

DB \_PRN|\_LWR ; m

DB \_PRN|\_LWR ; n

DB \_PRN|\_LWR ; o

DB \_PRN|\_LWR ; p

DB \_PRN|\_LWR ; q

DB \_PRN|\_LWR ; r

DB \_PRN|\_LWR ; s

DB \_PRN|\_LWR ; t

DB \_PRN|\_LWR ; u

DB \_PRN|\_LWR ; v

DB \_PRN|\_LWR ; w

DB \_PRN|\_LWR ; x

DB \_PRN|\_LWR ; y

DB \_PRN|\_LWR ; z

DB \_PRN|\_PCT ; {

DB \_PRN|\_PCT ; |

DB \_PRN|\_PCT ; }

DB \_PRN|\_PCT ; ~

DB \_CTL ; DEL

If we use comparisons then each function will consist of a number of comparison stages to determine if a provided character has an appropriate value. In most cases these functions are quite small but one or two of them may involve many stages (e.g. is punctuation). The execution time will vary according to the number of the tests a character is subjected to.

ISUPPER CMP #'A'

BCC FAIL

CMP #'Z'+1

BCS FAIL

; Drop thru here on success

ISLOWER CMP #'a'

BCC FAIL

CMP #'z'+1

BCS FAIL

; Drop thru here on success

ISALPHA CMP #'A'

BCC FAIL

CMP #'Z'+1

BCC PASS

CMP #'a'

BCC FAIL

CMP #'z'+1

BCS FAIL

PASS EQU \*

; Drop thru here on success

Which solution is best? As in so many cases it depends on your program. If you only need one or two tests and memory size is an issue then the comparison approach will generate less code but may be slightly slower (for the complex tests), otherwise the look up table is simple and fast.

### Some notes on my macro library

As I said in the introduction to this section all of the algorithms presented here are taken from my macro library. Coding simple algorithms like these as macros has several advantages over subroutine libraries on the 6502 processor, namely:

* The assembler adjusts them automatically to zero page or absolute addressing depending on the parameters.
* They can be used either inline (for speed) or expanded into subroutines (to save space) as needed.
* The same macro can be used several times but customized in each case to suit its use at that time.
* The macros can optimize the code they generate under some circumstances (e.g. \_XFR16 detects when the source and target addresses are the same and does nothing).

Another feature of the macros is that they will generate code for the 65SC02 processor using the additional instructions on that processor if the assembler defines the correct symbol. (This processor was used in the BBC Microcomputers 6502 second processor that's why I decided to support it).

The routines in the currently library are:

|  |  |
| --- | --- |
| **Macro Name** | **Description** |
| \_CLR16 | Clears 16 bits of memory to zero |
| \_CLR32 | Clears 32 bits of memory to zero |
| \_CLR32C | Clears 32 bits of memory to zero iteratively |
| \_XFR16 | Moves 16 bits of memory |
| \_XFR32 | Moves 32 bits of memory |
| \_XFR32C | Moves 32 bits of memory iteratively |
| \_SET16I | Load a 16 bit constant into memory |
| \_NOT16 | Compute the NOT of a 16 bit value |
| \_NOT32 | Compute the NOT of a 32 bit value |
| \_NOT32C | Compute the NOT of a 32 bit value iteratively |
| \_ORA16 | Compute the OR of two 16 bit values |
| \_ORA32 | Compute the OR of two 32 bit values |
| \_ORA32C | Compute the OR of two 32 bit values iteratively |
| \_AND16 | Compute the AND of two 16 bit values |
| \_AND32 | Compute the AND of two 32 bit values |
| \_AND32C | Compute the AND of two 32 bit values iteratively |
| \_EOR16 | Compute the EOR of two 16 bit values |
| \_EOR32 | Compute the EOR of two 32 bit values |
| \_EOR32C | Compute the EOR of two 32 bit values iteratively |
| \_ASL16 | Compute the arithmetic left shift of a 16 bit value |
| \_ASL32 | Compute the arithmetic left shift of a 32 bit value |
| \_ROL16 | Compute the left rotation of a 16 bit value |
| \_ROL32 | Compute the left rotation of a 32 bit value |
| \_LSR16 | Compute the logical right shift of a 16 bit value |
| \_LSR32 | Compute the logical right shift of a 32 bit value |
| \_ROR16 | Compute the right rotation of a 16 bit value |
| \_ROR32 | Compute the right rotation of a 32 bit value |
| \_INC16 | Increment a 16 bit value |
| \_INC32 | Increment a 32 bit value |
| \_DEC16 | Decrement a 16 bit value |
| \_DEC32 | Decrement a 32 bit value |
| \_ADD16 | Add two 16 bit values |
| \_ADD32 | Add two 32 bit values |
| \_SUB16 | Subtract two 16 bit values |
| \_SUB32 | Subtract two 32 bit values |
| \_NEG16 | Negate a 16 bit value |
| \_NEG32 | Negate a 32 bit value |
| \_ABS16 | Compute the absolute value of a 16 bit value |
| \_ABS32 | Compute the absolute value of a 32 bit value |
| \_MUL16 | Calculate the 16 bit product of two 16 bit values |
| \_MUL16X | Calculate the 32 bit product of two 16 bit values |
| \_MUL32 | Calculate the 32 bit product of two 32 bit values |
| \_MUL16I | Generate the code for a 16 bit constant multiply |
| \_DIV16 | Calculate the 16 bit quotient & remainder of a 16 bit value and 16 bit dividend |
| \_DIV16X | Calculate the 16 bit quotient & remainder of a 32 bit value and 16 bit dividend |
| \_DIV32 | Calculate the 32 bit quotient & remainder of a 32 bit value and 32 bit dividend |
| \_CMP16 | Compare two 16 bit values |
| \_CMP32 | Compare two 32 bit values |
| \_MEMFWD | Move a block for memory a forward direction |
| \_MEMREV | Not Implemented |
| \_MEMCPY | Not Implemented |
| \_STRLEN | Compute the length of a 'C' style string |
| \_STRCPY | Copy a 'C' style string |
| \_STRCMP | Compare two 'C' style strings |
| \_STRNCMP | Not implemented |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Examine the code for more details on the macro parameters and usage.

## Instruction Reference

Click on any of following links to go straight to the information for that instruction.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [ADC](https://www.nesdev.org/obelisk-6502-guide/reference.html#ADC) | [AND](https://www.nesdev.org/obelisk-6502-guide/reference.html#AND) | [ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL) | [BCC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCC) | [BCS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCS) | [BEQ](https://www.nesdev.org/obelisk-6502-guide/reference.html#BEQ) | [BIT](https://www.nesdev.org/obelisk-6502-guide/reference.html#BIT) | [BMI](https://www.nesdev.org/obelisk-6502-guide/reference.html#BMI) | [BNE](https://www.nesdev.org/obelisk-6502-guide/reference.html#BNE) | [BPL](https://www.nesdev.org/obelisk-6502-guide/reference.html#BPL) | [BRK](https://www.nesdev.org/obelisk-6502-guide/reference.html#BRK) | [BVC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVC) | [BVS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVS) | [CLC](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLC) |
| [CLD](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLD) | [CLI](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLI) | [CLV](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLV) | [CMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#CMP) | [CPX](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPX) | [CPY](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPY) | [DEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEC) | [DEX](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEX) | [DEY](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEY) | [EOR](https://www.nesdev.org/obelisk-6502-guide/reference.html#EOR) | [INC](https://www.nesdev.org/obelisk-6502-guide/reference.html#INC) | [INX](https://www.nesdev.org/obelisk-6502-guide/reference.html#INX) | [INY](https://www.nesdev.org/obelisk-6502-guide/reference.html#INY) | [JMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#JMP) |
| [JSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#JSR) | [LDA](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDA) | [LDX](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDX) | [LDY](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDY) | [LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR) | [NOP](https://www.nesdev.org/obelisk-6502-guide/reference.html#NOP) | [ORA](https://www.nesdev.org/obelisk-6502-guide/reference.html#ORA) | [PHA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHA) | [PHP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHP) | [PLA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLA) | [PLP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLP) | [ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL) | [ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR) | [RTI](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTI) |
| [RTS](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTS) | [SBC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SBC) | [SEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEC) | [SED](https://www.nesdev.org/obelisk-6502-guide/reference.html#SED) | [SEI](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEI) | [STA](https://www.nesdev.org/obelisk-6502-guide/reference.html#STA) | [STX](https://www.nesdev.org/obelisk-6502-guide/reference.html#STX) | [STY](https://www.nesdev.org/obelisk-6502-guide/reference.html#STY) | [TAX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAX) | [TAY](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAY) | [TSX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TSX) | [TXA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXA) | [TXS](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXS) | [TYA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TYA) |

