RTF65003 Reference Guide

2014

robfinch@finitron.ca

Finitron

5/13/2014

# RTF65003

Table of Contents

[RTF65003 1](#_Toc388435099)

[Compatibility 6](#_Toc388435100)

[Mode Switching 6](#_Toc388435101)

[General Registers 7](#_Toc388435102)

[Special Purpose Registers 7](#_Toc388435103)

[Addressing Modes 8](#_Toc388435104)

[Memory Addressing 8](#_Toc388435105)

[Vectors 9](#_Toc388435106)

[Native mode vector table: 9](#_Toc388435107)

[Instruction Set Summary 10](#_Toc388435108)

[ADD 10](#_Toc388435109)

[SUB 11](#_Toc388435110)

[CMP 11](#_Toc388435111)

[AND 12](#_Toc388435112)

[BIT 12](#_Toc388435113)

[OR 13](#_Toc388435114)

[EOR 13](#_Toc388435115)

[Load and Store Instructions 14](#_Toc388435116)

[LD 14](#_Toc388435117)

[LDA 14](#_Toc388435118)

[LDX 15](#_Toc388435119)

[LDY 15](#_Toc388435120)

[ST 16](#_Toc388435121)

[STA 16](#_Toc388435122)

[STX 17](#_Toc388435123)

[STY 17](#_Toc388435124)

[STZ 17](#_Toc388435125)

[Shift Operations / Read-modify-write memory operations. 18](#_Toc388435126)

[ASL 18](#_Toc388435127)

[ROL 18](#_Toc388435128)

[LSR 18](#_Toc388435129)

[ROR 18](#_Toc388435130)

[INC 19](#_Toc388435131)

[DEC 19](#_Toc388435132)

[Bitmap Instructions 20](#_Toc388435133)

[BMS 20](#_Toc388435134)

[BMC 20](#_Toc388435135)

[BMF 20](#_Toc388435136)

[BMT 20](#_Toc388435137)

[Semaphore Instructions 21](#_Toc388435138)

[SPL 21](#_Toc388435139)

[Short Form Instructions 22](#_Toc388435140)

[LDA 22](#_Toc388435141)

[STA 22](#_Toc388435142)

[CMP 22](#_Toc388435143)

[PHA 22](#_Toc388435144)

[PLA 22](#_Toc388435145)

[LDX 23](#_Toc388435146)

[STX 23](#_Toc388435147)

[CPX 23](#_Toc388435148)

[INX 23](#_Toc388435149)

[DEX 23](#_Toc388435150)

[PHX 23](#_Toc388435151)

[PLX 23](#_Toc388435152)

[LDY 24](#_Toc388435153)

[STY 24](#_Toc388435154)

[CPY 24](#_Toc388435155)

[INY 24](#_Toc388435156)

[DEY 24](#_Toc388435157)

[PHY 24](#_Toc388435158)

[PLY 24](#_Toc388435159)

[Flow Control 25](#_Toc388435160)

[JMP 25](#_Toc388435161)

[JSR 25](#_Toc388435162)

[RTS 25](#_Toc388435163)

[RTI 25](#_Toc388435164)

[BRA 26](#_Toc388435165)

[BEQ 26](#_Toc388435166)

[BNE 26](#_Toc388435167)

[BPL 26](#_Toc388435168)

[BMI 26](#_Toc388435169)

[BVS 26](#_Toc388435170)

[BVC 26](#_Toc388435171)

[BCS 26](#_Toc388435172)

[BCC 26](#_Toc388435173)

[BHI 26](#_Toc388435174)

[BLS 26](#_Toc388435175)

[BLT 26](#_Toc388435176)

[BLE 26](#_Toc388435177)

[BGT 26](#_Toc388435178)

[BGE 26](#_Toc388435179)

[BEQ 26](#_Toc388435180)

[BNE 26](#_Toc388435181)

[BPL 26](#_Toc388435182)

[BMI 26](#_Toc388435183)

[BVS 26](#_Toc388435184)

[BVC 26](#_Toc388435185)

[BCS 26](#_Toc388435186)

[BCC 26](#_Toc388435187)

[BRL 26](#_Toc388435188)

[BSR 27](#_Toc388435189)

[BRK 27](#_Toc388435190)

[NOP 27](#_Toc388435191)

[WAI 27](#_Toc388435192)

[STP 27](#_Toc388435193)

[Stack push and pop operations. 28](#_Toc388435194)

[PHP 28](#_Toc388435195)

[PHA 28](#_Toc388435196)

[PHX 28](#_Toc388435197)

[PHY 28](#_Toc388435198)

[PUSH R4 28](#_Toc388435199)

[PUSH 28](#_Toc388435200)

[PUSHA 28](#_Toc388435201)

[PLP 28](#_Toc388435202)

[PLA 28](#_Toc388435203)

[PLX 28](#_Toc388435204)

[PLY 28](#_Toc388435205)

[POP R4 28](#_Toc388435206)

[POP 28](#_Toc388435207)

[POPA 28](#_Toc388435208)

[Status Register Operations 28](#_Toc388435209)

[CLC 28](#_Toc388435210)

[SEC 28](#_Toc388435211)

[CLV 28](#_Toc388435212)

[CLI 28](#_Toc388435213)

[SEI 28](#_Toc388435214)

[CLD 28](#_Toc388435215)

[SED 28](#_Toc388435216)

[XCE 28](#_Toc388435217)

[Register to register transfer short forms. 29](#_Toc388435218)

[TAX 29](#_Toc388435219)

[TXA 29](#_Toc388435220)

[TAY 29](#_Toc388435221)

[TYA 29](#_Toc388435222)

[TAS 29](#_Toc388435223)

[TSA 29](#_Toc388435224)

[TYX 29](#_Toc388435225)

[TXY 29](#_Toc388435226)

[TSX 29](#_Toc388435227)

[TXS 29](#_Toc388435228)

[TRS 29](#_Toc388435229)

[TSR 29](#_Toc388435230)

[EXEC 29](#_Toc388435231)

[ATNI 29](#_Toc388435232)

[String Operations 30](#_Toc388435233)

[MVN 30](#_Toc388435234)

[MVP 30](#_Toc388435235)

[MVC 30](#_Toc388435236)

[CMPS 30](#_Toc388435237)

[Prefix Bytes 31](#_Toc388435238)

[BYTE 31](#_Toc388435239)

[UBYTE 31](#_Toc388435240)

[CHAR 31](#_Toc388435241)

[UCHAR 31](#_Toc388435242)

## Compatibility

The core is backwards compatible with the 65C02 and 65C816 instruction sets via processor emulation modes.

## Mode Switching

To switch to 65C02 emulation mode issue the instructions SEC, XCE.

To switch to 65C816 emulation mode issue the instructions CLC, XCE.

To switch to RTF65003 mode issue the instruction WDM, XCE.