### ADC - Add with Carry

A,Z,C,N = A+M+C

This instruction adds the contents of a memory location to the accumulator together with the carry bit. If overflow occurs the carry bit is set, this enables multiple byte addition to be performed.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set if overflow in bit 7 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set if sign bit is incorrect |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $69 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $65 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $75 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $6D | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $7D | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $79 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $61 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $71 | 2 | 5 (+1 if page crossed) |

See also: [SBC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SBC)

### AND - Logical AND

A,Z,N = A&M

A logical AND is performed, bit by bit, on the accumulator contents using the contents of a byte of memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $29 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $25 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $35 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $2D | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $3D | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $39 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $21 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $31 | 2 | 5 (+1 if page crossed) |

See also: [EOR](https://www.nesdev.org/obelisk-6502-guide/reference.html#EOR), [ORA](https://www.nesdev.org/obelisk-6502-guide/reference.html#ORA)

### ASL - Arithmetic Shift Left

A,Z,C,N = M\*2 or M,Z,C,N = M\*2

This operation shifts all the bits of the accumulator or memory contents one bit left. Bit 0 is set to 0 and bit 7 is placed in the carry flag. The effect of this operation is to multiply the memory contents by 2 (ignoring 2's complement considerations), setting the carry if the result will not fit in 8 bits.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to contents of old bit 7 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Accumulator](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $0A | 1 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $06 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $16 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $0E | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $1E | 3 | 7 |

See also: [LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR), [ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL), [ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR)

### BCC - Branch if Carry Clear

If the carry flag is clear then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $90 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BCS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCS)

### BCS - Branch if Carry Set

If the carry flag is set then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $B0 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BCC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BCC)

### BEQ - Branch if Equal

If the zero flag is set then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $F0 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BNE](https://www.nesdev.org/obelisk-6502-guide/reference.html#BNE)

### BIT - Bit Test

A & M, N = M7, V = M6

This instructions is used to test if one or more bits are set in a target memory location. The mask pattern in A is ANDed with the value in memory to set or clear the zero flag, but the result is not kept. Bits 7 and 6 of the value from memory are copied into the N and V flags.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if the result if the AND is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set to bit 6 of the memory value |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set to bit 7 of the memory value |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $24 | 2 | 3 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $2C | 3 | 4 |

### BMI - Branch if Minus

If the negative flag is set then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $30 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BPL](https://www.nesdev.org/obelisk-6502-guide/reference.html#BPL)

### BNE - Branch if Not Equal

If the zero flag is clear then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $D0 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BEQ](https://www.nesdev.org/obelisk-6502-guide/reference.html#BEQ)

### BPL - Branch if Positive

If the negative flag is clear then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $10 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BMI](https://www.nesdev.org/obelisk-6502-guide/reference.html#BMI)

### BRK - Force Interrupt

The BRK instruction forces the generation of an interrupt request. The program counter and processor status are pushed on the stack then the IRQ interrupt vector at $FFFE/F is loaded into the PC and the break flag in the status set to one.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Set to 1 |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $00 | 1 | 7 |

The interpretation of a BRK depends on the operating system. On the BBC Microcomputer it is used by language ROMs to signal run time errors but it could be used for other purposes (e.g. calling operating system functions, etc.).

### BVC - Branch if Overflow Clear

If the overflow flag is clear then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $50 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BVS](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVS)

### BVS - Branch if Overflow Set

If the overflow flag is set then add the relative displacement to the program counter to cause a branch to a new location.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Relative](https://www.nesdev.org/obelisk-6502-guide/addressing.html#REL) | $70 | 2 | 2 (+1 if branch succeeds +2 if to a new page) |

See also: [BVC](https://www.nesdev.org/obelisk-6502-guide/reference.html#BVC)

### CLC - Clear Carry Flag

C = 0

Set the carry flag to zero.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to 0 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $18 | 1 | 2 |

See also: [SEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEC)

### CLD - Clear Decimal Mode

D = 0

Sets the decimal mode flag to zero.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Set to 0 |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $D8 | 1 | 2 |

**NB**:  
The state of the decimal flag is uncertain when the CPU is powered up and it is not reset when an interrupt is generated. In both cases you should include an explicit CLD to ensure that the flag is cleared before performing addition or subtraction.

See also: [SED](https://www.nesdev.org/obelisk-6502-guide/reference.html#SED)

### CLI - Clear Interrupt Disable

I = 0

Clears the interrupt disable flag allowing normal interrupt requests to be serviced.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Set to 0 |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $58 | 1 | 2 |

See also: [SEI](https://www.nesdev.org/obelisk-6502-guide/reference.html#SEI)

### CLV - Clear Overflow Flag

V = 0

Clears the overflow flag.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set to 0 |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $B8 | 1 | 2 |

### CMP - Compare

Z,C,N = A-M

This instruction compares the contents of the accumulator with another memory held value and sets the zero and carry flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set if A >= M |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = M |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $C9 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $C5 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $D5 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $CD | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $DD | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $D9 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $C1 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $D1 | 2 | 5 (+1 if page crossed) |

See also: [CPX](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPX), [CPY](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPY)

### CPX - Compare X Register

Z,C,N = X-M

This instruction compares the contents of the X register with another memory held value and sets the zero and carry flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set if X >= M |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X = M |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $E0 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $E4 | 2 | 3 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $EC | 3 | 4 |

See also: [CMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#CMP), [CPY](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPY)

### CPY - Compare Y Register

Z,C,N = Y-M

This instruction compares the contents of the Y register with another memory held value and sets the zero and carry flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set if Y >= M |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if Y = M |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $C0 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $C4 | 2 | 3 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $CC | 3 | 4 |

See also: [CMP](https://www.nesdev.org/obelisk-6502-guide/reference.html#CMP), [CPX](https://www.nesdev.org/obelisk-6502-guide/reference.html#CPX)

### DEC - Decrement Memory

M,Z,N = M-1

Subtracts one from the value held at a specified memory location setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if result is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $C6 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $D6 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $CE | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $DE | 3 | 7 |

See also: [DEX](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEX), [DEY](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEY)

### DEX - Decrement X Register

X,Z,N = X-1

Subtracts one from the X register setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of X is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $CA | 1 | 2 |

See also: [DEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEC), [DEY](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEY)

### DEY - Decrement Y Register

Y,Z,N = Y-1

Subtracts one from the Y register setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if Y is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of Y is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $88 | 1 | 2 |

See also: [DEC](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEC), [DEX](https://www.nesdev.org/obelisk-6502-guide/reference.html#DEX)

### EOR - Exclusive OR

A,Z,N = A^M

An exclusive OR is performed, bit by bit, on the accumulator contents using the contents of a byte of memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $49 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $45 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $55 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $4D | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $5D | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $59 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $41 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $51 | 2 | 5 (+1 if page crossed) |

See also: [AND](https://www.nesdev.org/obelisk-6502-guide/reference.html#AND), [ORA](https://www.nesdev.org/obelisk-6502-guide/reference.html#ORA)

### INC - Increment Memory

M,Z,N = M+1

Adds one to the value held at a specified memory location setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if result is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $E6 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $F6 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $EE | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $FE | 3 | 7 |

See also: [INX](https://www.nesdev.org/obelisk-6502-guide/reference.html#INX), [INY](https://www.nesdev.org/obelisk-6502-guide/reference.html#INY)

### INX - Increment X Register

X,Z,N = X+1

Adds one to the X register setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of X is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $E8 | 1 | 2 |

See also: [INC](https://www.nesdev.org/obelisk-6502-guide/reference.html#INC), [INY](https://www.nesdev.org/obelisk-6502-guide/reference.html#INY)

### INY - Increment Y Register

Y,Z,N = Y+1

Adds one to the Y register setting the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if Y is zero |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of Y is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $C8 | 1 | 2 |

See also: [INC](https://www.nesdev.org/obelisk-6502-guide/reference.html#INC), [INX](https://www.nesdev.org/obelisk-6502-guide/reference.html#INX)

### JMP - Jump

Sets the program counter to the address specified by the operand.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $4C | 3 | 3 |
| [Indirect](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IND) | $6C | 3 | 5 |

NB:  
An original 6502 has does not correctly fetch the target address if the indirect vector falls on a page boundary (e.g. $xxFF where xx is any value from $00 to $FF). In this case fetches the LSB from $xxFF as expected but takes the MSB from $xx00. This is fixed in some later chips like the 65SC02 so for compatibility always ensure the indirect vector is not at the end of the page.

### JSR - Jump to Subroutine

The JSR instruction pushes the address (minus one) of the return point on to the stack and then sets the program counter to the target memory address.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $20 | 3 | 6 |

See also: [RTS](https://www.nesdev.org/obelisk-6502-guide/reference.html#RTS)

### LDA - Load Accumulator

A,Z,N = M

Loads a byte of memory into the accumulator setting the zero and negative flags as appropriate.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of A is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $A9 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $A5 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $B5 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $AD | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $BD | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $B9 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $A1 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $B1 | 2 | 5 (+1 if page crossed) |

See also: [LDX](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDX), [LDY](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDY)

### LDX - Load X Register

X,Z,N = M

Loads a byte of memory into the X register setting the zero and negative flags as appropriate.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of X is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $A2 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $A6 | 2 | 3 |
| [Zero Page,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPY) | $B6 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $AE | 3 | 4 |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $BE | 3 | 4 (+1 if page crossed) |

See also: [LDA](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDA), [LDY](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDY)

### LDY - Load Y Register

Y,Z,N = M

Loads a byte of memory into the Y register setting the zero and negative flags as appropriate.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if Y = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of Y is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $A0 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $A4 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $B4 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $AC | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $BC | 3 | 4 (+1 if page crossed) |

See also: [LDA](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDA), [LDX](https://www.nesdev.org/obelisk-6502-guide/reference.html#LDX)

### LSR - Logical Shift Right

A,C,Z,N = A/2 or M,C,Z,N = M/2

Each of the bits in A or M is shift one place to the right. The bit that was in bit 0 is shifted into the carry flag. Bit 7 is set to zero.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to contents of old bit 0 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if result = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Accumulator](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $4A | 1 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $46 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $56 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $4E | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $5E | 3 | 7 |

See also: [ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL), [ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL), [ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR)

### NOP - No Operation

The NOP instruction causes no changes to the processor other than the normal incrementing of the program counter to the next instruction.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $EA | 1 | 2 |

### ORA - Logical Inclusive OR

A,Z,N = A|M

An inclusive OR is performed, bit by bit, on the accumulator contents using the contents of a byte of memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $09 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $05 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $15 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $0D | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $1D | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $19 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $01 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $11 | 2 | 5 (+1 if page crossed) |

See also: [AND](https://www.nesdev.org/obelisk-6502-guide/reference.html#AND), [EOR](https://www.nesdev.org/obelisk-6502-guide/reference.html#EOR)

### PHA - Push Accumulator

Pushes a copy of the accumulator on to the stack.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $48 | 1 | 3 |

See also: [PLA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLA)

### PHP - Push Processor Status

Pushes a copy of the status flags on to the stack.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $08 | 1 | 3 |

See also: [PLP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PLP)

### PLA - Pull Accumulator

Pulls an 8 bit value from the stack and into the accumulator. The zero and negative flags are set as appropriate.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of A is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $68 | 1 | 4 |

See also: [PHA](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHA)

### PLP - Pull Processor Status

Pulls an 8 bit value from the stack and into the processor flags. The flags will take on new states as determined by the value pulled.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set from stack |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set from stack |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Set from stack |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Set from stack |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Set from stack |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set from stack |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set from stack |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $28 | 1 | 4 |

See also: [PHP](https://www.nesdev.org/obelisk-6502-guide/reference.html#PHP)

### ROL - Rotate Left

Move each of the bits in either A or M one place to the left. Bit 0 is filled with the current value of the carry flag whilst the old bit 7 becomes the new carry flag value.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to contents of old bit 7 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Accumulator](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $2A | 1 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $26 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $36 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $2E | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $3E | 3 | 7 |

See also: [ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL), [LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR), [ROR](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROR)

### ROR - Rotate Right

Move each of the bits in either A or M one place to the right. Bit 7 is filled with the current value of the carry flag whilst the old bit 0 becomes the new carry flag value.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to contents of old bit 0 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of the result is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Accumulator](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $6A | 1 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $66 | 2 | 5 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $76 | 2 | 6 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $6E | 3 | 6 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $7E | 3 | 7 |

See also [ASL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ASL), [LSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#LSR), [ROL](https://www.nesdev.org/obelisk-6502-guide/reference.html#ROL)

### RTI - Return from Interrupt

The RTI instruction is used at the end of an interrupt processing routine. It pulls the processor flags from the stack followed by the program counter.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set from stack |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set from stack |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Set from stack |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Set from stack |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Set from stack |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set from stack |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set from stack |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $40 | 1 | 6 |

### RTS - Return from Subroutine

The RTS instruction is used at the end of a subroutine to return to the calling routine. It pulls the program counter (minus one) from the stack.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $60 | 1 | 6 |

See also: [JSR](https://www.nesdev.org/obelisk-6502-guide/reference.html#JSR)

### SBC - Subtract with Carry

A,Z,C,N = A-M-(1-C)

This instruction subtracts the contents of a memory location to the accumulator together with the not of the carry bit. If overflow occurs the carry bit is clear, this enables multiple byte subtraction to be performed.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Clear if overflow in bit 7 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Set if sign bit is incorrect |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Immediate](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMM) | $E9 | 2 | 2 |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $E5 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $F5 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $ED | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $FD | 3 | 4 (+1 if page crossed) |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $F9 | 3 | 4 (+1 if page crossed) |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $E1 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $F1 | 2 | 5 (+1 if page crossed) |

See also: [ADC](https://www.nesdev.org/obelisk-6502-guide/reference.html#ADC)

### SEC - Set Carry Flag

C = 1

Set the carry flag to one.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Set to 1 |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $38 | 1 | 2 |

See also: [CLC](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLC)

### SED - Set Decimal Flag

D = 1

Set the decimal mode flag to one.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Set to 1 |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $F8 | 1 | 2 |

See also: [CLD](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLD)

### SEI - Set Interrupt Disable

I = 1

Set the interrupt disable flag to one.