## General Registers

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Usage |
| R0 | z | This register is always zero |  |
| R1 | acc | Accumulator | First parameter / return value |
| R2 | x | ‘x’ index register | Second parameter / Loop counter |
| R3 | y | ‘y’ index register | Third parameter |
| R4 |  |  |  |
| R5 |  |  |  |
| R6 |  |  |  |
| R7 |  |  |  |
| R8 |  |  |  |
| R9 |  |  |  |
| R10 |  |  |  |
| R11 |  |  |  |
| R12 |  |  |  |
| R13 |  |  |  |
| R14 |  |  |  |
| R15 |  |  |  |
|  |  |  |  |

## Special Purpose Registers

|  |  |  |  |
| --- | --- | --- | --- |
| Code |  |  |  |
| 0 | cc | Cache control register |  |
|  | ice | Bit 0 = instruction cache enable | |
| dce | Bit 1 = Data cache enable | |
| dcwa | Bit 2 = Data cache write allocate policy (1=allocate) | |
| … |  |  |  |
| 3 | ph | Multiplier product high order 32 bits | |
| 4 | tick | Tick counter | Read only |
| 5 | lfsr | Linear feedback shift register | Pseudo random |
| … |  |  | |
| 7 | abs8 | The upper eight bits of this register set the 16MB bank that the emulated processor modes reside within. | |
| 8 | vbr | Vector base register – the lower 9 bits are always zero  Bit 0 = interrupt policy (1=native mode for all external interrupts) | |
| 14 | sp8 | 8/16 bit mode stack pointer |  |
| 15 | sp | Native mode Stack pointer |  |
|  |  |  |  |

## Addressing Modes

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Clock Cycles |  |
| Rn | Register direct | 2 |  |
| #imm4 | Four bit sign extended immediate | 2 |  |
| #imm8 | Eight bit sign extended immediate | 2 |  |
| #imm16 | Sixteen bit sign extended immediate | 2 |  |
| #imm32 | Thirty-two bit immediate | 2 |  |
| zp,Rb | Zero page indexed | 4 |  |
| (zp,Rb) | Zero page indexed indirect (inner indexed) | 5 |  |
| (zp),Rb | Zero page indirect indexed (outer indexed) | 6 |  |
| abs | absolute | 4 |  |
| abs,Rb | Absolute indexed | 4 |  |
| (Rb) | Register indirect | 4 |  |
| Label | Relative branch | 2 |  |

## Memory Addressing

The cpu is word oriented with byte addressable memory. The default size of a memory operand is word size (32 bits). This may be overridden using size prefixes for byte (8 bit) or character (16 bit) data. Words and characters must be aligned on appropriate boundaries. Up to 4GB of data are supported and 4GB of code.Interrupt Handling

IF the NMOI bit is set in the VBR register, then the processor will automatically switch to native (32 bit) mode for interrupt handling. Upon executing an RTI instruction the processor will switch back to the mode it was in when the interrupt occurred.

# Vectors

The processor vectors through $0FFFFFFF8 on a reset, and $0FFFFFFF4 on a non-maskable interrupt. In 32 bit mode, or with native mode on interrupt set, all other vectoring is done through a vector table. The vector table allows for 512 entries. The vector table base address is established by the VBR register. During an external IRQ the processor looks at a vector number bus to determine the vector to use for the IRQ. This vector number may be hard-coded in which case all IRQ’s will be vectored to the same location.

In eight bit mode the IRQ/BRK vectors through the usual eight bit IRQ vector at $0FFFE.

### Native mode vector table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vector Number | Usage / Description | |  |  |
| 0 | BREAK instruction vector | |  |  |
| 1 | SLEEP vector (branch to self) | |  |  |
| 2 | Task reschedule interrupt | |  |  |
| 3 | Spinlock interrupt | |  |  |
| 4 | System call | |  |  |
| … |  | |  |  |
|  |  | |  |  |
| 448 | Spurious interrupt | |  |  |
| 449 | IRQ level 1 | 1000 Hz interrupt |  |  |
| 450 | IRQ level 2 | 100 Hz interrupt |  |  |
| … | Other IRQ levels |  |  |  |
| 463 | IRQ level 15 | keyboard interrupt |  |  |
| … |  | |  |  |
| 495 | Unimplemented instruction | |  |  |
| … |  | |  |  |
| 508 | Bus error – data load / store | |  |  |
| 509 | Bus error – instruction fetch | |  |  |
| 511 |  | |  |  |

# Instruction Set Summary

The cycle counts are assuming no wait states are required for either instructions or data and both instructions and data can be found in the cache.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADD | | Flags: v c n z | | | | | | | | | Bytes | Cyc |
| Byte 7 | Byte 6 | | Byte5 | Byte 4 | Byte 3 | | Byte 2 | | Byte 1 |  |  |  |
|  | | | | | 0 | Rt | Rb | Ra | 02h | ADD Rt,Ra,Rb | 3 | 2 |
|  | | | | | | | Rt | Ra | 77h | ADD Rt,Ra | 2 | 2 |
|  | | | | | | | Imm4 | Rt | 67h | ADD Rt,#imm4 | 2 | 2 |
|  | | | | | Imm8 | | Rt | Ra | 65h | ADD Rt,Ra,#imm8 | 3 | 2 |
|  | | | | Imm16 | | | Rt | Ra | 79h | ADD Rt,Ra,#imm16 | 4 | 2 |
|  | | Imm32 | | | | | Rt | Ra | 69h | ADD Rt,Ra,#imm32 | 6 | 2 |
|  | | | | Addr12 | | Rt | Rb | Ra | 75h | ADD Rt,Ra,zp,Rb | 4 | 4 |
|  | | | | Addr12 | | Rt | Rb | Ra | 61h | ADD Rt,Ra,(zp,Rb) | 4 | 5 |
|  | | | | Addr12 | | Rt | Rb | Ra | 71h | ADD Rt,Ra,(zp),Rb | 4 | 6 |
|  | | Addr32 | | | | | Rt | Ra | 6Dh | ADD Rt,Ra,abs | 6 | 4 |
| Addr32 | | | | | ~4 | Rt | Rb | Ra | 7Dh | ADD Rt,Ra,abs,Rb | 7 | 4 |
|  | | | | | ~4 | Rt | Rb | Ra | 72h | ADD Rt,Ra,(Rb) | 3 | 4 |
|  | | | | | Disp8 | | Rt | Ra | 63h | ADD Rt,Ra,d,sp | 3 | 4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SUB | Flags: v c n z | | | | | | | | | | | | Bytes | Cyc |
|  | |  | | | 1 | | | Rt | Rb | Ra | 02h | SUB Rt,Ra,Rb | 3 | 2 |
|  | |  |  | | | Imm8 | | | Rt | Ra | E5h | SUB Rt,Ra,#imm8 | 3 | 2 |
|  | | | | | | | | | Rt | Ra | F7h | SUB Rt,Ra | 2 | 2 |
|  | | | | | | | | | Imm4 | Rt | E7h | SUB Rt,#imm4 | 2 | 2 |
|  | |  | Imm16 | | | | | | Rt | Ra | F9h | SUB Rt,Ra,#imm16 | 4 | 2 |
|  | Imm32 | | | | | | | | Rt | Ra | E9h | SUB Rt,Ra,#imm32 | 6 | 2 |
|  | |  | Addr12 | | | | Rt | | Rb | Ra | F5h | SUB Rt,Ra,zp,Rb | 4 | 4 |
|  | |  | Addr12 | | | | Rt | | Rb | Ra | E1h | SUB Rt,Ra, (zp,Rb) | 4 | 5 |
|  | |  | Addr12 | | | | Rt | | Rb | Ra | F1h | SUB Rt,Ra, (zp),Rb | 4 | 6 |
|  | Addr32 | | | | | | | | Rt | Ra | EDh | SUB Rt,Ra,abs | 6 | 4 |
| Addr32 | | | | | |  | Rt | | Rb | Ra | FDh | SUB Rt,Ra,abs,Rb | 7 | 4 |
|  | | | | | ~4 | | | Rt | Rb | Ra | F2h | SUB Rt,Ra, (Rb) | 3 | 4 |
|  | | | | Disp8 | | | | | Rt | Ra | E3h | SUB Rt,Ra,d,sp | 3 | 4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CMP | Flags: c n z | | | | | | | | | | | | Bytes | |
|  | |  | | 1 | | | 0 | Rb | Ra | | 02h | CMP Ra,Rb | 3 | |
|  | |  |  | | Imm8 | | | 0 | Ra | | E5h | CMP Ra,#imm8 | 3 | |
|  | |  | Imm16 | | | | | 0 | Ra | | F9h | CMP Ra,#imm16 | 4 | |
|  | Imm32 | | | | | | | 0 | Ra | | E9h | CMP Ra,#imm32 | 6 | |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | F5h | CMP Ra,zp,Rb | 4 | |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | E1h | CMP Ra, (zp,Rb) | 4 | |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | F1h | CMP Ra, (zp),Rb | 4 | |
|  | Addr32 | | | | | | | 0 | Ra | | EDh | CMP Ra,abs | 6 | |
| Addr32 | | | | |  | 0 | | Rb | Ra | | FDh | CMP Ra,abs,Rb | 7 | |
|  | | | | ~4 | | | 0 | Rb | Ra | | F2h | CMP Ra, (Rb) | 3 | |
|  | | | | | Disp8 | | | 0 | | Ra | E3h | CMP Ra,d,sp | 3 |  |
|  | | | | | | | | Imm8 | | | C5h | CMP #imm8 | 2 | |

CMP is an alternate mnemonic for SUB where the target register is R0. CMP does not affect the overflow flag.