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Set to 1 |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $78 | 1 | 2 |

See also: [CLI](https://www.nesdev.org/obelisk-6502-guide/reference.html#CLI)

### STA - Store Accumulator

M = A

Stores the contents of the accumulator into memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $85 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $95 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $8D | 3 | 4 |
| [Absolute,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABX) | $9D | 3 | 5 |
| [Absolute,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABY) | $99 | 3 | 5 |
| [(Indirect,X)](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDX) | $81 | 2 | 6 |
| [(Indirect),Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IDY) | $91 | 2 | 6 |

See also: [STX](https://www.nesdev.org/obelisk-6502-guide/reference.html#STX), [STY](https://www.nesdev.org/obelisk-6502-guide/reference.html#STY)

### STX - Store X Register

M = X

Stores the contents of the X register into memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $86 | 2 | 3 |
| [Zero Page,Y](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPY) | $96 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $8E | 3 | 4 |

See also: [STA](https://www.nesdev.org/obelisk-6502-guide/reference.html#STA), [STY](https://www.nesdev.org/obelisk-6502-guide/reference.html#STY)

### STY - Store Y Register

M = Y

Stores the contents of the Y register into memory.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Zero Page](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPG) | $84 | 2 | 3 |
| [Zero Page,X](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ZPX) | $94 | 2 | 4 |
| [Absolute](https://www.nesdev.org/obelisk-6502-guide/addressing.html#ABS) | $8C | 3 | 4 |

See also: [STA](https://www.nesdev.org/obelisk-6502-guide/reference.html#STA), [STX](https://www.nesdev.org/obelisk-6502-guide/reference.html#STX)

### TAX - Transfer Accumulator to X

X = A

Copies the current contents of the accumulator into the X register and sets the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of X is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $AA | 1 | 2 |

See also: [TXA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXA)

### TAY - Transfer Accumulator to Y

Y = A

Copies the current contents of the accumulator into the Y register and sets the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if Y = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of Y is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $A8 | 1 | 2 |

See also: [TYA](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAY)

### TSX - Transfer Stack Pointer to X

X = S

Copies the current contents of the stack register into the X register and sets the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if X = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of X is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $BA | 1 | 2 |

See also: [TXS](https://www.nesdev.org/obelisk-6502-guide/reference.html#TXS)

### TXA - Transfer X to Accumulator

A = X

Copies the current contents of the X register into the accumulator and sets the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of A is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $8A | 1 | 2 |

See also: [TAX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAX)

### TXS - Transfer X to Stack Pointer

S = X

Copies the current contents of the X register into the stack register.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Not affected |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Not affected |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $9A | 1 | 2 |

See also: [TSX](https://www.nesdev.org/obelisk-6502-guide/reference.html#TSX)

### TYA - Transfer Y to Accumulator

A = Y

Copies the current contents of the Y register into the accumulator and sets the zero and negative flags as appropriate.

Processor Status after use:

|  |  |  |
| --- | --- | --- |
| [C](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | [Carry Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#C) | Not affected |
| [Z](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | [Zero Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#Z) | Set if A = 0 |
| [I](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | [Interrupt Disable](https://www.nesdev.org/obelisk-6502-guide/registers.html#I) | Not affected |
| [D](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | [Decimal Mode Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#D) | Not affected |
| [B](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | [Break Command](https://www.nesdev.org/obelisk-6502-guide/registers.html#B) | Not affected |
| [V](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | [Overflow Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#V) | Not affected |
| [N](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | [Negative Flag](https://www.nesdev.org/obelisk-6502-guide/registers.html#N) | Set if bit 7 of A is set |

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Opcode** | **Bytes** | **Cycles** |
| [Implied](https://www.nesdev.org/obelisk-6502-guide/addressing.html#IMP) | $98 | 1 | 2 |

See also: [TAY](https://www.nesdev.org/obelisk-6502-guide/reference.html#TAY)

## ****Downloads****

This section contains tools and code that you can download from this site and use in your own projects.

### **Useful Tools**

A quick search of the Internet yielded several 6502 macro assemblers of varying degrees of quality. The one that I use was developed by Frank Vorstenbosch and creates binary output files for ROMs that work perfectly in the pcBBC emulator. Its also completely free.

|  |  |
| --- | --- |
| [as65\_111.zip](https://www.nesdev.org/obelisk-6502-guide/files/as65_111.zip) | A very good 6502 (and 65SC02) assembler, ideal for writing BBC Microcomputer ROMs. |

Unfortunately Windows XP service pack 2 removed support for 16-bit MSDOS applications. I persuaded Frank to port he's assembler to Win 32 and you can down it (and others from [here](http://www.kingswood-consulting.co.uk/assemblers/)).

I have also been coding my own portable assembler development package based on Java.

|  |  |
| --- | --- |
| [6502.zip](https://www.nesdev.org/obelisk-6502-guide/files/6502.zip) | JAVA based 6502/65C02/65816 Relocatable Macro Assembler and Linker. |

### **Source Code**

Here are some 6502 assembly files that I've developed as part of my own projects. This first set of files are operating system independent and have some support for the 65SC02 processor.

|  |  |
| --- | --- |
| [maclib.inc](https://www.nesdev.org/obelisk-6502-guide/files/maclib.inc) | Various general purpose algorithms (e.g. arithmetic, memory, etc.) coded in the form of macros so that they can be used either inline or expanded as subroutines. It has not been thoroughly debugged yet and some routines are missing. |

NOLIST

;------------------------------------------------

; 6502 Standard Macro Library

;------------------------------------------------

; Copyright (C),1999 Andrew John Jacobs.

; All rights reserved.

;------------------------------------------------

;------------------------------------------------

; Revision History:

;

; 16-Aug-99 AJJ Initial version.

;

; 14-Nov-01 AJJ Finally got around to filling in

; some missing comments.

;

;------------------------------------------------

; Notes:

;

; This file contains a number of useful 6502

; algorithms for number, string and memory

; operations. The code is written in the form of

; macros rather than subroutines to make them

; more flexible.

;

; The routines in the library assume that 16 and

; 32 bit numbers are represented in little endian

; order, that is the least significant byte in

; the lowest memory location, so that they can

; be applied to addresses as well as pure

; numbers.

;

; The string routines assume that they are

; working with null terminated 'C' style strings.

;

; The main routines sacrifice code size for speed

; and are coded without any iteration. Compact

; versions which use iteration are provided for

; some algorithms and have the same name as the

; original routine with a 'C' suffix (eg. \_XFR32

; => \_XFR32C).

;

; Some of the macros use 65SC02 instructions for

; speed or to reduce the amount code generated if

; the assembler will accept them.

;

; Where possible the macros detect optimizable

; cases and generate more efficient code.

;

; Bugs & Enhancments:

;

; If you find a bug I missed or have a new

; routine you would like to submit to the library

; then mail me at:

;

; Andrew@obelisk.demon.co.uk

PAGE

;------------------------------------------------

; Basic Operations

;------------------------------------------------

; Clear 2 bytes of memory at any location by

; setting it to zero. If 65SC02 instructions are

; available then STZ is used.

;

; On exit: A = ??, X & Y are unchanged.

IF \_\_65SC02\_\_

\_CLR16 MACRO MEM

STZ MEM+0

STZ MEM+1

ENDM

ELSE

\_CLR16 MACRO MEM

LDA #0

STA MEM+0

STA MEM+1

ENDM

ENDIF

; Clear 4 bytes of memory at any location by

; setting it to zero. If 65SC02 instructions are

; available then STZ is used.

;

; On exit: A = ??, X & Y are unchanged.

IF \_\_65SC02\_\_

\_CLR32 MACRO MEM

STZ MEM+0

STZ MEM+1

STZ MEM+2

STZ MEM+3

ENDM

ELSE

\_CLR32 MACRO MEM

LDA #0

STA MEM+0

STA MEM+1

STA MEM+2

STA MEM+3

ENDM

ENDIF

; Clear 4 bytes of memory at any location by

; setting it to zero iteratively. If 65SC02

; instructions are available then STZ is used.

;

; On exit: A = ??, X = $FF, Y is unchanged.

IF \_\_65SC02\_\_

\_CLR32C MACRO MEM

LDX #3

\_LOOP\? STZ MEM,X

DEX

BPL \_LOOP\?

ENDM

ELSE

\_CLR32C MACRO MEM

LDA #0

LDX #3

\_LOOP\? STA MEM,X

DEX

BPL \_LOOP\?

ENDM

ENDIF

; Transfer 2 bytes of memory from one location to

; another using the accumulator. The order in

; which the bytes are moved depends on the

; relative positions of SRC and DST. If SRC and

; DST are the same then no code is generated.

;

; On exit: A = ??, X & Y are unchanged.

\_XFR16 MACRO SRC,DST

IF SRC != DST

IF SRC > DST

LDA SRC+0

STA DST+0

LDA SRC+1

STA DST+1

ELSE

LDA SRC+1

STA DST+1

LDA SRC+0

STA DST+0

ENDIF

ENDIF

ENDM

; Transfer 4 bytes of memory from one location to

; another using the accumulator. The order in

; which the bytes are moved depends on the

; relative positions of SRC and DST. If SRC and

; DST are the same then no code is generated.

;

; On exit: A = ??, X & Y are unchanged.

\_XFR32 MACRO SRC,DST

IF SRC != DST

IF SRC > DST

LDA SRC+0

STA DST+0

LDA SRC+1

STA DST+1

LDA SRC+2

STA DST+2

LDA SRC+3

STA DST+3

ELSE

LDA SRC+3

STA DST+3

LDA SRC+2

STA DST+2

LDA SRC+1

STA DST+1

LDA SRC+0

STA DST+0

ENDIF

ENDIF

ENDM

; Transfer 4 bytes of memory from one location to

; another iteratively using the accumulator. The

; transfer may fail if SRC and DST overlap. If

; SRC and DST are the same then no code is

; generated.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_XFR32C MACRO SRC,DST

IF SRC != DST

LDX #3

\_LOOP\? LDA SRC,X

STA DST,X

DEX

BPL \_LOOP\?

ENDIF

ENDM

; Set the value of a 16 bit location DST with

; the given constant value.

;

; On exit: A = ??, X & Y unchanged.

\_SET16I MACRO NUM,DST

IF NUM != 0

LDA #LO NUM

STA DST+0

LDA #HI NUM

STA DST+1

ELSE

\_CLR16 DST

ENDIF

PAGE

;------------------------------------------------

; Logical Operations

;------------------------------------------------

; Calculate the logical NOT of the 16 bit value

; at location VLA and stores it in location RES.

;

; On exit: A = ??, X & Y are unchanged.

\_NOT16 MACRO VLA,RES

LDA VLA+0

EOR #$FF

STA RES+0

LDA VLA+1

EOR #$FF

STA RES+1

ENDM

; Calculate the logical NOT of the 32 bit value

; at location VLA and stores it in location RES.

;

; On exit: A = ??, X & Y are unchanged.

\_NOT32 MACRO VLA,RES

LDA VLA+0

EOR #$FF

STA RES+0

LDA VLA+1

EOR #$FF

STA RES+1

LDA VLA+2

EOR #$FF

STA RES+2

LDA VLA+3

EOR #$FF

STA RES+3

ENDM

; Calculate the logical NOT of the 32 bit value

; at location VLA iteratively and stores it in

; location RES.

;

; On exit: A = ??, X = $FF, Y are unchanged.

\_NOT32C MACRO VLA,RES

LDX #3

\_LOOP\? LDA VLA,X

EOR #$FF

STA RES,X

DEX

BPL \_LOOP\?

ENDM

; Calculate the logical OR of the two 16 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_XFR16.

;

; On exit: A = ??, X & Y are unchanged.

\_ORA16 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

ORA VLB+0

STA RES+0

LDA VLA+1

ORA VLB+1

STA RES+1

ELSE

\_XFR16 VLA,RES

ENDIF

ENDM

; Calculate the logical OR of a 16 value at

; location VLA with a constant value and

; store the result at location RES.

;

; On exit: A = ??, X & Y are unchanged.

\_ORA16I MACRO VLA,NUM,RES

LDA VLA+0

ORA #LO NUM

STA RES+0

LDA VLA+1

ORA #HI NUM

STA RES+1

ENDM

; Calculate the logical OR of the two 32 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_XFR32.

;

; On exit: A = ??, X & Y are unchanged.

\_ORA32 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

ORA VLB+0

STA RES+0

LDA VLA+1

ORA VLB+1

STA RES+1

LDA VLA+2

ORA VLB+2

STA RES+2

LDA VLA+3

ORA VLB+3

STA RES+3

ELSE

\_XFR32 VLA,RES

ENDIF

ENDM

; Calculate the logical OR of the two 32 bit

; values at locations VLA and VLB iteratively.

; The result is stored in location RES. If VLA

; and VLB are the same the macro expands to a

; \_XFR32C.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_ORA32C MACRO VLA,VLB,RES

IF VLA != VLB

LDX #3

\_LOOP\? LDA VLA,X

ORA VLB,X

STA RES,X

DEX

BPL \_LOOP\?

ELSE

\_XFR32C VLA,RES

ENDIF

ENDM

; Calculate the logical AND of the two 16 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_XFR16.

;

; On exit: A = ??, X & Y are unchanged.

\_AND16 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

AND VLB+0

STA RES+0

LDA VLA+1

AND VLB+1

STA RES+1

ELSE

\_XFR16 VLA,RES

ENDIF

ENDM

; Calculate the logical AND of a 16 value at

; location VLA with a constant value and

; store the result at location RES.

;

; On exit: A = ??, X & Y are unchanged.

\_AND16I MACRO VLA,NUM,RES

LDA VLA+0

AND #LO NUM

STA RES+0

LDA VLA+1

AND #HI NUM

STA RES+1

ENDM

; Calculate the logical AND of the two 32 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_XFR32.

;

; On exit: A = ??, X & Y are unchanged.

\_AND32 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

AND VLB+0

STA RES+0

LDA VLA+1

AND VLB+1

STA RES+1

LDA VLA+2

AND VLB+2

STA RES+2

LDA VLA+3

AND VLB+3

STA RES+3

ELSE

\_XFR32 VLA,RES

ENDIF

ENDM

; Calculate the logical AND of the two 32 bit

; values at locations VLA and VLB iteratively.

; The result is stored in location RES. If VLA

; and VLB are the same the macro expands to a

; \_XFR32C.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_AND32C MACRO VLA,VLB,RES

IF VLA != VLB

LDX #3

\_LOOP\? LDA VLA,X

AND VLB,X

STA RES,X

DEX

BPL \_LOOP\?

ELSE

\_XFR32C VLA,RES

ENDIF

ENDM

; Calculate the exclusive OR of the two 16 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_CLR16.

;

; On exit: A = ??, X & Y are unchanged.

\_EOR16 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

EOR VLB+0

STA RES+0

LDA VLA+1

EOR VLB+1

STA RES+1

ELSE

\_CLR16 RES

ENDIF

ENDM

; Calculate the exclusive OR of a 16 value at

; location VLA with a constant value and

; store the result at location RES.

;

; On exit: A = ??, X & Y are unchanged.

\_EOR16I MACRO VLA,NUM,RES

LDA VLA+0

EOR #LO NUM

STA RES+0

LDA VLA+1

EOR #HI NUM

STA RES+1

ENDM

; Calculate the exclusive OR of the two 32 bit

; values at locations VLA and VLB. The result is

; stored in location RES. If VLA and VLB are the

; same the macro expands to a \_CLR32.

;

; On exit: A = ??, X & Y are unchanged.

\_EOR32 MACRO VLA,VLB,RES

IF VLA != VLB

LDA VLA+0

EOR VLB+0

STA RES+0

LDA VLA+1

EOR VLB+1

STA RES+1

LDA VLA+2

EOR VLB+2

STA RES+2

LDA VLA+3

EOR VLB+3

STA RES+3

ELSE

\_CLR32 RES

ENDIF

ENDM

; Calculate the exclusive OR of the two 32 bit

; values at locations VLA and VLB iteratively.