* Opcode C5h compares the accumulator to a sign extended eight bit value.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AND | Flags: n z | | | | | | | | | | | | Bytes |
|  | |  | | 3 | | | Rt | Rb | Ra | | 02h | AND Rt,Ra,Rb | 3 |
|  | |  |  | |  | | | Rt | Ra | | 37h | AND Rt,Ra | 2 |
|  | |  |  | |  | | | Imm4 | Rt | | 27h | AND Rt,#imm4 | 2 |
|  | |  |  | | Imm8 | | | Rt | Ra | | 25h | AND Rt,Ra,#imm8 | 3 |
|  | |  | Imm16 | | | | | Rt | Ra | | 39h | AND Rt,Ra,#imm16 | 4 |
|  | Imm32 | | | | | | | Rt | Ra | | 29h | AND Rt,Ra,#imm32 | 6 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 35h | AND Rt,Ra,zp,Rb | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 21h | AND Rt,Ra, (zp,Rb) | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 31h | AND Rt,Ra, (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | Ra | | 2Dh | AND Rt,Ra,abs | 6 |
| Addr32 | | | | |  | Rt | | Rb | Ra | | 3Dh | AND Rt,Ra,abs,Rb | 7 |
|  | | | | ~4 | | | Rt | Rb | Ra | | 32h | AND Rt,Ra, (Rb) | 3 |
|  | | | | | Disp8 | | | Rt | | Ra | 23h | AND Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BIT | Flags: v n z | | | | | | | | | | | | Bytes |
|  | |  | | 3 | | | 0 | Rb | Ra | | 02h | BIT Ra,Rb | 3 |
|  | |  |  | | Imm8 | | | 0 | Ra | | 25h | BIT Ra,#imm8 | 3 |
|  | |  | Imm16 | | | | | 0 | Ra | | 39h | BIT Ra,#imm16 | 4 |
|  | Imm32 | | | | | | | 0 | Ra | | 29h | BIT Ra,#imm32 | 6 |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | 35h | BIT Ra,zp,Rb | 4 |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | 21h | BIT Ra, (zp,Rb) | 4 |
|  | |  | Addr12 | | | 0 | | Rb | Ra | | 31h | BIT Ra, (zp),Rb | 4 |
|  | Addr32 | | | | | | | 0 | Ra | | 2Dh | BIT Ra,abs | 6 |
| Addr32 | | | | |  | 0 | | Rb | Ra | | 3Dh | BIT Ra,abs,Rb | 7 |
|  | | | | ~4 | | | 0 | Rb | Ra | | 32h | BIT Ra, (Rb) | 3 |
|  | | | | | Disp8 | | | Rt | | Ra | 23h | BIT Ra,d,sp | 3 |

Bit is the AND operation with no target register; the overflow status is set to bit 30 of the memory op

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OR | Flags: n z | | | | | | | | | | | | Bytes |
|  | |  | | 5 | | | Rt | Rb | Ra | | 02h | OR Rt,Ra,Rb | 3 |
|  | |  |  | |  | | | Rt | Ra | | 17h | OR Rt,Ra | 2 |
|  | |  |  | |  | | | Imm4 | Rt | | 07h | OR Rt,#imm4 | 2 |
|  | |  |  | | Imm8 | | | Rt | Ra | | 05h | OR Rt,Ra,#imm8 | 3 |
|  | |  | Imm16 | | | | | Rt | Ra | | 19h | OR Rt,Ra,#imm16 | 4 |
|  | Imm32 | | | | | | | Rt | Ra | | 09h | OR Rt,Ra,#imm32 | 6 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 15h | OR Rt,Ra,zp,Rb | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 01h | OR Rt,Ra, (zp,Rb) | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 11h | OR Rt,Ra, (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | Ra | | 0Dh | OR Rt,Ra,abs | 6 |
| Addr32 | | | | |  | Rt | | Rb | Ra | | 1Dh | OR Rt,Ra,abs,Rb | 7 |
|  | | | | ~4 | | | Rt | Rb | Ra | | 12h | OR Rt,Ra, (Rb) | 3 |
|  | | | | | Disp8 | | | Rt | | Ra | 03h | OR Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EOR | Flags: n z | | | | | | | | | | | | Bytes |
|  | |  | | 4 | | | Rt | Rb | Ra | | 02h | EOR Rt,Ra,Rb | 3 |
|  | |  |  | |  | | | Rt | Ra | | 57h | EOR Rt,Ra | 2 |
|  | |  |  | |  | | | Imm4 | Rt | | 47h | EOR Rt,#imm4 | 2 |
|  | |  |  | | Imm8 | | | Rt | Ra | | 45h | EOR Rt,Ra,#imm8 | 3 |
|  | |  | Imm16 | | | | | Rt | Ra | | 59h | EOR Rt,Ra,#imm16 | 4 |
|  | Imm32 | | | | | | | Rt | Ra | | 49h | EOR Rt,Ra,#imm32 | 6 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 55h | EOR Rt,Ra,zp,Rb | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 41h | EOR Rt,Ra, (zp,Rb) | 4 |
|  | |  | Addr12 | | | Rt | | Rb | Ra | | 51h | EOR Rt,Ra, (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | Ra | | 4Dh | EOR Rt,Ra,abs | 6 |
| Addr32 | | | | |  | Rt | | Rb | Ra | | 5Dh | EOR Rt,Ra,abs,Rb | 7 |
|  | | | | ~4 | | | Rt | Rb | Ra | | 52h | EOR Rt,Ra, (Rb) | 3 |
|  | | | | | Disp8 | | | Rt | | Ra | 43h | EOR Rt,Ra,d,sp | 3 |

## Load and Store Instructions

Arithmetic and logical instructions can take a memory operand as the third operand of the instruction. This effectively turns all these instructions into load instructions. Hence there isn’t an explicit load instruction. The OR or EOR instructions can readily be used to perform a load operation.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LD | Flags: n z | | | | | | | | | | | | Bytes |
|  | |  |  | | Imm8 | | | Rt | 0 | | 05h | LD Rt,#imm8 | 3 |
|  | |  | Imm16 | | | | | Rt | 0 | | 19h | LD Rt,#imm16 | 4 |
|  | Imm32 | | | | | | | Rt | 0 | | 09h | LD Rt,#imm32 | 6 |
|  | |  | Addr12 | | | Rt | | Rb | 0 | | 15h | LD Rt,zp,Rb | 4 |
|  | |  | Addr12 | | | Rt | | Rb | 0 | | 01h | LD Rt, (zp,Rb) | 4 |
|  | |  | Addr12 | | | Rt | | Rb | 0 | | 11h | LD Rt, (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | 0 | | 0Dh | LD Rt,abs | 6 |
| Addr32 | | | | |  | Rt | | Rb | 0 | | 1Dh | LD Rt,abs,Rb | 7 |
|  | | | | ~4 | | | Rt | Rb | 0 | | 12h | LD Rt, (Rb) | 3 |
|  | | | | | Disp8 | | | Rt | | 0 | 03h | LD Rt,Ra,d,sp | 3 |
|  | | | | | | | | Rt | Ra | | 7Bh | LD Rt,Ra | 2 |

LD is an alternate mnemonic for the OR instruction where register Ra is zero.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDA | Flags: n z | | | | | | | | | | | | Bytes |
|  | |  |  | | Imm8 | | | 1 | 0 | | 05h | LDA #imm8 | 3 |
|  | |  | Imm16 | | | | | 1 | 0 | | 19h | LDA #imm16 | 4 |
|  | Imm32 | | | | | | | 1 | 0 | | 09h | LDA #imm32 | 6 |
|  | |  | Addr12 | | | 1 | | Rb | 0 | | 15h | LDA zp,Rb | 4 |
|  | |  | Addr12 | | | 1 | | Rb | 0 | | 01h | LDA (zp,Rb) | 4 |
|  | |  | Addr12 | | | 1 | | Rb | 0 | | 11h | LDA (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | 0 | | 0Dh | LDA abs | 6 |
| Addr32 | | | | |  | 1 | | Rb | 0 | | 1Dh | LDA abs,Rb | 7 |
|  | | | | ~4 | | | 1 | Rb | 0 | | 12h | LDA (Rb) | 3 |
|  | | | | | Disp8 | | | 1 | | 0 | 03h | LDA d,sp | 3 |

LDA is an alternate mnemonic for the OR instruction where register Ra is zero and register Rt is one.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDX | Flags: n z | | | | | | | | | | | Bytes |
|  | |  |  | | Imm8 | | | 2 | 0 | 05h | LDX #imm8 | 3 |
|  | |  | Imm16 | | | | | 2 | 0 | 19h | LDX #imm16 | 4 |
|  | Imm32 | | | | | | | 2 | 0 | 09h | LDX #imm32 | 6 |
|  | |  | Addr12 | | | 2 | | Rb | 0 | 15h | LDX zp,Rb | 4 |
|  | |  | Addr12 | | | 2 | | Rb | 0 | 01h | LDX (zp,Rb) | 4 |
|  | |  | Addr12 | | | 2 | | Rb | 0 | 11h | LDX (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | 0 | 0Dh | LDX abs | 6 |
| Addr32 | | | | |  | 2 | | Rb | 0 | 1Dh | LDX abs,Rb | 7 |
|  | | | | ~4 | | | 2 | Rb | 0 | 12h | LDX (Rb) | 3 |

LDX is an alternate mnemonic for the OR instruction where register Ra is zero and register Rt is two.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDY | Flags: n z | | | | | | | | | | | Bytes |
|  | |  |  | | Imm8 | | | 3 | 0 | 05h | LDY #imm8 | 3 |
|  | |  | Imm16 | | | | | 3 | 0 | 19h | LDY #imm16 | 4 |
|  | Imm32 | | | | | | | 3 | 0 | 09h | LDY #imm32 | 6 |
|  | |  | Addr12 | | | 3 | | Rb | 0 | 15h | LDY zp,Rb | 4 |
|  | |  | Addr12 | | | 3 | | Rb | 0 | 01h | LDY (zp,Rb) | 4 |
|  | |  | Addr12 | | | 3 | | Rb | 0 | 11h | LDY (zp),Rb | 4 |
|  | Addr32 | | | | | | | Rt | 0 | 0Dh | LDY abs | 6 |
| Addr32 | | | | |  | 3 | | Rb | 0 | 1Dh | LDY abs,Rb | 7 |
|  | | | | ~4 | | | 3 | Rb | 0 | 12h | LDY (Rb) | 3 |

LDY is an alternate mnemonic for the OR instruction where register Ra is zero and register Rt is three.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ST | Flags: none | | | | | | | | | | Bytes |
|  | |  | Addr12 | | 0 | Rb | Ra | | 95h | ST Ra,zp,Rb | 4 |
|  | |  | Addr12 | | 0 | Rb | Ra | | 81h | ST Ra, (zp,Rb) | 4 |
|  | |  | Addr12 | | 0 | Rb | Ra | | 91h | ST Ra, (zp),Rb | 4 |
|  | Addr32 | | | | | 0 | Ra | | 8Dh | ST Ra,abs | 6 |
| Addr32 | | | |  | 0 | Rb | Ra | | 9Dh | ST Ra,abs,Rb | 7 |
|  | | | | | | Rb | Ra | | 92h | ST Ra, (Rb) | 2 |
|  | | | | Disp8 | | 0 | | Ra | 83h | ST Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STA | Flags: none | | | | | | | | | | Bytes |
|  | |  | Addr12 | | 0 | Rb | 1 | | 95h | STA zp,Rb | 4 |
|  | |  | Addr12 | | 0 | Rb | 1 | | 81h | STA (zp,Rb) | 4 |
|  | |  | Addr12 | | 0 | Rb | 1 | | 91h | STA (zp),Rb | 4 |
|  | Addr32 | | | | | 0 | 1 | | 8Dh | STA abs | 6 |
| Addr32 | | | |  | 0 | Rb | 1 | | 9Dh | STA abs,Rb | 7 |
|  | | | | | | Rb | 1 | | 92h | STA (Rb) | 2 |
|  | | | | Disp8 | | 0 | | 1 | 83h | STA Ra,d,sp | 3 |

STA is an alternate mnemonic for ST where the register to be stored is R1 (the accumulator).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STX | Flags: none | | | | | | | | | | Bytes |
|  | |  | Addr12 | | 0 | Rb | 2 | | 95h | STX zp,Rb | 4 |
|  | |  | Addr12 | | 0 | Rb | 2 | | 81h | STX (zp,Rb) | 4 |
|  | |  | Addr12 | | 0 | Rb | 2 | | 91h | STX (zp),Rb | 4 |
|  | Addr32 | | | | | 0 | 2 | | 8Dh | STX abs | 6 |
| Addr32 | | | |  | 0 | Rb | 2 | | 9Dh | STX abs,Rb | 7 |
|  | | | | | | Rb | 2 | | 92h | STX (Rb) | 2 |
|  | | | | Disp8 | | 0 | | 2 | 83h | STX d,sp | 3 |

STX is an alternate mnemonic for ST where the register to be stored is R2 (the x index register). There are also additional short-hand forms for the STX instruction.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STY | Flags: none | | | | | | | | | | Bytes |
|  | |  | Addr12 | | 0 | Rb | 3 | | 95h | STY zp,Rb | 4 |
|  | |  | Addr12 | | 0 | Rb | 3 | | 81h | STY (zp,Rb) | 4 |
|  | |  | Addr12 | | 0 | Rb | 3 | | 91h | STY (zp),Rb | 4 |
|  | Addr32 | | | | | 0 | 3 | | 8Dh | STY abs | 6 |
| Addr32 | | | |  | 0 | Rb | 3 | | 9Dh | STY abs,Rb | 7 |
|  | | | | | | Rb | 3 | | 92h | STY (Rb) | 2 |
|  | | | | Disp8 | | 0 | | 3 | 83h | STY d,sp | 3 |

STY is an alternate mnemonic for ST where the register to be stored is R3 (the y index register). There are also additional short-hand forms for the STY instruction.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STZ | Flags: none | | | | | | | | | | Bytes |
|  | |  | Addr12 | | 0 | Rb | 0 | | 95h | STZ zp,Rb | 4 |
|  | |  | Addr12 | | 0 | Rb | 0 | | 81h | STZ (zp,Rb) | 4 |
|  | |  | Addr12 | | 0 | Rb | 0 | | 91h | STZ (zp),Rb | 4 |
|  | Addr32 | | | | | 0 | 0 | | 8Dh | STZ abs | 6 |
| Addr32 | | | |  | 0 | Rb | 0 | | 9Dh | STZ abs,Rb | 7 |
|  | | | | | | Rb | 0 | | 92h | STZ (Rb) | 2 |
|  | | | | Disp8 | | 0 | | 0 | 83h | STZ d,sp | 3 |

STZ is an alternate mnemonic for ST where the register to be stored is R0.