; The result is stored in location RES. If VLA

; and VLB are the same the macro expands to a

; \_XFR32C.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_EOR32C MACRO VLA,VLB,RES

IF VLA != VLB

LDX #3

\_LOOP\? LDA VLA,X

EOR VLB,X

STA RES,X

DEX

BPL \_LOOP\?

ELSE

\_CLR32C RES

ENDIF

ENDM

PAGE

;------------------------------------------------

; Shift Operations

;------------------------------------------------

; Perform an arithmetic shift left on the 16 bit

; number at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ASL16 MACRO VLA,RES

IF VLA != RES

LDA VLA+0

ASL A

STA RES+0

LDA VLA+1

ROL A

STA RES+1

ELSE

ASL VLA+0

ROL VLA+1

ENDIF

ENDM

; Perform an arithmetic shift left on the 32 bit

; number at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ASL32 MACRO VLA,RES

IF VLA != RES

LDA VLA+0

ASL A

STA RES+0

LDA VLA+1

ROL A

STA RES+1

LDA VLA+2

ROL A

STA RES+2

LDA VLA+3

ROL A

STA RES+3

ELSE

ASL VLA+0

ROL VLA+1

ROL VLA+2

ROL VLA+3

ENDIF

ENDM

; Perform a left rotation on the 16 bit number at

; location VLA and store the result at location

; RES. If VLA and RES are the same then the

; operation is applied directly to the memory,

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ROL16 MACRO VLA,RES

IF VLA != RES

LDA VLA+0

ROL A

STA RES+0

LDA VLA+1

ROL A

STA RES+1

ELSE

ROL VLA+0

ROL VLA+1

ENDIF

ENDM

; Perform a left rotation on the 32 bit number at

; location VLA and store the result at location

; RES. If VLA and RES are the same then the

; operation is applied directly to the memory,

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ROL32 MACRO VLA,RES

IF VLA != RES

LDA VLA+0

ROL A

STA RES+0

LDA VLA+1

ROL A

STA RES+1

LDA VLA+2

ROL A

STA RES+2

LDA VLA+3

ROL A

STA RES+3

ELSE

ROL VLA+0

ROL VLA+1

ROL VLA+2

ROL VLA+3

ENDIF

ENDM

; Perform an logical shift right on the 16 bit

; number at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_LSR16 MACRO VLA,RES

IF VLA != RES

LDA VLA+1

LSR A

STA RES+1

LDA VLA+0

ROR A

STA RES+0

ELSE

LSR VLA+1

ROR VLA+0

ENDIF

ENDM

; Perform an logical shift right on the 32 bit

; number at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_LSR32 MACRO VLA,RES

IF VLA != RES

LDA VLA+3

LSR A

STA RES+3

LDA VLA+2

ROR A

STA RES+2

LDA VLA+1

ROR A

STA RES+1

LDA VLA+0

ROR A

STA RES+0

ELSE

LSR VLA+3

ROR VLA+2

ROR VLA+1

ROR VLA+0

ENDIF

ENDM

; Perform a right rotation on the 16 bit number

; at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ROR16 MACRO VLA,RES

IF VLA != RES

LDA VLA+1

ROR A

STA RES+1

LDA VLA+0

ROR A

STA RES+0

ELSE

ROR VLA+1

ROR VLA+0

ENDIF

ENDM

; Perform a right rotation on the 32 bit number

; at location VLA and store the result at

; location RES. If VLA and RES are the same then

; the operation is applied directly to the memory

; otherwise it is done in the accumulator.

;

; On exit: A = ??, X & Y are unchanged.

\_ROR32 MACRO VLA,RES

IF VLA != RES

LDA VLA+3

ROR A

STA RES+3

LDA VLA+2

ROR A

STA RES+2

LDA VLA+1

ROR A

STA RES+1

LDA VLA+0

ROR A

STA RES+0

ELSE

ROR VLA+3

ROR VLA+2

ROR VLA+1

ROR VLA+0

ENDIF

ENDM

PAGE

;------------------------------------------------

; Arithmetic Operations

;------------------------------------------------

; Increment the 16 bit value at location MEM

; by one.

;

; On exit: A, X & Y are unchanged.

\_INC16 MACRO MEM

INC MEM+0

BNE \_DONE\?

INC MEM+1

\_DONE\? EQU \*

ENDM

; Increment the 32 bit value at location MEM

; by one.

;

; On exit: A, X & Y are unchanged.

\_INC32 MACRO MEM

INC MEM+0

BNE \_DONE\?

INC MEM+1

BNE \_DONE\?

INC MEM+2

BNE \_DONE\?

INC MEM+3

\_DONE\? EQU \*

ENDM

; Decrement the 16 bit value at location MEM

; by one.

;

; On exit: A = ??, X & Y are unchanged.

\_DEC16 MACRO MEM

LDA MEM+0

BNE \_DONE\?

DEC MEM+1

\_DONE\? DEC MEM+0

ENDM

; Decrement the 32 bit value at location MEM

; by one.

;

; On exit: A = ??, X & Y are unchanged.

\_DEC32 MACRO MEM

LDA MEM+0

BNE \_DEC0\?

LDA MEM+1

BNE \_DEC1\?

LDA MEM+2

BNE \_DEC2\?

DEC MEM+3

\_DEC2\? DEC MEM+2

\_DEC1\? DEC MEM+1

\_DEC0\? DEC MEM+0

ENDM

; Add two 16 bit numbers together and store the

; result in another memory location. RES may be

; the same as either VLA or VLB.

;

; On exit: A = ??, X & Y are unchanged.

\_ADD16 MACRO VLA,VLB,RES

IF VLA != VLB

CLC

LDA VLA+0

ADC VLB+0

STA RES+0

LDA VLA+1

ADC VLB+1

STA RES+1

ELSE

\_ASL16 VLA,RES

ENDIF

ENDM

; Add two 32 bit numbers together and store the

; result in another memory location. RES may be

; the same as either VLA or VLB.

;

; On exit: A = ??, X & Y are unchanged.

\_ADD32 MACRO VLA,VLB,RES

IF VLA != VLB

CLC

LDA VLA+0

ADC VLB+0

STA RES+0

LDA VLA+1

ADC VLB+1

STA RES+1

LDA VLA+2

ADC VLB+2

STA RES+2

LDA VLA+3

ADC VLB+3

STA RES+3

ELSE

\_ASL32 VLA,RES

ENDIF

ENDM

; Subtract two 16 bit numbers and store the

; result in another memory location. RES may be

; the same as VLA or VLB.

;

; On exit: A = ??, X & Y are unchanged.

\_SUB16 MACRO VLA,VLB,RES

SEC

LDA VLA+0

SBC VLB+0

STA RES+0

LDA VLA+1

SBC VLB+1

STA RES+1

ENDM

; Subtract two 32 bit numbers and store the

; result in another memory location. RES may be

; the same as VLA or VLB.

;

; On exit: A = ??, X & Y are unchanged.

\_SUB32 MACRO VLA,VLB,RES

SEC

LDA VLA+0

SBC VLB+0

STA RES+0

LDA VLA+1

SBC VLB+1

STA RES+1

LDA VLA+2

SBC VLB+2

STA RES+2

LDA VLA+3

SBC VLB+3

STA RES+3

ENDM

; Negate the signed 16 bit number at location VLA

; and stored the result at location RES. RES may

; be the same as VLA.

;

; On exit: A = ??, X & Y are unchanged.

\_NEG16 MACRO VLA,RES

SEC

LDA #0

SBC VLA+0

STA RES+0

LDA #0

SBC VLA+1

STA RES+1

ENDM

; Negate the signed 32 bit number at location VLA

; and stored the result at location RES. RES may

; be the same as VLA.