## Shift Operations / Read-modify-write memory operations.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ASL | Flags: c n z | | | | | | | | | | | Bytes |
|  | |  | |  | | | |  |  | 0Ah | ASL acc | 1 |
|  | |  | |  | | | | Rt | Ra | 06h | ASL Rt,Ra | 2 |
|  | |  | |  | Addr12 | | | | Rb | 16h | ASL zp,Rb | 3 |
|  |  | | Addr32 | | | | | | | 0Eh | ASL abs | 5 |
|  | Addr32 | | | | | | | Rb | 0 | 1Eh | ASL abs,Rb | 6 |
|  |  | | | | Eh | | Rt | Rb | Ra | 02h | ASL Rt,Ra,Rb | 3 |
|  |  | | | | ~3 | Imm5 | | Rt | Ra | 24h | ASL Rt,Ra,#imm8 | 3 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROL | Flags: c n z | | | | | | | | | Bytes |
|  | |  | |  | |  |  | 2Ah | ROL acc | 1 |
|  | |  | |  | | Rt | Ra | 26h | ROL Rt,Ra | 2 |
|  | |  | |  | Addr12 | | Rb | 36h | ROL zp,Rb | 3 |
|  |  | | Addr32 | | | | | 2Eh | ROL abs | 5 |
|  | Addr32 | | | | | Rb | 0 | 3Eh | ROL abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LSR | Flags: c n z | | | | | | | | | | | Bytes |
|  | |  | |  | | | |  |  | 4Ah | LSR acc | 1 |
|  | |  | |  | | | | Rt | Ra | 46h | LSR Rt,Ra | 2 |
|  | |  | |  | Addr12 | | | | Rb | 56h | LSR zp,Rb | 3 |
|  |  | | Addr32 | | | | | | | 4Eh | LSR abs | 5 |
|  | Addr32 | | | | | | | Rb | 0 | 5Eh | LSR abs,Rb | 6 |
|  |  | | | | Fh | | Rt | Rb | Ra | 02h | LSR Rt,Ra,Rb | 3 |
|  |  | | | | ~3 | Imm5 | | Rt | Ra | 34h | LSR Rt,Ra,#imm8 | 3 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROR | Flags: c n z | | | | | | | | | Bytes |
|  | |  | |  | |  |  | 6Ah | ROR acc | 1 |
|  | |  | |  | | Rt | Ra | 66h | ROR Rt,Ra | 2 |
|  | |  | |  | Addr12 | | Rb | 76h | ROR zp,Rb | 3 |
|  |  | | Addr32 | | | | | 6Eh | ROR abs | 5 |
|  | Addr32 | | | | | Rb | 0 | 7Eh | ROR abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INC | Flags: n z | | | | | | | | | Bytes |
|  | |  | |  | |  |  |  |  |  |
|  | |  | |  | | Rt | Ra | E6h | INC Rt,Ra | 2 |
|  | |  | |  | Addr12 | | Rb | F6h | INC zp,Rb | 3 |
|  |  | | Addr32 | | | | | EEh | INC abs | 5 |
|  | Addr32 | | | | | Rb | 0 | FEh | INC abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DEC | Flags: n z | | | | | | | | | Bytes |
|  | |  | |  | |  |  |  |  |  |
|  | |  | |  | | Rt | Ra | C6h | DEC Rt,Ra | 2 |
|  | |  | |  | Addr12 | | Rb | D6h | DEC zp,Rb | 3 |
|  |  | | Addr32 | | | | | CEh | DEC abs | 5 |
|  | Addr32 | | | | | Rb | 0 | DEh | DEC abs,Rb | 6 |

## Bitmap Instructions

The bitmap instructions take a bit number in the accumulator and combine it with the address supplied in the instruction in order to set,clear,flip or test a bit in memory. A size prefix may not be used with bitmap instructions.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BMS | Flags: | | | | | | | | | Bytes |
|  | |  | |  | Addr12 | | Rb | 0642h | BMS zp,Rb | 4 |
|  |  | | Addr32 | | | | | 0E42h | BMS abs | 6 |
|  | Addr32 | | | | | Rb | 0 | 1E42h | BMS abs,Rb | 7 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BMC | Flags: | | | | | | | | | Bytes |
|  | |  | |  | Addr12 | | Rb | 2642h | BMS zp,Rb | 4 |
|  |  | | Addr32 | | | | | 2E42h | BMS abs | 6 |
|  | Addr32 | | | | | Rb | 0 | 3E42h | BMS abs,Rb | 7 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BMF | Flags: | | | | | | | | | Bytes |
|  | |  | |  | Addr12 | | Rb | 4642h | BMS zp,Rb | 4 |
|  |  | | Addr32 | | | | | 4E42h | BMS abs | 6 |
|  | Addr32 | | | | | Rb | 0 | 5E42h | BMS abs,Rb | 7 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BMT | Flags: n z | | | | | | | | | Bytes |
|  | |  | |  | Addr12 | | Rb | 6642h | BMS zp,Rb | 3 |
|  |  | | Addr32 | | | | | 6E42h | BMS abs | 5 |
|  | Addr32 | | | | | Rb | 0 | 7E42h | BMS abs,Rb | 6 |

## Semaphore Instructions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SPL | Flags: | | | | | | Bytes |
|  |  | Addr32 | | | 8E42h | SPL abs | 6 |
|  | Addr32 | | Rb | 0 | 9E42h | SPL abs,Rb | 7 |

The SPL (spinlock) instruction pauses the processor until the specified memory address is non-zero. Once a non-zero value is detected at the memory address the next instruction is executed. This instruction is meant to be used in implementing semaphores. The SPL instruction polls in a continuous loop which is interruptible. If the SPL instruction polls more than a certain number of times, it generates an interrupt itself, which may allow the operating system to schedule other tasks.

## Short Form Instructions

The short form instructions are provided to improve code density over other encodings. All the short form instructions have analogous longer formats.

Accumulator short-form instructions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDA | Flags: n z | | | | | | | | | | | Bytes | Cyc |
|  | |  | Immediate32 | | | | | | | A9h | LDA #i32 | 5 | 2 |
|  | |  | | | Immediate16 | | | | | B9h | LDA #i16 | 3 | 2 |
|  | |  | | | | | Immediate8 | | | A5h | LDA #i8 | 2 | 2 |
|  | |  | |  | Addr12 | | | | Rb | B5h | LDA zp,Rb | 3 |  |
|  |  | | Addr32 | | | | | | | ADh | LDA abs | 5 |  |
|  | Addr32 | | | | | | Rb | | 0 | BDh | LDA abs,Rb | 6 |  |
| STA | Flags: | | | | | | | | | | | Bytes |  |
|  | |  | |  | Addr12 | | | | Rb | 95h | STA zp,Rb | 3 |  |
|  |  | | Addr32 | | | | | | | 8Dh | STA abs | 5 |  |
|  | Addr32 | | | | | | Rb | 0 | | 9Dh | STA abs,Rn | 6 |  |
| CMP | Flags: c n z | | | | | | | | | | | Bytes |  |
|  | |  |  | | | Immediate8 | | | | C5H | CMP #i8 | 2 |  |
| PHA | Flags: | | | | | | | | | 48h | PHA | 1 |  |
| PLA | Flags: | | | | | | | | | 68h | PLA | 1 |  |

X index register short-form instructions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDX | Flags: n z | | | | | | | | | | Bytes | Cyc |
|  | |  | Immediate32 | | | | | | A2h | LDX #i32 | 5 | 2 |
|  | |  | | | Immediate16 | | | | B2h | LDX #i16 | 3 | 2 |
|  | |  | | | | Immediate8 | | | A6h | LDX #i8 | 2 | 2 |
|  | |  | |  | Addr12 | | | Rb | B6h | LDX zp,Rb | 3 |  |
|  |  | | Addr32 | | | | | | AEh | LDX abs | 5 |  |
|  | Addr32 | | | | | Rb | | 0 | BEh | LDX abs,Rb | 6 |  |
| STX | Flags: | | | | | | | | | | Bytes |  |
|  | |  | |  | Addr12 | | | Rb | 96h | STX zp,Rb | 3 |  |
|  |  | | Addr32 | | | | | | 8Eh | STX abs | 5 |  |
| CPX | Flags: c n z | | | | | | | | | | Bytes |  |
|  | |  | Immediate32 | | | | | | E0h | CPX #i32 | 5 |  |
|  | |  | | | | | Imm8 | | E2h | CPX #i8 | 2 |  |
|  | |  | |  | Addr12 | | | Rb | E4h | CPX zp,Rb | 3 |  |
|  |  | | Addr32 | | | | | | ECh | CPX abs | 5 |  |
| INX | Flags: n z | | | | | | | | E8h | INX | 1 | 2 |
| DEX | Flags: n z | | | | | | | | CAh | DEX | 1 | 2 |
| PHX | Flags: | | | | | | | | DAh | PHX | 1 |  |
| PLX | Flags: | | | | | | | | FAh | PLX | 1 |  |

Y index register short-form instructions.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDY | Flags: n z | | | | | | | | | | Bytes |
|  | |  | Immediate32 | | | | | | A0h | LDY #i32 | 5 |
|  | |  | |  | Addr12 | | | Rb | B4h | LDY zp,Rb | 3 |
|  |  | | Addr32 | | | | | | ACh | LDY abs | 5 |
|  | Addr32 | | | | | Rb | | 0 | BCh | LDY abs,Rb | 6 |
| STY | Flags: | | | | | | | | | | Bytes |
|  | |  | |  | Addr12 | | | Rb | 94h | STY zp,Rb | 3 |
|  |  | | Addr32 | | | | | | 8Ch | STY abs | 5 |
| CPY | Flags: c n z | | | | | | | | | | Bytes |
|  | |  | Immediate32 | | | | | | C0h | CPY #i32 | 5 |
|  | |  | | | | | Immed.8 | | C1h | CPY #i8 | 2 |
|  | |  | |  | Addr12 | | | Rb | C4h | CPY zp,Rb | 3 |
|  |  | | Addr32 | | | | | | CCh | CPY abs | 5 |
| INY | Flags: n z | | | | | | | | C8h | INY | 1 |
| DEY | Flags: n z | | | | | | | | 88h | DEY | 1 |
| PHY | Flags: | | | | | | | | 5Ah | PHY | 1 |
| PLY | Flags: | | | | | | | | 7Ah | PLY | 1 |

## Flow Control

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| JMP | Flags: | | | | | | | | Bytes | Cyc |
|  | |  | Address32 | | | | 5Ch | JMP abs | 5 | 2 |
|  | |  | | Address16 | | | 4Ch | JMP abs | 3 | 2 |
|  | |  | Address32 | | | | 6Ch | JMP (abs) | 5 |  |
|  | |  | Address32 | | | | 7Ch | JMP (abs,x) | 5 |  |
|  | |  |  | | 0 | Ra | D2h | JMP (Rn) | 2 | 2 |
| JSR | Flags: | | | | | | | | Bytes |  |
|  | |  | Address32 | | | | 22h | JSR abs | 5 |  |
|  | |  | | Address16 | | | 20h | JSR abs | 3 |  |
|  | |  | Address32 | | | | FCh | JSR (abs,x) | 5 |  |
|  | |  |  | | 0 | Ra | C2h | JSR (Rn) | 2 |  |
| RTS | Flags: | | | | | | 60h | RTS | 1 |  |
| RTI | Flags: z n c v b d i | | | | | | 40h | RTI | 1 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BRA | Flags: | | | Disp8 | 80h | BRA disp | 2 | 2 |
| BEQ | Flags: | | | Disp8 | F0h | BEQ disp | 2 | 2 |
| BNE | Flags: | | | Disp8 | D0h | BNE disp | 2 | 2 |
| BPL | Flags: | | | Disp8 | 10h | BPL disp | 2 | 2 |
| BMI | Flags: | | | Disp8 | 30h | BMI disp | 2 | 2 |
| BVS | Flags: | | | Disp8 | 70h | BVS disp | 2 | 2 |
| BVC | Flags: | | | Disp8 | 50h | BVC disp | 2 | 2 |
| BCS | Flags: | | | Disp8 | B0h | BCS disp | 2 | 2 |
| BCC | Flags: | | | Disp8 | 90h | BCC disp | 2 | 2 |
| BHI | Flags: | | | Disp8 | 13h | BHI disp | 2 | 2 |
| BLS | Flags: | | | Disp8 | 33h | BLS disp | 2 | 2 |
| BLT | Flags: | | | Disp8 | B3h | BLT disp | 2 | 2 |
| BLE | Flags: | | | Disp8 | F3h | BLE disp | 2 | 2 |
| BGT | Flags: | | | Disp8 | D3h | BGT disp | 2 | 2 |
| BGE | Flags: | | | Disp8 | 93h | BGE disp | 2 | 2 |
| BEQ | Flags: | Disp16 | | 01h | F0h | BEQ disp | 4 | 2 |
| BNE | Flags: | Disp16 | | 01h | D0h | BNE disp | 4 | 2 |
| BPL | Flags: | Disp16 | | 01h | 10h | BPL disp | 4 | 2 |
| BMI | Flags: | Disp16 | | 01h | 30h | BMI disp | 4 | 2 |
| BVS | Flags: | Disp16 | | 01h | 70h | BVS disp | 4 | 2 |
| BVC | Flags: | Disp16 | | 01h | 50h | BVC disp | 4 | 2 |
| BCS | Flags: | Disp16 | | 01h | B0h | BCS disp | 4 | 2 |
| BCC | Flags: | Disp16 | | 01h | 90h | BCC disp | 4 | 2 |
| BRL | Flags: | | Disp16 | | 82h | BRL disp | 3 | 2 |

The range of branches is extended to 16 bits when the displacement is specified as FFh.

In 32 bit mode branch displacements are relative to the address of the instruction following the branch. In 32 bit mode a branch to self instruction, (displacement = -2), triggers the SLEEP interrupt.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| BSR |  | Disp16 | 62h | BSR disp | 3 |  |
| BRK | Flags: b | | 00h | BRK | 1 |  |
| NOP | Flags: | | EAh | NOP | 1 | 2 |
| WAI | Flags: | | CBh | WAI | 1 | 2+ |
| STP | Flags: | | DBh | STP | 1 | 2 |

## Stack push and pop operations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PHP | Flags: | | | 08h | PHP | 1 |
| PHA | Flags: | | | 48h | PHA | 1 |
| PHX | Flags: | | | DAh | PHX | 1 |
| PHY | Flags: | | | 5Ah | PHY | 1 |
| PUSH R4 | Flags: | | | 0Fh | PUSH R4 | 1 |
| PUSH | Flags: | ~4 | Ra | 0Bh | PUSH Ra | 2 |
| PUSHA | Flags: | 0Bh | | 42h | PUSHA | 2 |
| PLP | Flags: z c v n i b d | | | 28h | PLP | 1 |
| PLA | Flags: | | | 68h | PLA | 1 |
| PLX | Flags: | | | FAh | PLX | 1 |
| PLY | Flags: | | | 7Ah | PLY | 1 |
| POP R4 | Flags: | | | 2Fh | POP R4 | 1 |
| POP | Flags: | Rt | ~4 | 2Bh | POP Rt | 2 |
| POPA | Flags: | 2Bh | | 42h | POP Rt | 2 |