;

; On exit: A = ??, X & Y are unchanged.

\_NEG32 MACRO VLA,RES

SEC

LDA #0

SBC VLA+0

STA RES+0

LDA #0

SBC VLA+1

STA RES+1

LDA #0

SBC VLA+2

STA RES+2

LDA #0

SBC VLA+3

STA RES+3

ENDM

; Calculates the absolute value of signed 16 bit

; number at location VLA and stores it in the RES

; location. Less code is generated if VLA and RES

; are the same location. If 65SC02 instructions

; are available a BRA is used to shorten the

; generated code.

;

; On exit: A = ??, X & Y are unchanged.

\_ABS16 MACRO VLA,RES

BIT VLA+0

IF VLA != RES

BPL \_MOVE\?

\_NEG16 VLA,RES

IF \_\_65SC02\_\_

BRA \_DONE\?

ELSE

JMP \_DONE\?

ENDIF

\_MOVE\? EQU \*

\_XFR16 VLA,RES

ELSE

BPL \_DONE\?

\_NEG16 VLA,RES

ENDIF

\_DONE\? EQU \*

ENDM

; Calculates the absolute value of signed 32 bit

; number at location VLA and stores it in the RES

; location. Less code is generated if VLA and RES

; are the same location. If 65SC02 instructions

; are available a BRA is used to shorten the

; generated code.

;

; On exit: A = ??, X & Y are unchanged.

\_ABS32 MACRO VLA,RES

BIT VLA+0

IF VLA != RES

BPL \_MOVE\?

\_NEG32 VLA,RES

IF \_\_65SC02\_\_

BRA \_DONE\?

ELSE

JMP \_DONE\?

ENDIF

\_MOVE\? EQU \*

\_XFR32 VLA,RES

ELSE

BPL \_DONE\?

\_NEG32 VLA,RES

ENDIF

\_DONE\? EQU \*

ENDM

; Calculate the 16 bit product of two 16 bit

; unsigned numbers. Any overflow during the

; calculation is lost. The number at location

; VLA is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_MUL16 MACRO VLA,VLB,RES

\_CLR16 RES

LDX #16

\_LOOP\? EQU \*

\_ASL16 RES,RES

\_ASL16 VLA,VLA

BCC \_NEXT\?

\_ADD16 VLB,RES,RES

\_NEXT\? DEX

BPL \_LOOP\?

ENDM

; Calculate the 32 bit product of two 16 bit

; unsigned numbers. The number at location VLA

; is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_MUL16X MACRO VLA,VLB,RES

\_CLR32 RES

LDX #16

\_LOOP\? EQU \*

\_ASL32 RES,RES

\_ASL16 VLA,VLA

BCC \_NEXT\?

\_ADD16 VLB,RES,RES

BCC \_NEXT\?

\_INC16 RES+2

\_NEXT\? EQU \*

DEX

BPL \_LOOP\?

ENDM

; Calculate the 32 bit product of two 32 bit

; unsigned numbers. Any overflow during the

; calculation is lost. The number at location

; VLA is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_MUL32 MACRO VLA,VLB,RES

\_CLR32 RES

LDX #32

\_LOOP\? EQU \*

\_ASL32 RES,RES

\_ASL32 VLA,VLA

BCC \_NEXT\?

\_ADD32 VLB,RES,RES

\_NEXT\? EQU \*

DEX

BPL \_LOOP\?

ENDM

; These two macros write the code necessary

; to multiply a 16 bit at location VLA by

; a 16 bit constant NUM and store the 16 bit

; result in location RES.

;

; On exit: A = ??, X & Y unchanged.

\_MUL16I MACRO VLA,NUM,RES

IF NUM = 1

\_XFR16 VLA,RES

ELSE

\_CLR16 RES

\_\_MUL16I VLA,NUM,RES

ENDIF

ENDM

\_\_MUL16I MACRO VLA,NUM,RES

IF NUM & $FFFE

\_\_MUL16I VLA,(NUM/2),RES

\_ASL16 RES,RES

ENDIF

IF NUM & $0001

\_ADD16 VLA,RES,RES

ENDIF

ENDM

; Divide the 16 bit number at location VLA

; by the 16 bit number at location VLB

; leaving the 16 bit quotient at QUO and

; the 16 bit remainder in REM. The value in

; location VLA is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_DIV16 MACRO VLA,VLB,QUO,REM

\_CLR16 REM

LDX #16

\_LOOP\? EQU \*

\_ASL16 VLA,VLA

\_ROL16 REM,REM

\_SUB16 REM,VLB,REM

BCS \_NEXT\?

\_ADD16 REM,VLB,REM

\_NEXT\? EQU \*

\_ROL16 QUO,QUO

DEX

BPL \_LOOP\?

ENDM

; Divide the 32 bit number at location VLA

; by the 16 bit number at location VLB

; leaving the 16 bit quotient at QUO and

; the 16 bit remainder in REM. The value in

; location VLA is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_DIV16X MACRO VLA,VLB,QUO,REM

\_CLR16 REM

LDX #32

\_LOOP\? EQU \*

\_ASL32 VLA,VLA

\_ROL16 REM,REM

\_SUB16 REM,VLB,REM

BCS \_NEXT\?

\_ADD16 REM,VLB,REM

\_NEXT\? EQU \*

\_ROL16 QUO,QUO

DEX

BPL \_LOOP\?

ENDM

; Divide the 32 bit number at location VLA

; by the 32 bit number at location VLB

; leaving the 32 bit quotient at QUO and

; the 32 bit remainder in REM. The value in

; location VLA is destroyed.

;

; On exit: A = ??, X = $FF, Y is unchanged.

\_DIV32 MACRO VLA,VLB,QUO,REM

\_CLR32 REM

LDX #32

\_LOOP\? EQU \*

\_ASL32 VLA,VLA

\_ROL32 REM,REM

\_SUB32 REM,VLB,REM

BCS \_NEXT\?

\_ADD32 REM,VLB,REM

\_NEXT\? EQU \*

\_ROL32 QUO,QUO

DEX

BPL \_LOOP\?

ENDM

PAGE

;------------------------------------------------

; Comparative Operations

;------------------------------------------------

; Compares two 16 bit values in memory areas VLA

; and VLB. The comparison starts with the most

; significant bytes and returns as soon as a

; difference is detected.

;

; On exit: A = ??, X & Y are unchanged.

\_CMP16 MACRO VLA,VLB

LDA VLA+1

CMP VLB+1

BNE \_DONE\?

LDA VLA+0

CMP VLB+0

\_DONE\? EQU \*

ENDM

; Compares two 32 bit values in memory areas VLA

; and VLB. The comparison starts with the most

; significant bytes and returns as soon as a

; difference is detected.

;

; On exit: A = ??, X & Y are unchanged.

\_CMP32 MACRO VLA,VLB

LDA VLA+3

CMP VLB+3

BNE \_DONE\?

LDA VLA+2

CMP VLB+2

BNE \_DONE\?

LDA VLA+1

CMP VLB+1

BNE \_DONE\?

LDA VLA+0

CMP VLB+0

\_DONE\? EQU \*

ENDM

PAGE

;------------------------------------------------

; Memory Operations

;------------------------------------------------

; Transfers a block of memory from one place to

; another by copying the bytes starting at the

; front of the block and going forward. SRC and

; DST are destroyed during the copy.

;

; On exit: A, X & Y = ??.

\_MEMFWD MACRO SRC,DST,LEN

LDY #0

LDX LEN+1

BEQ \_FRAG\?

\_PAGE\? LDA (SRC),Y

STA (DST),Y

INY

BNE \_PAGE\?

INC SRC+1

INC DST+1

DEX

BNE \_PAGE\?

\_FRAG\? CPY LEN+0

BEQ \_DONE\?