## Status Register Operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLC | Flags: c | 18h | CLC | 1 | 2 |
| SEC | Flags: c | 38h | SEC | 1 | 2 |
| CLV | Flags: v | B8h | CLV | 1 | 2 |
| CLI | Flags: i | 58h | CLI | 1 | 2 |
| SEI | Flags: i | 78h | SEI | 1 | 2 |
| CLD | Flags: d | D8h | CLD | 1 | 2 |
| SED | Flags: d | F8h | SED | 1 | 2 |
| XCE | Flags: e,c | FBh | XCE | 1 | 2 |

## Register to register transfer short forms.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TAX | Flags: n z | | | | AAh | TAX | 1 | 2 |
| TXA | Flags: n z | | | | 8Ah | TXA | 1 | 2 |
| TAY | Flags: n z | | | | A8h | TAY | 1 | 2 |
| TYA | Flags: n z | | | | 98h | TYA | 1 | 2 |
| TAS | Flags: | | | | 1Bh | TAS | 1 | 2 |
| TSA | Flags: n z | | | | 3Bh | TSA | 1 | 2 |
| TYX | Flags: n z | | | | BBh | TYX | 1 | 2 |
| TXY | Flags: n z | | | | 9Bh | TXY | 1 | 2 |
| TSX | Flags: n z | | | | BAh | TSX | 1 | 2 |
| TXS | Flags: | | | | 9Ah | TXS | 1 | 2 |
| TRS | Flags: | Spr | Ra | | 8Bh | TRS Ra,spr | 2 | 2 |
| TSR | Flags: n z | Rt | | Spr | ABh | TSR spr,Rt | 2 | 2 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EXEC | Flags: | Rb | Ra | EBh | EXEC Ra,Rb | 2 |

The EXEC instruction executes the instruction contained in the specified register pair.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ATNI | Flags: | Rb | Ra | 4Bh | ATNI Ra,Rb | 2 |

The ATNI instruction adds a 64 bit value specified in a register pair to the next instruction. The opcodes for the next instruction executed are a combination of the instruction from memory added to registers.

## String Operations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MVN | Flags: |  |  | 44h | MVN | 1 |
| MVP | Flags: |  |  | 54h | MVP | 1 |
| MVC | Flags: |  |  | EBh | MVC | 1 |
| CMPS | Flags: c v n z | 44h | | 42h | CMPS | 2 |

MVN – moves a block of memory beginning at low addresses and incrementing addresses and decrementing the accumulator until the accumulator which contains the count is zero. The x register contains the source address, the y register the destination address. Both x and y are incremented during the move, and the accumulator is decremented.

MVP – moves a block of memory beginning at a high address and decrementing until the accumulator is zero.

MVC – moves a constant value - stores the value contained in the .x register to a block of memory specified by the .y register. The accumulator contains the number of words to store. This instruction is useful for initializing a block of memory to a constant value.

CMPS – compares two strings located by the x and y registers until the count in the accumulator expires, or the strings are different. The flags are set based on the comparison of the last pair of words compared.

## Prefix Bytes

Prefix bytes are used to override the default word sized operation for smaller memory operands. The assembler will automatically output prefix bytes when a size for the memory operand is specified. For example: LDA.B $1234 loads and sign extends a byte from memory address $1234.

|  |  |  |
| --- | --- | --- |
|  | Opcode | Description |
| BYTE | 87h | Causes a signed byte load / store |
| UBYTE | A7h | Causes an unsigned byte load / store |
| CHAR | 97h | Cause a signed character (16 bit) load / store |
| UCHAR | B7h | Causes an unsigned char (16 bit) load / store |

Opcode Map – 32 bit mode

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
| 0- | BRK | OR (d,r) | RR | OR d,s | TSB d,r | OR #i8 | ASL r,r | OR #i4 | PHP | OR #i32 | ASL acc | PUSH r | TSB abs | OR abs | ASL abs | PUSH R4 |
| 1- | BPL disp | OR (d),r | OR (r) | BHI | TRB d,r | OR d,r | ASL d,r | OR r | CLC | OR #i16 | INA | TAS | TRB abs | OR abs,r | ASL abs,r | ~ |
| 2- | JSR abs16 | AND (d,r) | JSL abs | AND d,s | ASL #i8 | AND #i8 | ROL r,r | AND #i4 | PLP | AND #i32 | ROL acc | POP r | JSR (abs) | AND abs | ROL abs | POP R4 |
| 3- | BMI disp | AND (d),r | AND (r) | BLS | LSR #i8 | AND d,r | ROL d,r | AND r | SEC | AND #i16 | DEA | TSA | ~ | AND abs,r | ROL abs,r | ~ |
| 4- | RTI | EOR (d,r) | PG2 | EOR d,s | MVP | EOR #i8 | LSR r,r | EOR #i4 | PHA | EOR #i32 | LSR acc | ATNI | JMP abs16 | EOR abs | LSR abs | ~ |
| 5- | BVC disp | EOR (d),r | EOR (r) | ACBR disp | MVN | EOR d,r | LSR d,r | EOR r | CLI | EOR #i16 | PHY | CACHE | JML abs | EOR abs,r | LSR abs,r | ~ |
| 6- | RTS | ADD (d,r) | BSR | ADD d,s | MVC | ADD #i8 | ROR r,r | ADD #i4 | PLA | ADD #i32 | ROR acc | RTL | JMP (abs) | ADD abs | ROR abs | ~ |
| 7- | BVS disp | ADD (d),r | ADD (r) | ~ | ~ | ADD d,r | ROR d,r | ADD r | SEI | ADD #i16 | PLY | LD r | JMP (abs,x) | ADD abs,r | ROR abs,r | ~ |
| 8- | BRA disp | ST (d,r) | BRL disp | ST d,s | ST d,r | SUB sp,#i8 | CMP r,r | BYTE | DEY | SUB sp,#i32 | TXA | TRS | STY abs | STA abs | STX abs | ~ |
| 9- | BCC disp | ST (d),r | ST (r) | BGE | STY d,r | STA d,r | STX d,r | CHAR | TYA | SUB sp,#i16 | TXS | TXY | ST abs | STA abs,r | ST abs,r | ~ |
| A- | LDY #i32 | LDY #i8 | LDX #i32 | ~ | ~ | LDA #i8 | LDX #i8 | UBYTE | TAY | LDA #i32 | TAX | TSR | LDY abs | LDA abs | LDX abs | ~ |
| B- | BCS disp | LDY #i16 | LDX #i16 | BLT | LDY d,r | LDA d,r | LDX d,r | UCHAR | CLV | LDA #i16 | TSX | TYX | LDY abs,r | LDA abs,r | LDX abs,r | ~ |
| C- | CPY #i32 | CPY #i8 | JSR (r) | ~ | CPY d,r | CMP #i8 | DEC r | ~ | INY | CMP #i32 | DEX | WAI | CPY abs | ~ | DEC abs | ~ |
| D- | BNE disp | BXZ | JMP (r) | BGT | ~ |  | DEC d,r | R | CLD | CMP #i16 | PHX | STP | INT #i9 | INT #i9 | DEC abs,r | ~ |
| E- | CPX #i32 | SUB (d,r) | CPX #i8 | SUB d,s | CPX d,r | SUB #i8 | INC r | SUB #i4 | INX | SUB #i32 | NOP | EXEC | CPX abs | SUB abs | INC abs | ~ |
| F- | BEQ disp | SUB (d),r | SUB(r) | BLE | PEA abs | SUB d,r | INC d,r | SUB r | SED | SUB #i16 | PLX | XCE | JSR (abs,x) | SUB abs,r | INC abs,r | ~ |

Register-Register (RR) Instructions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 02 | ADD | SUB | ~ | AND | EOR | OR | ~ | ~ | MUL | MULS | DIV | DIVS | MOD | MODS | ASL | LSR |

Register (R) Instructions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | SXB | SXC | ZXB | ZXC | RBO | NOT | COM |  |  |  |  |  |  |  |  |  |

Opcode Map – 32 bit mode – Page Two (PG2 prefix) Instructions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
| 0- | ~ | MUL (d,r) | RR | MUL d,s |  | MUL #i8 | BMS d,r | ~ | ICOFF | MUL #i32 | ~ | PUSHA | ~ | MUL abs | BMS abs | ~ |
| 1- | ~ | MUL (d),r | MUL (r) | ~ |  | MUL d,r | ~ | ~ | TOFF | MUL #i16 | ~ | ~ | ~ | MUL abs,r | BMS abs,r | ~ |
| 2- | ~ | MULS (d,r) | ~ | MULS d,s |  | MULS #i8 | BMC d,r | ~ | ICON | MULS #i32 | ~ | POPA | ~ | MULS abs | BMC abs | ~ |
| 3- | ~ | MULS  (d),r | MULS (r) | ~ | ~ | MULS d,r | ~ | ~ | TON | MULS #i16 | ~ | ~ | ~ | MULS abs,r | BMC abs,r | ~ |
| 4- | ~ | DIV (d,r) | ~ | DIV d,s | CMPS | DIV #i8 | BMF d,r | ~ | ~ | DIV #i32 | ~ | ~ | ~ | DIV abs | BMF abs | ~ |
| 5- | ~ | DIV (d),r | DIV (r) | ~ | ~ | DIV d,r | ~ | ~ | hoff | DIV #i16 | ~ | ~ | ~ | DIV abs,r | BMF abs,x | ~ |
| 6- | ~ | DIVS (d,r) | ~ | DIVS d,s | ~ | DIVS #i8 | BMT d,r | ~ | ~ | DIVS #i32 | ~ | ~ | ~ | DIVS abs | BMT abs | ~ |
| 7- | ~ | DIVS (d),r | DIVS (r) | ~ | ~ | DIVS d,r | ~ | ~ | ~ | DIVS #i16 | ~ | ~ | ~ | DIVS abs,r | BMT abs,r | ~ |
| 8- | ~ | MOD (d,r) | ~ | MOD d,s | ~ | MOD #i8 | ~ | ~ | ~ | MOD #i32 | ~ | ~ | ~ | MOD abs | SPL abs | ~ |
| 9- | ~ | MOD (d),r | MOD (r) | ~ | ~ | MOD d,r | ~ | ~ | ~ | MOD #i16 | ~ | ~ | ~ | MOD abs,r | SPL abs,r | ~ |
| A- | ~ | MODS (d,r) | ~ | MODS d,s | ~ | MODS #i8 | ~ | ~ | ~ | MODS #i32 | ~ | ~ | ~ | MODS abs | ~ | ~ |
| B- | ~ | MODS (D),r | MODS (r) | ~ | ~ | MODS d,r | ~ | ~ | ~ | MODS #i16 | ~ | ~ | ~ | MODS abs,r | ~ | ~ |
| C- | ~ | LEA (d,r) | ~ | LEA d,s | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | LEA abs | ~ | ~ |
| D- | ~ | LEA (d),r | LEA (r) | ~ | ~ | LEA d,r | ~ | ~ | ~ | ~ | ~ | ~ | ~ | LEA abs,r | ~ | ~ |
| E- | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
| F- | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |

Register-Register (RR) Instructions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4202 |  | DIF | ~ | SGN |  |  | ~ | ~ | SQRT |  |  |  |  |  |  |  |

Opcode Map – 8 bit mode W65C02 compatible

|  |  |
| --- | --- |
|  | = Enhanced instructions not found on the 65C02 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
| 0- | BRK | ORA (d,x) | ~ | ~ | TSB d,r | ORA d | ASL d | ~ | PHP | OR #i8 | ASL acc | ~ | TSB abs | ORA abs | ASL abs | ~ |
| 1- | BPL disp | ORA (d),y | ORA (d) | ~ | TRB d,r | OR d,x | ASL d,x | ~ | CLC | OR abs,y | INA | TAS | TRB abs | ORA abs,x | ASL abs,x | ~ |
| 2- | JSR abs | AND (d,x) | JSL abs24 | ~ | BIT d | AND d | ROL d | ~ | PLP | AND #i8 | ROL acc | ~ | BIT abs | AND abs | ROL abs | ~ |
| 3- | BMI disp | AND (d),y | AND (d) | ~ | BIT d,x | AND d,x | ROL d,x | ~ | SEC | AND abs,y | DEA | TSA | BIT abs,x | AND abs,x | ROL abs,x | ~ |
| 4- | RTI | EOR (d,x) | WDM | ~ | ~ | EOR d | LSR d | ~ | PHA | EOR #i8 | LSR acc | ~ | JMP abs | EOR abs | LSR abs | ~ |
| 5- | BVC disp | EOR (d),y | EOR (d) | ~ | ~ | EOR d,x | LSR d,x | ~ | CLI | EOR abs,y | PHY | ~ | JML abs24 | EOR abs,x | LSR abs,x | ~ |
| 6- | RTS | ADC (d,x) | ~ | ~ | STZ d | ADC d | ROR d | ~ | PLA | ADC #i8 | ROR acc | RTL | JMP (abs) | ADC abs | ROR abs | ~ |
| 7- | BVS disp | ADC (d),y | ADC (d) | ~ | STZ d,x | ADC d,x | ROR d,x | ~ | SEI | ADC abs,y | PLY | ~ | JMP (abs,x) | ADC abs,x | ROR abs,x | ~ |
| 8- | BRA disp | STA (d,x) | BRL disp | ~ | STY d | STA d | STX d | ~ | DEY | BIT # | TXA | ~ | STY abs | STA abs | STX abs | ~ |
| 9- | BCC disp | STA (d),y | STA (d) | ~ | STY d,x | STA d,x | STX d,y | ~ | TYA | STA abs,y | TXS | TXY | STZ abs | STA abs,x | STZ abs,x | ~ |
| A- | LDY #i8 | LDA (d,x) | LDX #i8 | ~ | LDY d | LDA d | LDX d | ~ | TAY | LDA #i8 | TAX | ~ | LDY abs | LDA abs | LDX abs | ~ |
| B- | BCS disp | LDA (d),y | LDA (d) | ~ | LDY d,x | LDA d,x | LDX d,y | ~ | CLV | LDA abs,y | TSX | TYX | LDY abs,x | LDA abs,x | LDX abs,x | ~ |
| C- | CPY #i8 | CMP (d,x) | ~ | ~ | CPY d | CMP d | DEC d | ~ | INY | CMP #i8 | DEX | WAI | CPY abs | CMP abs | DEC abs | ~ |
| D- | BNE disp | CMP (d),y | CMP (d) | ~ | ~ | CMP d,x | DEC d,r | ~ | CLD | CMP abs,y | PHX | STP | ~ | CMP abs,x | DEC abs,x | ~ |
| E- | CPX #i8 | SBC(d,x) | ~ | ~ | CPX d | SUB d | INC d | ~ | INX | SBC #i8 | NOP | ~ | CPX abs | SBC abs | INC abs | ~ |
| F- | BEQ disp | SBC (d),y | SBC(r) | ~ | ~ | SUB d,x | INC d,r | ~ | SED | SBC abs,y | PLX | NAT | JSR (abs,x) | SBC abs,x | INC abs,x | ~ |