LDA (SRC),Y

STA (DST),Y

INY

BNE \_FRAG\?

\_DONE\? EQU \*

ENDM

; Transfers a block of memory from one place to

; another by copying the bytes starting at the

; end of the block and going backwards.

\_MEMREV MACRO SRC,DST,LEN

NOP

ENDM

; Tranfers a block of memory from one location to

; another. Depending on the relative positions of

; the blocks an appropriate transfer method is

; used.

\_MEMCPY MACRO SRC,DST,LEN

\_CMP16 SRC,DST

BCC \_SAFE\?

\_MEMFWD SRC,DST,LEN

IF \_\_65SC02\_\_

BRA \_DONE\?

ELSE

JMP \_DONE\?

ENDIF

\_SAFE\? \_MEMREV SRC,DST,LEN

\_DONE\? EQU \*

ENDM

PAGE

;------------------------------------------------

; String Operations

;------------------------------------------------

; Calculates length of a null terminated string

; by searching for its end. The address of the

; string in STR is destroyed during the search.

;

; On exit: A & Y = ??, X is unchanged.

\_STRLEN MACRO STR,LEN

LDY #0

STY LEN+1

\_LOOP\? LDA (STR),Y

BEQ \_DONE\?

INY

BNE \_LOOP\?

INC LEN+1

INC STR+1

IF \_\_65SC02\_\_

BRA \_LOOP\?

ELSE

JMP \_LOOP\?

ENDIF

\_DONE\? STY LEN+0

ENDM

; Copies a null terminated string from one memory

; location to another. The source and destination

; addresses are destroyed during the copy process.

;

; On exit: A & Y = ??, X is unchanged.

\_STRCPY MACRO SRC,DST

LDY #0

\_LOOP\? LDA (SRC),Y

STA (DST),Y

BEQ \_DONE\?

INY

BNE \_LOOP\?

INC SRC+1

INC DST+1

IF \_\_65SC02\_\_

BRA \_LOOP\?

ELSE

JMP \_LOOP\?

ENDIF

\_DONE\? EQU \*

ENDM

;

\_STRCMP MACRO VLA,VLB

LDY #0

\_LOOP\? LDA (VLA),Y

CMP (VLB),Y

BNE \_DONE\?

LDA (VLA),Y

BEQ \_DONE\?

INY

BNE \_LOOP\?

INC VLA+1

INC VLB+1

IF \_\_65SC02\_\_

BRA \_LOOP\?

ELSE

JMP \_LOOP\?

ENDIF

\_DONE\? EQU \*

ENDM

;

\_STRNCMP MACRO VLA,VLB,LEN

ENDM

LIST

The following source files are useful for developing BBC Microcomputer applications or ROMs.

|  |  |
| --- | --- |
| [bbc.inc](https://www.nesdev.org/obelisk-6502-guide/files/bbc.inc) | Symbol definitions for all the standard vector locations and operating system entry points. |

PAGE

;--------------------------------------------------------

; BBC Microcomputer Machine Operating System (MOS) Defs.

;--------------------------------------------------------

; Copyright (C),1998 Andrew John Jacobs.

; All rights reserved.

;--------------------------------------------------------

;--------------------------------------------------------

; Revision History:

;

;

;

;--------------------------------------------------------

;--------------------------------------------------------

; Operating System Vector Addresses

;--------------------------------------------------------

USERV EQU $0200

BRKV EQU $0202

IRQ1V EQU $0204

IRQ2V EQU $0206

CLIV EQU $0208

BYTEV EQU $020A

WORDV EQU $020C

WRCHV EQU $020E

RDCHV EQU $0210

FILEV EQU $0212

ARGSV EQU $0214

BGETV EQU $0216

BPUTV EQU $0218

GPBPV EQU $021A

FINDV EQU $021C

FSCV EQU $021E

EVENTV EQU $0220

UPTV EQU $0222

NETV EQU $0224

VDUV EQU $0226

KEYV EQU $0228

INSV EQU $022A

REMV EQU $022C

CNPV EQU $022E

IND1V EQU $0230

IND2V EQU $0232

IND3V EQU $0234

;--------------------------------------------------------

; Operating System Entry Points

;--------------------------------------------------------

OSFIND EQU $FFCE

OSGPBP EQU $FFD1

OSBPUT EQU $FFD4

OSBGET EQU $FFD7

OSARGS EQU $FFDA

OSFILE EQU $FFDD

OSRDCH EQU $FFE0

OSASCI EQU $FFE3

OSNEWL EQU $FFE7

OSWRCH EQU $FFEE

OSBYTE EQU $FFF1

OSWORD EQU $FFF4

OSCLI EQU $FFF7

|  |  |
| --- | --- |
| [debug.inc](https://www.nesdev.org/obelisk-6502-guide/files/debug.inc) | A useful debugging routine that prints all the registers and status flags. |

PAGE

;--------------------------------------------------------

; Debugging Utility Functions

;--------------------------------------------------------

; Copyright (C),1999 Andrew John Jacobs

; All rights reserved.

;--------------------------------------------------------

; Prints the address of the next instruction to be

; executed, the processor status flags and the values in

; all the registers.

DUMP\_REGISTERS PHP ;Push everything

PHA

TXA

PHA

TYA

PHA

TSX ;Get stack top

LDA #'P' ;Dump PC

JSR OSWRCH

LDA #'C'

JSR OSWRCH

JSR DUMP\_EQUALS

LDA $0105,X

LDY $0106,X

CLC ;Correct return

ADC #1 ; address

BCC \*+3

INY

JSR DUMP\_WORD

JSR DUMP\_SPACE ;Dump status flags

LDY #0

LDA $0104,X

DUMP\_STATUS ROL A

PHA

LDA DUMP\_BITS,Y

BCS \*+4

LDA #'.'

JSR OSWRCH

PLA

INY

CPY #8

BNE DUMP\_STATUS

JSR DUMP\_SPACE ;Dump A

LDA #'A'

JSR OSWRCH

JSR DUMP\_EQUALS

LDA $0103,X

JSR DUMP\_BYTE

JSR DUMP\_SPACE ;Dump X

LDA #'X'

JSR OSWRCH

JSR DUMP\_EQUALS

LDA $0102,X

JSR DUMP\_BYTE

JSR DUMP\_SPACE ;Dump Y

LDA #'Y'

JSR OSWRCH

JSR DUMP\_EQUALS

LDA $0101,X

JSR DUMP\_BYTE

JSR DUMP\_SPACE ;Dump S

LDA #'S'

JSR OSWRCH

JSR DUMP\_EQUALS

CLC

TXA

ADC #6

JSR DUMP\_BYTE

JSR OSNEWL

PLA ;Restore everything

TAY

PLA

TAX

PLA

PLP

RTS

; Symbolic codes for status register bits

DUMP\_BITS DB 'NV.BDIZC'

; Prints a 16 bit word held in A (LO) & Y (HI) on the

; screen in hexadecimal.

;

; A is are destroyed. X and Y are preserved.

DUMP\_WORD PHA

TYA

JSR DUMP\_BYTE

PLA

; Prints an 8 bit byte held in A on the screen in

; hexadecimal.

;

; A is destroyed. X and Y are preserved.

DUMP\_BYTE PHA

LSR A

LSR A

LSR A

LSR A

JSR DUMP\_NYBBLE

PLA

AND #$0F

; Converts the 4 bit nybble in A to an ASCII character

; representing its hexadecimal value and prints it.

;

; A is destroyed. X and Y are preserved.

DUMP\_NYBBLE ORA #'0'

CMP #'9'+1

BCC \*+4

ADC #6

JMP OSWRCH

; Utility routines to print space and equals characters.

;

; A is destroyed. X and Y are preserved.

DUMP\_SPACE LDA #' '

BNE \*+4

DUMP\_EQUALS LDA #'='

JMP OSWRCH